LabVIEW™

Manuel de l'utilisateur

NATIONAL INSTRUMENTS™

Edition de juillet 1998
Référence 321190B-01

Filiales francophones

National Instruments Belgium nv
Leuvensesteenweg 613
B-1930 Zaventem

National Instruments Canada–Montréal
1000 Boulevard St. Jean,
Suite 316
Point-Claire, Québec H9R 5P1

National Instruments France
Centre d'Affaires Paris-Nord
Immeuble "Le Continental"
BP 217
93153 Le Blanc-Mesnil Cedex

National Instruments Suisse
Sonnenbergstr. 53
CH-5408 Ennetbaden

Support internet

E-mail : support@natinst.com
Site FTP : ftp.natinst.com
Adresse web : http://www.natinst.com

Support Bulletin Board

BBS France : 01 48 65 15 59
BBS Etats-Unis : 512 794 5422

Support téléphonique en français

Belgique	Tél. : 02 757 00 20	Fax : 02 757 03 11	Tél. : 405 120 (Luxembourg)
Canada (Québec)	Tél. : 514 694 8521	Fax : 514 694 4399	
France	Tél. : 01 48 14 24 24	Fax : 01 48 14 24 14	
Suisse	Tél. : 056 200 51 51	Fax : 056 200 51 55	Tél. : 022 980 05 11 (Genève)

Les filiales

Allemagne 089 741 31 30, Australie 03 9879 5166, Autriche 0662 45 79 90 0, Brésil 011 288 3336, Canada (Ontario) 905 785 0085, Corée 02 596 7456, Danemark 45 76 26 00, Espagne 91 640 0085, Finlande 09 725 725 11, Hong Kong 2645 3186, Israël 03 6120092, Italie 02 413091, Japon 03 5472 2970, Mexique 5 520 2635, Norvège 32 84 84 00, Pays-Bas 0348 433466, Royaume-Uni 01635 523545, Singapour 2265886, Suède 08 730 49 70, Taiwan 02 377 1200

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759 USA Tél: (+1) 512 794 0100

Informations importantes

Limitations de garantie

Les disquettes ou CD sur lesquels vous recevrez le logiciel de National Instruments sont garantis contre les pannes survenant lors de l'exécution de programmes, qui seraient dues à des défauts matériels ou de fabrication. La période de cette garantie est de 90 jours à partir de la date de livraison, attestée par les reçus ou autres documents. Le cas échéant, National Instruments corrigera ou remplacera la disquette ou le CD qui ne permettrait pas l'exécution normale des programmes, à condition qu'un tel défaut soit stipulé au cours de la période de garantie. National Instruments ne garantit pas que le fonctionnement de ses logiciels ne sera pas interrompu ni se déroulera sans erreur.

Un numéro RMA (Return Material Authorization) d'autorisation de retour de matériel doit être délivré par l'usine et clairement indiqué sur l'emballage du produit afin que celui-ci puisse être accepté pour la réparation sous garantie. National Instruments prendra à sa charge les frais de transport pour renvoyer à son propriétaire les éléments couverts par la garantie.

National Instruments estime avoir fait tout ce qu'il fallait pour que les informations contenues dans ce manuel soient exactes. Ce document a été soigneusement relu pour la précision des informations techniques qu'il contient. Au cas où il resterait malgré tout des erreurs techniques ou des fautes typographiques, National Instruments se réserve le droit d'apporter des modifications aux futures éditions de ce document sans préavis aux détenteurs de cette édition. Le lecteur est prié de prendre contact directement avec National Instruments s'il suspecte des erreurs. National Instruments ne pourra être tenu responsable des problèmes liés à l'utilisation de ce document ou aux informations qu'il contient.

À L'EXCEPTION DE CE QUI EST SPÉCIFIÉ ICI, NATIONAL INSTRUMENTS N'ACCORDE AUCUNE AUTRE GARANTIE, EXPLICITE OU IMPLICITE, ET REJETTE PARTICULIÈREMENT TOUTE GARANTIE LIÉE À L'ACTE DE VENTE ET À L'ADÉQUATION DE SES PRODUITS À UN BESOIN PARTICULIER. LES DROIT DES UTILISATEURS À RECOUVRER LES DOMMAGES CAUSÉS PAR UNE FAUTE OU NÉGLIGENCE DE LA PART DE NATIONAL INSTRUMENTS SERONT LIMITÉS AUX SOMMES VERSÉES PAR L'UTILISATEUR. NATIONAL INSTRUMENTS NE SERA PAS PASSIBLE DE DOMMAGES ET INTÉRÊTS À LA SUITE DE PERTES DE DONNÉES OU DE PROFITS, OU DE DOMMAGES (ACCIDENTELS OU NON) LIÉS À L'UTILISATION DE SES PRODUITS, MÊME S'IL EN AVAIT ÉTÉ PRÉALABLEMENT AVERTI. Cette limitation de la responsabilité de National Instruments s'appliquera quelles que soient la nature et l'origine du préjudice, que ce soit à la suite d'un contrat ou la conséquence d'un acte délictueux, y compris par négligence. Toute action contre National Instruments doit être intentée dans l'année qui suit la cause de cette action. National Instruments ne pourra être tenu responsable de tout retard en performance dû à des causes qui iraient au-delà de ce qu'il lui est raisonnablement possible de faire. La garantie fournie ici ne couvre pas les dommages, défauts, dysfonctionnements ou défauts de service dus à des erreurs commises par l'utilisateur dans l'interprétation des instructions de National Instruments en ce qui concerne l'installation, le fonctionnement et la maintenance. Elle ne couvre pas non plus les négligences, les modifications ou mauvais usages du produit de la part de l'utilisateur, les chutes de tension ou les surtensions, le feu, les inondations, les accidents, les agissements de tierces personnes, et tout autre événement incontrôlable.

Copyright

Marques déposées

MISE EN GARDE CONCERNANT L'UTILISATION DES PRODUITS DE National Instruments DANS LES APPLICATIONS MÉDICALES ET HOSPITALIÈRES

Les produits de National Instruments ne sont pas conçus avec des composants et suivant des méthodes de tests prévus pour assurer un niveau de fiabilité convenant à leur utilisation dans les applications de traitement et de diagnostic sur les personnes. Les applications des produits de National Instruments impliquant des traitements médicaux ou cliniques peuvent potentiellement occasionner des blessures accidentelles à cause d'une panne des produits, ou à cause d'une erreur de la part de l'utilisateur ou du concepteur de l'application. Toute utilisation ou application des produits National Instruments pour ou dans des traitements médicaux ou cliniques doit être effectuée par un personnel médical correctement formé et qualifié, et toutes les garanties médicales d'usage, tous les équipements et toutes les procédures qui sont appropriées à cette situation particulière pour éviter les blessures graves ou la mort, doivent toujours être mis en œuvre lorsque l'on utilise les produits de National Instruments. Les produits de National Instruments N'ONT PAS été conçus pour se substituer à toute forme de procédé, procédure ou équipement utilisée pour la surveillance médicale ou pour garantir la santé publique dans les traitements médicaux ou cliniques.

Table des matières

Avant-propos

Chapitre 1
Introduction

PARTIE I
Introduction à la programmation en G

Chapitre 2
Création de VIs

Chapitre 3
Boucles et graphes déroulants

Chapitre 4
Structure Condition, structure Séquence et boîte de calcul

Chapitre 5
Tableaux, clusters et graphes

Chapitre 6
Chaînes de caractères et E/S sur fichiers

PARTIE II
Interfaces d'E/S

Chapitre 7
Initiation aux drivers d'instruments LabVIEW

Chapitre 8
Tutorial VISA de LabVIEW

Chapitre 9
Introduction aux fonctions GPIB de LabVIEW

Chapitre 10
VIs de port série

Partie III
Analyse

Chapitre 11
Introduction à l'analyse dans LabVIEW

Chapitre 12
Génération de signaux

Chapitre 13
Traitement des signaux numériques

Chapitre 14
Fenêtres de lissage

Chapitre 15
Analyse et mesure de spectre

Chapitre 16
Filtrage

Chapitre 17
Ajustement de courbe

Chapitre 18
Algèbre linéaire

Chapitre 19
Probabilités et statistiques

PARTIE IV
Communication inter-applications et sur réseau

Chapitre 20
Introduction à la communication

Chapitre 21
TCP et UDP

Chapitre 22
Support d'ActiveX

Chapitre 23
Utilisation du protocole DDE

Chapitre 24
AppleEvents

Chapitre 25
Communication programme à programme

Partie V
Programmation en G avancée

Chapitre 26
Personnalisation de VIs

Chapitre 27
Attributs d'objets de la face-avant

Chapitre 28
Conception des programmes

Chapitre 29
Autres ressources disponibles

Annexe A
Références d'analyse

Annexe B
Questions fréquemment posées

Annexe C
Informations à l'attention du client

Glossaire

Index

Figures

Tableaux

Exercices

Avant-propos

Le *Manuel de l'utilisateur LabVIEW* fournit des informations sur la création d'instruments virtuels (VIs). Ce manuel comporte également des informations sur les interfaces via lesquelles vous pouvez entrer et sortir des données en utilisant les VIs LabVIEW pour réaliser des opérations d'analyse. Ce manuel explique également comment LabVIEW gère la communication inter-applications et sur réseau. Lisez les *Notes d'information LabVIEW* avant d'utiliser le *Manuel de l'utilisateur LabVIEW.*

Organisation de ce manuel

Le *Manuel de l'utilisateur LabVIEW* est organisé comme suit.

- Le chapitre 1, *Introduction*, présente l'approche unique avec laquelle LabVIEW aborde la programmation. Il explique également comment commencer à utiliser LabVIEW pour développer des programmes.

Partie I, Introduction à la programmation en G

Cette section contient des informations de base sur la création d'instruments virtuels (VIs), l'utilisation de VIs dans d'autres VIs, les structures de programmation telles que les boucles, et les structures de données telles que les tableaux et les chaînes de caractères.

La Partie 1, *Introduction à la programmation en G*, contient les chapitres suivants.

- Le chapitre 2, *Création de VIs*, explique comment créer un VI comprenant la face-avant, qui est l'interface utilisateur, et le diagramme, qui est le code source. Une fois que vous avez créé un VI, vous pouvez l'utiliser dans d'autres VIs.
- Le chapitre 3, *Boucles et graphes déroulants*, vous montre comment répéter les portions du diagramme en utilisant une boucle While et une boucle For. Ce chapitre explique également comment afficher plusieurs points graphiquement, un à la fois, sur un graphe déroulant.
- Le chapitre 4, *Structure Condition, structure Séquence et boîte de calcul*, explique comment utiliser la structure Condition, qui est une structure conditionnelle, la structure Séquence, qui aide à établir l'ordre d'exécution, et la Boîte de calcul, qui aide à exécuter des formules mathématiques.

- Le chapitre 5, *Tableaux, clusters et graphes*, vous montre comment afficher un groupe ou tableau de points de données sur un graphe. Vous pouvez passer les paramètres d'échelle aussi bien qu'un tableau de points sur un graphe en créant un cluster, qui est un groupe de différents types de données.
- Le chapitre 6, *Chaînes de caractères et E/S sur fichiers*, présente les commandes et les indicateurs de chaînes de caractères ainsi que les opérations d'entrée et de sortie sur fichiers.

Partie II, Interfaces d'E/S

Cette section contient des informations de base sur les interfaces via lesquelles vous pouvez entrer et sortir des données : acquisition de données, GPIB, série et VXI. Reportez-vous au *Manuel de base d'acquisition de données LabVIEW* pour des informations sur l'acquisition de données en temps réel. VISA (Virtual Instrument Software Architecture : architecture logicielle d'instrument virtuel) est une bibliothèque logicielle unique qui sert d'interface avec les instruments GPIB, série et VXI. Les applications LabVIEW développées spécialement pour un instrument spécifique sont appelées drivers d'instruments. National Instruments fournit plusieurs drivers d'instruments utilisant la bibliothèque VISA, mais vous pouvez aussi construire vos propres drivers d'instruments.

La Partie II, *Interfaces d'E/S*, contient les chapitres suivants.

- Le chapitre 7, *Initiation aux drivers d'instruments LabVIEW*, explique comment créer et utiliser des drivers d'instruments National Instruments.
- Le chapitre 8, *Tutorial VISA de LabVIEW*, vous montre comment implémenter des applications VISA courantes en utilisant une communication basée message et basée registre, ainsi que des événements et le verrouillage.
- Le chapitre 9, *Introduction aux fonctions GPIB de LabVIEW*, explique comment fonctionne le GPIB et la différence entre les interfaces IEEE 488 et IEEE 488.2.
- Le chapitre 10, *VIs de port série*, explique les facteurs importants affectant la communication série.

Partie III, Analyse

Cette section contient des informations de base sur l'analyse des données, le traitement et la génération de signal, l'algèbre linéaire, l'ajustement de courbe, la probabilité et les statistiques.

La Partie III, *Analyse*, contient les chapitres suivants.

- Le chapitre 11, *Introduction à l'analyse dans LabVIEW*, introduit des concepts qui s'appliquent à toutes les applications d'analyse, notamment la fonctionnalité supportée, les conventions de notation et d'appellation, et les méthodes d'échantillonnage de signal.
- Le chapitre 12, *Génération de signaux*, explique comment produire des signaux utilisant la fréquence normalisée et comment construire un générateur de fonction simulée.
- Le chapitre 13, *Traitement des signaux numériques*, indique les différences entre la transformée de Fourier rapide (FFT) et la transformée de Fourier discrète (DFT).
- Le chapitre 14, *Fenêtres de lissage*, explique comment l'utilisation de fenêtres prévient la fuite spectrale et améliore l'analyse de signaux acquis.
- Le chapitre 15, *Analyse et mesure de spectre*, indique comment déterminer l'amplitude et le spectre de phase, développer un analyseur de spectre et calculer la distorsion harmonique totale (THD).
- Le chapitre 16, *Filtrage*, explique comment filtrer des fréquences non désirées depuis des signaux utilisant des filtres à réponse impulsionnelle infinie (RII), des filtres à réponse impulsionnelle finie (RIF) et des filtres non linéaires.
- Le chapitre 17, *Ajustement de courbe*, présente comment extraire des informations d'un ensemble de données pour obtenir une description de tendance de données.
- Le chapitre 18, *Algèbre linéaire*, explique comment réaliser l'analyse et le calcul de matrice.
- Le chapitre 19, *Probabilités et statistiques*, explique des concepts fondamentaux de probabilité et de statistiques, et montre comment utiliser ces concepts pour la résolution de problèmes du monde réel.

Partie IV, Communication inter-applications et sur réseau

Cette section contient des informations de base sur la communication inter-applications et sur réseau.

La Partie IV, *Communication inter-applications et sur réseau*, contient les chapitres suivants.

- Le chapitre 20, *Introduction à la communication*, présente la façon dont LabVIEW gère la communication inter-applications et sur réseau.
- Le chapitre 21, *TCP et UDP*, explique les concepts de base du protocole de gestion de transmission (TCP), du protocole Internet (IP) et des adresses Internet.
- Le chapitre 22, *Support d'ActiveX*, indique comment LabVIEW peut être un serveur et un client ActiveX. ActiveX est identique à la communication Automation OLE.
- Le chapitre 23, *Utilisation du protocole DDE*, explique comment utiliser l'échange dynamique de données (DDE) pour communiquer entre des applications Windows. On peut utiliser le protocole DDE dans une application de type client, serveur ou en réseau.
- Le chapitre 24, *AppleEvents*, indique comment AppleEvents est utilisé pour communiquer entre LabVIEW et d'autres applications Macintosh. LabVIEW peut être un serveur et un client AppleEvents.
- Le chapitre 25, *Communication programme à programme*, explique comment LabVIEW peut communiquer avec d'autres applications Macintosh en utilisant la communication programme à programme (PPC).

Partie V, Programmation en G avancée

Cette section contient des informations sur la personnalisation de VIs, le contrôle d'objets de la face-avant par programme et des conseils pour la conception d'applications complexes.

La Partie V, *Programmation en G avancée*, contient les chapitres suivants.

- Le chapitre 26, *Personnalisation de VIs*, vous explique comment utiliser les options **Configuration du VI…** et **Configuration du nœud de VI…** pour personnaliser l'apparence et le comportement d'un VI lorsqu'il s'exécute.
- Le chapitre 27, *Attributs d'objets de la face-avant*, décrit les objets appelés Attribute Nodes, qui sont des **nœuds** de diagramme spéciaux contrôlant l'apparence et les caractéristiques fonctionnelles des commandes et des indicateurs.

- Le chapitre 28, *Conception des programmes*, explique des techniques à utiliser lors de la création de programmes et donne des conseils concernant le style de programmation.
- Le chapitre 29, *Autres ressources disponibles*, fournit des informations sur les ressources que vous pouvez utiliser pour créer vos applications avec succès.

Annexes, glossaire et index

- L'Annexe A, *Références d'analyse*, liste les matériaux de référence utilisés pour produire les VIs d'analyse dans LabVIEW. Ces références contiennent plus d'informations sur les théories et les algorithmes implémentés dans la bibliothèque d'analyse.
- L'Annexe B, *Questions fréquemment posées*, répond aux questions courantes sur les communications sur réseau avec LabVIEW et sur les E/S d'instruments, GPIB et série.
- L'Annexe C, *Informations à l'attention du client*, contient des formulaires qui vous aideront à regrouper les informations nécessaires à la résolution de problèmes techniques. Vous trouverez également dans cette annexe un formulaire que vous pouvez utiliser pour apporter vos commentaires sur la documentation du produit.
- Le *Glossaire* contient une liste alphabétique des termes utilisés dans ce manuel, notamment les abréviations, les acronymes, les préfixes métriques, les mnémoniques et les symboles.
- L'*Index* contient une liste alphabétique des termes et des sujets clés de ce manuel, y compris le numéro de la page où vous pouvez les retrouver.

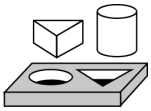
Conventions utilisées dans ce manuel

Les conventions suivantes sont utilisées dans ce manuel :

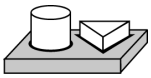
< >
: Les crochets à chevrons contiennent le nom d'une touche du clavier — par exemple, <Maj>. Les crochets à chevrons contenant des nombres séparés par des points de suspension représentent un ensemble de valeurs associé à un bit ou un nom de signal — par exemple, DBIO<3.0>.

-
: Un trait d'union entre deux ou plusieurs noms de touches, dans des crochets à chevrons signifie que vous devez appuyer en même temps sur les touches désignées — par exemple, <Control-Alt-Supprimer>.

» Le symbole **»** vous conduit à travers des éléments du menu et des options de boîtes de dialogue imbriqués vers une action finale. La séquence **Fichier»Mise en page»Options»Fontes de substitution** vous invite à dérouler le menu **Fichier**, à sélectionner l'élément **Mise en page** puis **Options,** et enfin à sélectionner les options **Fontes de substitution** dans la dernière boîte de dialogue.



Cette icône associée à un texte en caractères gras signifie le commencement d'un exercice qui contient des instructions que vous pouvez suivre pas à pas pour en apprendre davantage sur LabVIEW.



Cette icône associée à un texte en caractères gras signifie la fin d'un exercice qui contient des instructions que vous pouvez suivre pas à pas pour en apprendre davantage sur LabVIEW.

☞ Cette icône associée à un texte en caractères gras et en italique signale une remarque, vous apportant des informations importantes.

⚠ Cette icône associée à un texte en caractères gras et en italique signale un avertissement vous indiquant des précautions à prendre pour éviter une blessure, la perte de données ou une panne du système.

gras Un texte en caractères gras signale des noms de menus, d'éléments de menu, des paramètres, des boîtes de dialogue, des boutons ou des options de boîtes de dialogue, des icônes, des fenêtres, des onglets Windows 95, ou des DEL.

italique gras Un texte en caractères gras et en italique signale l'objectif d'un exercice, une remarque, un avertissement ou une mise en garde.

`gras Courier` Un texte en caractères gras *et en police Courier signale des messages et des réponses imprimés automatiquement à l'écran par l'ordinateur.*

italique Un texte en italique signale des variables, une emphase, une référence croisée ou l'introduction à un concept clé. Cette fonte indique également un texte dans lequel vous fournissez le mot ou la valeur approprié, comme dans Windows 3.*x*.

`Courier` Un texte en police de caractères Courier signale un texte ou des caractères que vous devez entrer littéralement en utilisant le clavier, des sections de code, des exemples de programmation et des exemples de syntaxe. Cette fonte est également utilisée pour les noms de lecteurs, les chemins, les répertoires, les programmes, les sous-programmes, les noms de périphériques, les fonctions, les opérations, les variables, les noms de

fichier et les extensions, et enfin pour les déclarations et commentaires extraits de programmes.

chemins — Les chemins de ce manuel sont signalés par des barres obliques inverses (\) qui séparent des noms de lecteurs, de répertoires, de dossiers et de fichiers.

plate-forme — Un texte dans cette fonte signale des informations relatives à une plate-forme spécifique.

Références bibliographiques

- *Manuel de référence de programmation en G*
- *Manuel de base d'acquisition de données LabVIEW*
- *Manuel de référence des fonctions et des VIs de LabVIEW*
- *Guide de prise en main de LabVIEW*
- *Référence en ligne de LabVIEW*, disponible en sélectionnant **Aide»Référence en ligne**
- *Tutorial en ligne de LabVIEW* **(Windows uniquement),** que vous lancez dans la boîte de dialogue LabVIEW
- *Carte de référence rapide de programmation en G*
- *Carte de démarrage de LabVIEW*
- *Notes d'information de LabVIEW*
- *Notes de mise à jour de LabVIEW*

Communication avec l'utilisateur

National Instruments aimerait recevoir vos commentaires sur ses produits et manuels. Les applications que vous développez avec nos produits nous intéressent et nous aimerions vous aider en cas de problèmes avec ces derniers. Pour que vous puissiez nous contacter facilement, ce manuel contient des formulaires de commentaires et de configuration que vous êtes invité à remplir. Vous trouverez ces formulaires à l'Annexe C, *Informations à l'attention du client*, à la fin de ce manuel.

1

Introduction

Ce chapitre présente l'approche unique avec laquelle LabVIEW aborde la programmation. Vous y trouverez également une explication des notions de base nécessaires au développement de programmes à l'aide de LabVIEW. Ce chapitre fait également référence à d'autres chapitres ou manuels pour des informations complémentaires.

Qu'est-ce que LabVIEW ?

LabVIEW est un environnement de développement de programme, tout comme les environnements de développement BASIC ou C modernes, ainsi que LabWindows/CVI de National Instruments. LabVIEW diffère toutefois de ces applications sur un point important. En effet, alors que les autres systèmes de programmation emploient des langages *textuels* pour créer des lignes de code, LabVIEW utilise un langage de programmation *graphique*, le *G*, pour créer des programmes sous la forme de diagrammes.

LabVIEW, comme le langage C ou BASIC, est un système de programmation à usage général, avec des bibliothèques de fonctions étendues convenant à toute tâche de programmation. LabVIEW comprend des bibliothèques pour l'acquisition de données, le contrôle d'instruments série et GPIB, ainsi que pour l'analyse, la présentation et le stockage de données. LabVIEW comprend également des outils d'élaboration de programme traditionnels, de manière à pouvoir définir des points d'arrêt, animer l'exécution pour visualiser le transfert les données dans le programme, et exécuter pas à pas le programme pour faciliter la mise au point et le développement de programme.

Comment fonctionne LabVIEW ?

Bien que LabVIEW soit un système de programmation à usage général, il comporte également des bibliothèques de fonctions et des outils de développement conçus spécifiquement pour l'acquisition de données et le contrôle d'instruments. Les programmes LabVIEW sont appelés *VIs* (pour *Virtual Instruments – Instruments virtuels)* en raison de leur apparence et de leur fonctionnement pouvant imiter ceux d'instruments réels. Ils sont

cependant identiques aux fonctions des langages de programmation conventionnels.

Un VI comprend une interface utilisateur interactive (un diagramme de flux de données équivalent au code source) et des connexions entre icônes permettant au VI d'être appelé par des VIs d'un niveau supérieur. Plus précisément, les VIs sont structurés comme suit :

- L'interface utilisateur interactive d'un VI est appelée *face-avant*, en raison de sa ressemblance avec la face-avant d'un instrument réel. La face-avant peut contenir des boutons rotatifs, des boutons-poussoirs, des graphiques et d'autres commandes et indicateurs. Vous pouvez entrer vos données à l'aide d'une souris et d'un clavier, et voir les résultats sur l'écran de l'ordinateur.
- Le VI reçoit des instructions d'un *diagramme*, que vous construisez en G. Le diagramme est une solution graphique à un problème de programmation. Le diagramme est également le code source du VI.
- Les VIs suivent un format modulaire hiérarchique. Vous pouvez les utiliser comme programmes principaux ou comme sous-programmes au sein d'autres programmes. Un VI utilisé à l'intérieur d'un autre VI est appelé un *sous-VI*. L'*icône* et le *connecteur* d'un VI fonctionnent comme une liste graphique de paramètres permettant aux autres VIs de transmettre des données à un sous-VI.

C'est grâce à ces fonctions que LabVIEW tire le meilleur parti du concept de *programmation modulaire*. Vous divisez une application en une série de tâches, que vous divisez à nouveau jusqu'à ce qu'une application compliquée devienne une série de simples sous-tâches. Vous construisez un VI pour accomplir chaque sous-tâche et combinez ces VIs dans un autre diagramme pour accomplir une tâche plus importante. A la fin, votre VI principal contient une série de sous-VIs représentant les fonctions d'une application.

Chaque sous-VI pouvant être exécuté de façon autonome, indépendamment du reste de l'application, la mise au point en est d'autant simplifiée. En outre, comme plusieurs sous-VIs secondaires réalisent souvent des tâches communes à différentes applications, vous pouvez ainsi développer un ensemble spécialisé de sous-VIs correspondant exactement aux applications que vous serez amené à développer.

Programmation en G

Le G est le langage de programmation convivial, à flux de données graphiques, sur lequel est fondé LabVIEW. Le G simplifie les calculs scientifiques, le contrôle/commande de processus, ainsi que les applications d'essais et de mesures. Vous pouvez également utiliser ce langage pour de nombreuses autres applications.

La Partie I, *Introduction à la programmation en G*, traite des fonctionnalités du langage G nécessaires au lancement de la plupart des applications LabVIEW. Pour une explication plus complète des fonctionnalités de LabVIEW, reportez-vous au *Manuel de référence de programmation en G*.

Les concepts de base du G traités dans ce manuel sont décrits dans la liste suivante.

- VIs : les instruments virtuels (VIs) possèdent trois éléments principaux : la face-avant, le diagramme et l'icône/connecteur. La face-avant représente l'interface utilisateur du VI. Quant au diagramme, il s'agit du code exécutable que vous créez à l'aide des nœuds, des terminaux et des fils de liaison. L'icône/connecteur permet de personnaliser un VI et de l'utiliser en tant que sous-VI dans le diagramme d'un autre VI. Pour plus d'informations sur les VIs, reportez-vous au chapitre 2, *Création de VIs*, et au chapitre 26, *Personnalisation de VIs*.
- Boucles et graphes : le G possède deux structures permettant de répéter l'exécution d'un sous-diagramme : la *boucle While* et la *boucle For*. Les deux structures apparaissent sous forme de boîtes dont on peut changer les dimensions. Vous placez la portion de diagramme à répéter à l'intérieur du cadre représentant la boucle. La boucle While s'exécute aussi longtemps que la valeur du terminal conditionnel reste TRUE (vrai). Quant à la boucle For, elle s'exécute un nombre défini de fois. Les graphes sont utilisés pour afficher des informations sur les tendances en temps réel pour l'opérateur. Pour plus d'informations sur les boucles et les graphes, reportez-vous au chapitre 3, *Boucles et graphes déroulants*.
- Structures Condition et Séquence : la *structure Condition* est une structure de contrôle de branchement conditionnel qui exécute un sous-diagramme basé sur une information d'entrée définie. Une *structure Séquence* est une structure de contrôle de programme qui exécute ses sous-diagrammes selon un ordre numérique. Pour plus d'informations sur les structures Condition et Séquence, reportez-vous au chapitre 4, *Structure Condition, structure Séquence et boîte de calcul*.

- Attribute Nodes : les *attribute nodes* sont des nœuds de diagramme particuliers que vous pouvez utiliser pour contrôler l'apparence et les fonctionnalités des commandes et des indicateurs. Pour plus d'informations sur les Attributes Nodes, reportez-vous au chapitre 27, *Attributs d'objets de la face-avant*.
- Tableaux, clusters et graphes : un *tableau* est un regroupement d'éléments de données du même type, dont vous pouvez modifier la taille. Quant au *cluster*, c'est un regroupement à taille fixée d'éléments de données du même type ou d'un type différent. Les graphes sont généralement utilisés pour afficher des données. Pour plus d'informations sur les tableaux, les clusters et les graphes, reportez-vous au chapitre 5, *Tableaux, clusters et graphes*.

Organisation du système LabVIEW (Windows)

Une fois l'installation terminée conformément aux *Notes d'informations LabVIEW* livrées avec votre logiciel, votre répertoire LabVIEW doit contenir les fichiers suivants.

- `LABVIEW.EXE` : représente le programme LabVIEW. Lancez ce programme pour démarrer LabVIEW.
- Répertoire `vi.lib` : contient les bibliothèques de VIs de LabVIEW, y compris les VIs GPIB, d'analyse et d'acquisition de données (DAQ). La plupart de ces VIs sont disponibles dans la palette **Fonctions**.
- Répertoire `examples` : contient de nombreux sous-répertoires d'exemples. Ce répertoire contient également un VI appelé `readme.vi` servant de guide pour les exemples.
- `serpdrv` et `daqdrv` : ces fichiers font partie de l'interface de communication de LabVIEW avec, respectivement, le port série et le DAQ. Ces fichiers doivent se trouver dans le même répertoire que `vi.lib`.
- Répertoire `resource`
  - `labview.rsc`, `lvstring.rsc` et `lvicon.rsc` : ce sont les fichiers de données utilisés par l'application LabVIEW.
  - **(Windows 3.1)** `lvdevice.dll` : ce fichier fournit des services de séquencement pour LabVIEW et doit se trouver dans le même répertoire que `vi.lib` pour que LabVIEW puisse s'exécuter.
  - **(Windows 3.1)** `lvimage.dll` : ce fichier permet à LabVIEW de charger des images créées à l'aide de différents programmes graphiques.

  - `labview50.tlb` : ce fichier est une bibliothèque de types pour permettre à LabVIEW de se comporter comme un serveur ActiveX.
  - `ole_container.dll` : ce fichier permet à LabVIEW d'afficher et de mettre à jour les containers ActiveX.
  - `lvwutil32.dll` : ce fichier est utilisé par l'Assistant Solutions, qui construit des exemples d'E/S d'instrument et DAQ selon vos besoins.
  - `lvjpeg.dll` et `lvpng.dll` : ces fichiers fournissent un support pour l'affichage des graphiques PNG et JPEG dans des fichiers HTML, lorsque vous imprimez la documentation d'un VI dans un fichier HTML.
- Répertoire `Cintools` : contient les fichiers nécessaires à la création des CIN (Code Interface Node – nœud d'interface de code), qui permettent de relier un code C aux VIs de LabVIEW.
- Fichier `visarc` : fait partie de l'interface de LabVIEW avec VISA (Virtual Instrument Software Architecture – architecture logicielle d'instruments virtuels). VISA fournit une bibliothèque à interface unique pour contrôler des instruments série, GPIB et VXI.
- `labview.ini` : contient les options de configuration de LabVIEW.
- Répertoire `Project` : contient les fichiers qui deviendront des éléments du menu **Projet** de LabVIEW.
- Répertoire `menus` : contient les fichiers utilisés pour configurer la structure des palettes **Commandes** et **Fonctions**.
- Répertoire `Instr.lib` : contient les drivers d'instruments utilisés pour contrôler les instruments série, GPIB et VXI. Lorsque vous installez des drivers d'instruments de National Instruments, placez-les dans ce répertoire pour qu'ils puissent être ajoutés à la palette **Fonctions**.
- Répertoire `Help` : contient une documentation en ligne complète, ainsi que le fichier d'aide "Chercher un exemple" qui vous aide à trouver les exemples communs à votre application.
- Répertoire `Tutorial` : contient les fichiers nécessaires à l'exécution du tutorial en ligne, un didacticiel interactif traitant des concepts de base de l'environnement LabVIEW.
- Répertoire `Activity` : c'est ici que vous pouvez enregistrer les VIs que vous avez créés, tout en effectuant les activités décrites dans ce manuel.

- Répertoire `User.lib` : c'est ici que vous pouvez enregistrer les VIs que vous avez créés et qui sont souvent utilisés. Les VIs de ce répertoire sont affichés dans la palette **Fonctions**.
- Répertoire `Wizard` : ce répertoire crée l'option **Assistant Solutions** dans le menu **Fichier**. Vous pouvez utiliser ce répertoire pour ajouter des éléments au menu **Fichier**.

LabVIEW installe les drivers pour le matériel GPIB, d'acquisition des données et VXI. Pour plus d'informations concernant la configuration, consultez le chapitre 2, *Installation et configuration du matériel d'acquisition de données*, dans le *Manuel de base d'acquisition de données LabVIEW*, le *Manuel de référence de VI VXI* et le chapitre 8, *Tutorial VISA de LabVIEW*, dans ce manuel.

Ecran de démarrage sous Windows

Lorsque vous lancez LabVIEW, une boîte de dialogue de navigation de bienvenue s'affiche, depuis laquelle vous pouvez accéder facilement aux notes de présentation, aux commandes courantes et aux "Trucs et astuces". Si vous ne voulez pas que la boîte de dialogue de navigation s'affiche, vous pouvez la désactiver grâce à la case à cocher située au bas de la boîte de dialogue. Pour la réactiver, utilisez la boîte de dialogue Préférences.

Lorsque tous les VIs sont fermés, une boîte de dialogue similaire apparaît. Le bouton **Petite boîte de dialogue** permet d'obtenir une version simplifiée de cette boîte de dialogue : avec uniquement les boutons **Nouveau**, **Ouvrir** et **Quitter**.

Organisation du système LabVIEW (Macintosh)

Une fois l'installation terminée conformément aux *Notes d'informations LabVIEW* livrées avec votre logiciel, votre dossier LabVIEW doit contenir les fichiers suivants.

- `LabVIEW` : représente le programme LabVIEW. Lancez ce programme pour démarrer LabVIEW.
- Dossier `vi.lib` : contient les bibliothèques de VIs de LabVIEW, y compris les VIs GPIB, d'analyse et d'acquisition de données (DAQ). La plupart de ces VIs sont disponibles dans la palette **Fonctions**.
- Dossier `examples` : contient de nombreux sous-dossiers d'exemples. Ce dossier contient également un VI appelé `readme.vi` servant de guide pour les exemples.

- Dossier `resource`
  - `lvstring.rsrc` et `lvicon.rsrc` : fichiers de données utilisés par l'application LabVIEW.
  - `lvjpeg.lib` et `lvpng.lib` : ces fichiers fournissent un support pour l'affichage des graphiques PNG et JPEG dans des fichiers HTML lorsque vous imprimez la documentation d'un VI dans un fichier HTML.
- Dossier `cintools` : contient les fichiers nécessaires à la création des CIN (Code Interface Node – nœud d'interface de code), qui permettent de relier un code C aux VIs de LabVIEW.
- Fichier `visarc` : fait partie de l'interface de LabVIEW avec VISA (Virtual Instrument Software Architecture – architecture logicielle d'instruments virtuels). VISA fournit une bibliothèque à interface unique pour contrôler des instruments série, GPIB et VXI.
- Dossier `Project` : contient les fichiers qui deviendront des éléments du menu **Projet** de LabVIEW.
- Dossier `menus` : contient les fichiers utilisés pour configurer la structure des palettes **Commandes** et **Fonctions**.
- Dossier `instr.lib` : contient les drivers d'instruments utilisés pour contrôler les instruments série, GPIB et VXI. Lorsque vous installez des drivers d'instruments de National Instruments, placez-les dans ce répertoire pour qu'ils puissent être ajoutés à la palette **Fonctions**.
- Dossier `help` : contient une documentation en ligne complète, ainsi que le fichier d'aide "Chercher un exemple" qui vous aide à trouver les exemples communs à votre application.
- Dossier `activity` : c'est ici que vous pouvez enregistrer les VIs que vous avez créés, tout en effectuant les activités décrites dans ce manuel.
- Dossier `user.lib` : c'est ici que vous pouvez enregistrer les VIs que vous avez créés et qui sont souvent utilisés. Les VIs de ce répertoire sont affichés dans la palette **Fonctions**.
- Dossier `Wizard` : ce dossier crée l'option **Assistant Solutions** dans le menu **Fichier** (**Macintosh PCI uniquement**). Vous pouvez utiliser ce dossier pour ajouter des éléments au menu **Fichier**.

En outre, l'utilitaire d'installation de LabVIEW permet d'installer plusieurs fichiers de drivers afin de pouvoir utiliser des cartes enfichables DAQ et/ou GPIB.

- `Dossier Système:Tableaux de bord:NI-488 INIT` : ce tableau de bord contient les drivers des cartes GPIB. Vous pouvez l'utiliser pour configurer vos cartes, mais vous n'avez que rarement besoin de changer des paramètres.
- `Dossier Système:Tableaux de bord:NI-DAQ` : ce tableau de bord charge les drivers DAQ en mémoire. Vous pouvez l'utiliser pour configurer l'emplacement et le comportement de vos cartes DAQ et de vos modules SCXI.
- `Dossier Système:Extensions:NI-DMA/DSP` : les drivers GPIB et DAQ utilisent cette extension. Celle-ci fournit un support pour le transfert DMA (accès direct à la mémoire) des données, permettant ainsi un taux de transfert des données plus rapide. Cette extension offre également un support pour les cartes NI-DSP.

LabVIEW installe les drivers pour le matériel GPIB et d'acquisition de données. Pour des informations concernant la configuration, consultez le chapitre 2, *Installation et configuration du matériel d'acquisition de données*, dans le *Manuel de base d'acquisition de données LabVIEW.*

Organisation du système LabVIEW (UNIX)

Une fois l'installation terminée conformément aux *Notes d'informations LabVIEW* livrées avec votre logiciel, votre répertoire LabVIEW doit contenir les fichiers suivants.

- `labview` : représente le programme LabVIEW. Lancez ce programme pour démarrer LabVIEW.
- Répertoire `vi.lib` : contient les bibliothèques de VIs de LabVIEW, y compris les VIs GPIB, d'analyse et d'acquisition de données (DAQ). La plupart de ces VIs sont disponibles dans la palette **Fonctions**.
- Répertoire `examples` : contient de nombreux sous-répertoires d'exemples. Ce répertoire contient également un VI appelé `readme.vi` servant de guide pour les exemples.
- `serpdrv` : ce fichier fait partie de l'interface de LabVIEW avec la communication du port série. Ce fichier doit se trouver dans le même répertoire que `vi.lib`.

- Répertoire `resource`
  - `labview.rsc`, `lvstring.rsc` et `lvicon.rsc` : fichiers de données utilisés par l'application LabVIEW.
  - `lvjpeg.lib` et `lvpng.lib` : ces fichiers fournissent un support pour l'affichage des graphiques PNG et JPEG dans des fichiers HTML lorsque vous imprimez la documentation d'un VI dans un fichier HTML.
- Répertoire `cintools` : contient les fichiers nécessaires à la création des CIN (Code Interface Node – nœud d'interface de code), qui permettent de relier un code C aux VIs de LabVIEW.
- Fichier `visarc` : fait partie de l'interface de LabVIEW avec VISA (Virtual Instrument Software Architecture – architecture logicielle d'instruments virtuels). VISA fournit une bibliothèque à interface unique pour contrôler des instruments série, GPIB et VXI.
- Répertoire `Project` : contient les fichiers qui deviendront des éléments du menu **Projet** de LabVIEW.
- Répertoire `menus` : contient les fichiers utilisés pour configurer la structure des palettes **Commandes** et **Fonctions**.
- Répertoire `instr.lib` : contient les drivers d'instruments utilisés pour contrôler les instruments série, GPIB et VXI. Lorsque vous installez des drivers d'instruments de National Instruments, placez-les dans ce répertoire pour qu'ils puissent être ajoutés à la palette **Fonctions**.
- Répertoire `help` : contient une documentation en ligne complète, ainsi que le fichier d'aide "Chercher un exemple" qui vous aide à trouver les exemples communs à votre application.
- Répertoire `activity` : c'est ici que vous pouvez enregistrer les VIs que vous avez créés, tout en effectuant les activités décrites dans ce manuel.
- Répertoire `user.lib` : c'est ici que vous pouvez enregistrer les VIs que vous avez créés et qui sont souvent utilisés. Les VIs de ce répertoire sont affichés dans la palette **Fonctions**.
- Répertoire `Wizard` : ce répertoire crée l'option **Assistant Solutions** dans le menu **Fichier**. Vous pouvez utiliser ce répertoire pour ajouter des éléments au menu **Fichier**.
- Répertoire `acrobat` : contient une documentation en ligne au format Acrobat (`.pdf`).
- Répertoire `acroread` : contient des fichiers de l'utilitaire Acrobat d'Adobe.

Support des toolkits

Les fichiers installés dans `vi.lib\addons` s'affichent automatiquement au niveau supérieur des palettes **Commandes** et **Fonctions**. De nouveaux toolkits peuvent utiliser cette caractéristique pour rendre les fichiers plus accessibles après leur installation. Si vos fichiers ont déjà été installés par les toolkits dans un autre endroit, vous pouvez les déplacer dans le répertoire `addons` pour permettre un accès plus simple. Si vous souhaitez ajouter vos propres VIs dans les palettes, nous vous recommandons de les placer dans `user.lib` ou de les ajouter dans un type de palette personnalisé.

Par où commencer ?

Ce manuel fournit des informations de base sur la façon de construire une application dans LabVIEW. Pour vous familiariser avec l'environnement de LabVIEW, consultez le *Tutorial en ligne LabVIEW* **(Windows uniquement)**, le *Guide de prise en main LabVIEW* et la Partie I, *Introduction à la programmation en G*, figurant dans ce manuel.

La plupart des applications LabVIEW sont divisées en plusieurs tâches : interface d'E/S pour les capteurs ou les instruments, affichage des données sur la face-avant, analyse des données, stockage des données et transfert des données sur un réseau. Pour en apprendre davantage sur chacune de ces tâches, reportez-vous à la Partie II, *Interfaces d'E/S*, à la Partie III, *Analyse* et à la Partie IV, *Communication inter-applications et sur réseau*. Pour les techniques de programmation en G avancées, reportez-vous à la Partie V, *Programmation en G avancée*, dans ce manuel.

Pour générer ou trouver des exemples similaires à votre application, reportez-vous à l'Assistant Solutions **(Windows et Macintosh PCI uniquement)** ou au fichier d'aide en ligne "Chercher un exemple" **(Windows uniquement)**, auquel vous pouvez accéder depuis la boîte de dialogue de démarrage de LabVIEW.

Pour plus d'informations sur les fonctions et sur les VIs, reportez-vous au *Manuel de référence des VIs et des fonctions de LabVIEW* ainsi qu'à l'aide en ligne.

Partie I

Introduction à la programmation en G

Cette section contient des informations de base sur la création d'instruments virtuels (VIs), l'utilisation de VIs dans d'autres VIs, les structures de programmation telles que des boucles, et les structures de données telles que des tableaux et des chaînes de caractères.

La Partie 1, *Introduction à la programmation en G*, contient les chapitres suivants.

- Le chapitre 2, *Création de VIs*, explique comment créer un VI comprenant la face-avant, qui est l'interface utilisateur, et le diagramme, qui est le code source. Une fois que vous avez créé un VI, vous pouvez l'utiliser dans d'autres VIs.
- Le chapitre 3, *Boucles et graphes déroulants*, vous montre comment répéter les portions du diagramme en utilisant une boucle While et une boucle For. Ce chapitre explique également comment afficher plusieurs points graphiquement, un à la fois, sur un graphe.
- Le chapitre 4, *Structure Condition, structure Séquence et boîte de calcul*, explique comment utiliser la structure Condition, qui est une structure conditionnelle, la structure Séquence, qui aide à établir l'ordre d'exécution, et la Boîte de calcul, qui aide à l'exécution de formules mathématiques.
- Le chapitre 5, *Tableaux, clusters et graphes*, vous montre comment afficher un groupe ou tableau de points sur un graphe. Vous pouvez passer les paramètres d'échelle aussi bien qu'un tableau de points sur un graphe en créant un cluster, qui est un groupe de différents types de données.
- Le chapitre 6, *Chaînes de caractères et E/S sur fichiers*, explique comment manipuler des chaînes de caractères et les écrire dans un fichier ASCII.

☞ **Remarque**

(Windows 3.1) ***Vous devez enregistrer les VIs que vous créez dans la Partie 1 dans les bibliothèques de VIs. Les bibliothèques de VIs vous permettent d'utiliser des noms de fichiers qui ont plus de huit caractères. En outre, les VIs nécessaires aux exercices de la Partie I se trouvent dans la bibliothèque de VI*** `LabVIEW\Activity\Activity.llb`***. Reportez-vous à la section*** *Enregistrement des VIs* ***au chapitre 2****, Edition de VIs****, du*** *Manuel de référence de programmation en G* ***pour plus d'informations sur les bibliothèques de VIs.***

2

Création de VIs

Ce chapitre introduit les concepts de base des instruments virtuels. Vous y trouverez également deux exercices destinés à vous apprendre comment effectuer les tâches suivantes :

- Création de l'icône et du connecteur
- Utilisation d'un VI comme sous-VI

Qu'est-ce qu'un instrument virtuel ?

Un instrument virtuel (VI) est un programme conçu en langage de programmation graphique G. La face-avant d'un instrument virtuel possède souvent une interface utilisateur similaire aux instruments physiques. Le langage G est également doté de fonctions intégrées similaires aux VIs, mais ne possédant pas de face-avant ou de diagramme comme les VIs. Les icônes de fonctions ont toujours un arrière-plan de couleur jaune.

Comment construire un VI ?

Pour passer maître dans la création des applications LabVIEW, vous devez comprendre et tirer parti de la nature hiérarchique du VI. Ainsi, après avoir créé un VI, vous pouvez l'utiliser comme sous-VI dans le diagramme d'un VI principal.

Hiérarchie des VIs

Lorsque vous créez une application, vous démarrez par le VI principal et définissez les entrées et les sorties de l'application. Vous construisez ensuite des sous-VIs permettant d'effectuer les opérations nécessaires sur les données lorsque celles-ci passent à travers le diagramme. Si un diagramme possède de nombreuses icônes, regroupez-les dans un VI de niveau inférieur afin que le diagramme reste simple. Cette approche modulaire facilite la mise au point, la compréhension et la maintenance des applications.

Comme avec d'autres applications, vous pouvez enregistrer votre VI comme un fichier dans un répertoire standard. En G, vous pouvez également enregistrer plusieurs VIs dans un fichier unique appelé "bibliothèque de VIs".

Si vous utilisez Windows 3.1, vous devez enregistrer vos VIs dans des bibliothèques de VIs afin de pouvoir utiliser des noms de fichiers longs (jusqu'à 255 caractères), ainsi que des majuscules ou des minuscules.

Il n'est pas nécessaire d'utiliser des bibliothèques de VIs, à moins que vous ne deviez transférer vos VIs sur Windows 3.1. Il est plus utile d'enregistrer des VIs comme des fichiers individuels que d'utiliser des bibliothèques de VIs, car vous pouvez copier, renommer et supprimer les fichiers plus facilement que lorsque vous utilisez une bibliothèque de VIs. Pour parcourir une liste des avantages et des inconvénients des bibliothèques de VIs et des fichiers individuels, consultez la section *Enregistrement des VIs* au chapitre 2, *Edition des VIs*, du *Manuel de référence de programmation en G*.

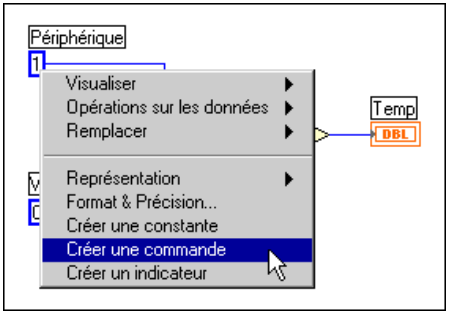
Les bibliothèques de VIs ont les mêmes fonctionnalités que les autres répertoires en ce qui concerne le chargement, l'enregistrement et l'ouverture de fichiers. Elles ne sont toutefois pas hiérarchiques, en d'autres termes, vous ne pouvez pas créer une bibliothèque de VIs à l'intérieur d'une autre bibliothèque de VIs. Vous ne pouvez pas non plus créer un nouveau répertoire à l'intérieur d'une bibliothèque de VIs. Il est impossible de répertorier les VIs d'une bibliothèque de VIs à l'extérieur de l'environnement LabVIEW.

Après la création d'une bibliothèque de VIs, celle-ci apparaît dans la boîte de dialogue de fichier LabVIEW comme un dossier, avec l'indication **VI** libellée sur l'icône du dossier. Les répertoires normaux apparaissent également comme un dossier, mais sans l'appellation VI.

Même si vous ne prévoyez pas d'enregistrer vos propres VIs dans des bibliothèques de VIs, il est bon de vous familiariser avec leur fonctionnement. Dans plusieurs exercices de ce manuel, on vous demande d'enregistrer vos VIs dans le répertoire `LabVIEW\Activity`. Les solutions de ces exercices se trouvent dans le répertoire `LabVIEW\Activity\Solution`.

Commandes, constantes et indicateurs

Une commande est un objet que vous placez sur votre face-avant pour entrer des données dans un VI de façon interactive ou dans un sous-VI par programmation. Un indicateur est un objet que vous placez sur votre face-avant pour afficher une sortie. Les commandes et les indicateurs du G ressemblent, respectivement, aux paramètres d'entrée et de sortie des langages de programmation traditionnels. Pour placer des commandes et des indicateurs sur la face-avant et les câbler aux fonctions ou aux VIs du diagramme, vous pouvez également créer des commandes ou des indicateurs directement depuis le diagramme. Pour cela, *ouvrez le menu local* du terminal d'entrée d'une fonction ou d'un VI sur le diagramme et sélectionnez **Créer une commande**. Ceci permet de créer une commande avec un type de données correct et déjà câblée au terminal.



Vous pouvez créer un indicateur câblé au terminal de sortie en ouvrant le menu local du terminal et en sélectionnant **Créer un indicateur**. Au lieu de placer des constantes sur le diagramme et de les câbler aux fonctions et aux VIs, vous pouvez ouvrir le menu local sur le terminal d'une fonction ou d'un VI et sélectionner **Créer une constante**. Vous ne pouvez pas supprimer une commande ou un indicateur à partir du diagramme. Comme avec tous les objets de la face-avant, vous devez vous placer dans la face-avant, sélectionner l'outil Flèche, puis supprimer l'objet.

Chaque fois que vous créez une nouvelle commande ou un nouvel indicateur sur la face-avant, LabVIEW crée le terminal correspondant dans le diagramme. Les symboles des terminaux indiquent le type de données de la commande ou de l'indicateur. Par exemple, un terminal DBL représente un nombre à virgule flottante double précision ; un terminal TF est un booléen ; un terminal I16 représente un entier 16 bits ; et un terminal ABC représente une chaîne de caractères. Pour plus d'informations sur les types de données du G et leur représentation graphique, consultez la *Carte de référence de programmation en G*.

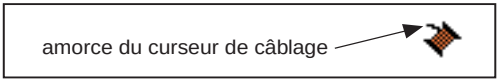
Terminaux

Les terminaux sont des régions (sur un VI ou une fonction) par lesquelles les données sont transmises. Les terminaux sont analogues aux paramètres des langages de programmation textuels. Il est important de câbler correctement les terminaux d'une fonction ou d'un VI. Vous pouvez également afficher le connecteur de l'icône pour faciliter le câblage. Pour ce faire, ouvrez le menu local de la fonction ou du VI et choisissez **Visualiser»Terminaux**. Pour revenir à l'icône, ouvrez le menu local de la fonction ou du VI et sélectionnez de nouveau **Visualiser»Terminaux**.

Fils de liaison

Un *fil de liaison* représente un chemin emprunté par les données entre les nœuds. Les fils de liaison ont des couleurs identifiant le type de données transmises. Ainsi, les fils de liaison bleus transmettent des entiers, les fils de liaison oranges transmettent des nombres à virgule flottante, les fils de liaison verts transmettent des booléens, tandis que les fils de liaison roses transmettent des chaînes de caractères. Pour plus d'informations sur les styles et les couleurs des fils de liaison, consultez la *Carte de référence de programmation en G*.

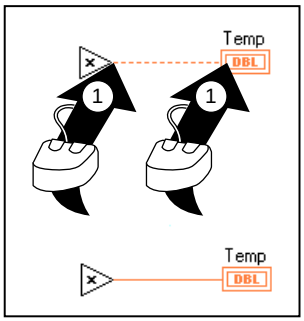
Pour câbler un terminal à un autre, cliquez, avec l'outil Bobine, sur le premier terminal, déplacez l'outil sur le second terminal, puis cliquez sur le second terminal. Vous pouvez commencer par le premier ou le second terminal, l'ordre n'a pas d'importance. L'amorce de l'outil Bobine est l'extrémité de la portion de fil déjà déroulée.



Dans les illustrations de câblage de cette section, la flèche à l'extrémité du symbole de la souris indique l'endroit où il faut cliquer. Quant au nombre imprimé sur la flèche, il indique le nombre de fois qu'il faut cliquer sur le bouton de la souris.

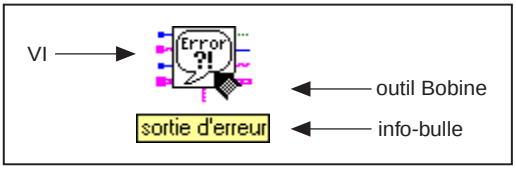
Lorsque l'outil Bobine se trouve sur un terminal, la zone du terminal se met à clignoter, vous indiquant que vous pouvez cliquer dessus pour connecter le fil de liaison à ce terminal. Ne gardez pas le bouton de la souris enfoncé lorsque vous déplacez l'outil Bobine d'un terminal à un autre. Vous pouvez changer la direction d'un fil de liaison en déplaçant la souris de façon perpendiculaire à la direction actuelle. Pour changer plusieurs fois de direction avec le même fil de liaison, cliquez sur le bouton de la souris pour

déposer le fil, puis déplacez la souris perpendiculairement. Vous pouvez également utiliser la barre d'espacement pour modifier la direction d'un fil de liaison.



Info-bulles

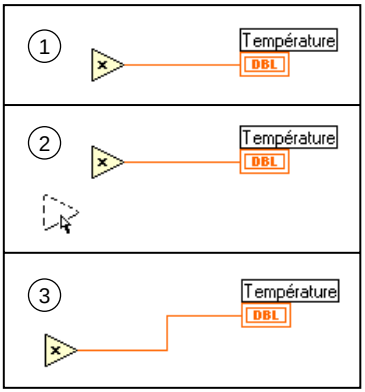
Lorsque vous déplacez l'outil Bobine sur le terminal d'un nœud, une *info-bulle* décrivant ce terminal s'affiche sur l'écran. Les info-bulles sont de petits énoncés de texte jaune qui affichent le nom de chaque terminal. Ces info-bulles sont destinées à vous aider à câbler les terminaux. L'illustration suivante affiche l'info-bulle qui apparaît lorsque vous placez l'outil Bobine sur la sortie du VI "Gestionnaire simple d'erreurs" (Simple Error Handler.vi).



Remarque ***Lorsque vous placez l'outil Bobine sur un nœud, le G affiche les souches du fil de liaison représentant chaque entrée et sortie. La souche du fil de liaison possède un petit point à son extrémité s'il s'agit de l'entrée d'un nœud.***

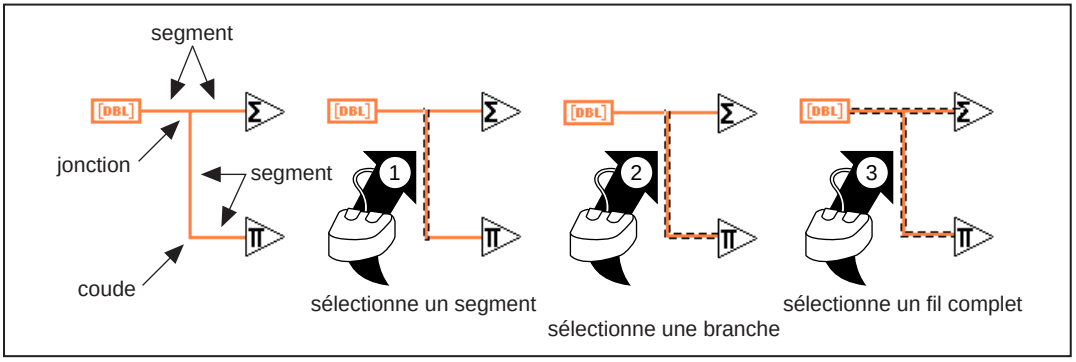
Adaptation des liaisons

Vous pouvez déplacer les objets câblés de façon individuelle ou groupée en faisant glisser les objets sélectionnés sur un nouvel endroit grâce à l'outil Flèche.



Sélection et suppression des fils de liaison

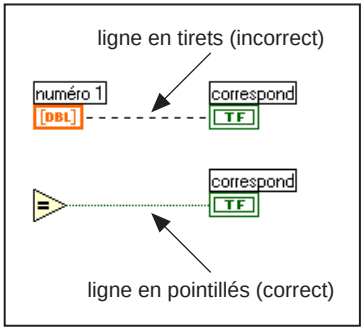
Il se peut que vous câbliez des nœuds de façon incorrecte. Dans ce cas, sélectionnez le fil de liaison que vous souhaitez supprimer et appuyez sur la touche <Supprimer>. Une portion de fil consiste en une simple ligne de liaison horizontale ou verticale. Le point de rencontre de trois ou quatre portions de fil s'appelle une *jonction*. Une branche de liaison contient toutes les portions de fil d'une jonction à une autre, d'un terminal à la jonction suivante, ou d'un terminal à un autre s'il n'y a pas de jonction entre les deux. Vous pouvez sélectionner une portion de fil en cliquant dessus avec l'outil Flèche. Si vous double-cliquez sur le fil, vous sélectionnez une branche et, si vous cliquez trois fois dessus, vous sélectionnez le fil de liaison entier.

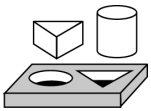


Liaisons incorrectes

Un fil de liaison discontinu représente une mauvaise liaison. Vous pouvez avoir une liaison incorrecte pour de nombreuses raisons, comme la connexion de deux commandes, ou la connexion d'un terminal source à un terminal destination lorsque les types de données ne correspondent pas (par exemple, la connexion d'un numérique à un booléen). Vous pouvez enlever un mauvais fil de liaison en cliquant dessus avec l'outil Flèche et en appuyant sur la touche <Supprimer>. Si vous choisissez **Edition» Supprimer les fils incorrects** ou <Ctrl-B>, vous supprimez tous les mauvais fils de liaison du diagramme. C'est une solution efficace pour laquelle vous pouvez opter si votre VI refuse de s'exécuter ou s'il renvoie le message d'erreur **`Le signal a des segments déconnectés`**.

Remarque ***Ne confondez pas un fil de liaison noir discontinu avec un fil de liaison en pointillé. Ce dernier représente un type de données booléen, comme indiqué dans l'illustration suivante.***





Exercice 2-1. Créer un VI

Votre objectif est de construire un VI.

Supposons que vous possédiez des capteurs délivrant des valeurs de température et de volume mesurées sous la forme de tensions. Vous utiliserez un VI dans le répertoire `LabVIEW\Activity` pour simuler en volts les mesures de la température et du volume. Vous écrirez ensuite un VI pour mettre à l'échelle ces mesures, respectivement en degrés Fahrenheit et en litres.

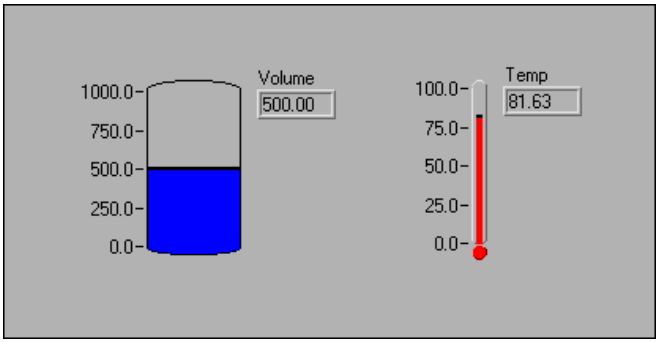
1. Ouvrez une nouvelle face-avant en sélectionnant **Fichier»Nouveau**. Si vous avez fermé tous les VIs, sélectionnez **Nouveau VI** dans la boîte de dialogue LabVIEW.

☞ **Remarque** ***Si la palette* Commandes *n'est pas visible, sélectionnez* Fenêtres»Palette de commandes *pour afficher la palette. Vous pouvez également accéder à la palette* Commandes *en ouvrant le menu local dans une zone libre de la face-avant. Pour ouvrir le menu local, cliquez avec le bouton droit de votre souris (<Option>-cliquez pour un Macintosh).***

2. Sélectionnez un **Réservoir** dans la palette **Commandes»Numérique**, et placez-le sur la face-avant.
3. Tapez `Volume` dans la zone de texte de l'étiquette et cliquez n'importe où sur la face-avant.

☞ **Remarque** ***Si vous cliquez en dehors de la zone de texte sans entrer de texte, l'étiquette disparaît. Pour afficher de nouveau l'étiquette, ouvrez le menu local de la commande et sélectionnez* Visualiser»Etiquette.**

4. Changez l'échelle de l'indicateur du réservoir pour afficher le volume du réservoir entre 0,0 et 1000,0.
   a. En utilisant l'outil Texte, double-cliquez sur la valeur `10,0` de l'échelle du réservoir pour la mettre en surbrillance.
   b. Entrez la valeur `1000` et cliquez avec le bouton de la souris n'importe où sur la face-avant. Les incréments intermédiaires sont automatiquement mis à l'échelle.
5. Placez un thermomètre provenant de la palette **Commandes» Numérique** sur la face-avant. Libellez-le `Temp` et modifiez à nouveau l'échelle pour obtenir une graduation entre 0 et 100.
6. Votre face-avant doit ressembler à l'illustration suivante.



7. Ouvrez le diagramme en choisissant **Fenêtres»Diagramme**. Sélectionnez les objets listés ci-dessous dans la palette **Fonctions** et placez-les sur le diagramme.

☞ **Remarque** ***Si la palette* Fonctions *n'est pas visible, sélectionnez* Fenêtres»Visualiser la palette de fonctions *pour afficher la palette. Vous pouvez également accéder à la palette* Fonctions *en ouvrant le menu local dans une zone libre du diagramme.***

8. Placez chacun des objets suivants sur le diagramme.

Contrôle de processus (Processor Monitor.vi) (**Fonctions»Sélectionner un VI** dans le répertoire `LabVIEW\Activity`) : simule la lecture d'une valeur de température et de volume à partir d'un capteur ou d'un transducteur. Cette valeur est exprimée en volts.

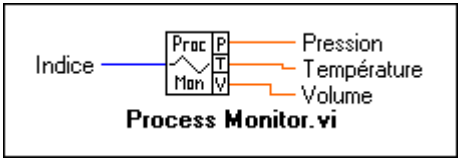
Générateur de nombre aléatoire (**Fonctions»Numérique**) : génère un nombre entre 0 et 1.

Fonction Multiplier (**Fonctions»Numérique**) : multiplie deux nombres et retourne leur produit. Dans cet exercice, vous avez besoin de deux objets. Prenez-en un dans la palette, puis effectuez un copier-coller pour créer le second.

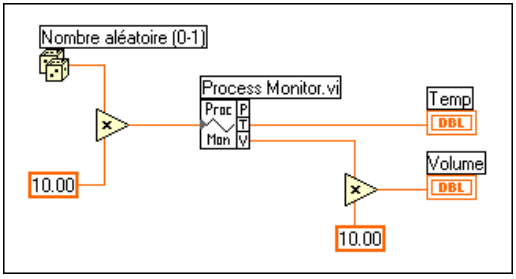
Constante numérique (**Fonctions»Numérique**) : vous avez besoin de deux objets. Prenez-en un dans la palette. En utilisant l'outil Texte, faites passer sa valeur sur `10,00`. Copiez et collez-le pour créer le second objet.

☞ **Remarque** ***Pour créer une constante, vous pouvez également ouvrir le menu local sur le terminal d'une fonction ou d'un VI avec l'outil Bobine. Sélectionnez* Créer une constante *dans le menu local. Une constante du type de données adéquat apparaît.***

9. Pour afficher les entrées et les sorties d'une fonction ou d'un VI, sélectionnez **Visualiser l'aide** dans le menu **Aide**, puis placez le curseur sur une fonction et un VI. La fenêtre d'aide du VI "Contrôle de processus" (Processor Monitor.vi) est affichée ci-dessous.

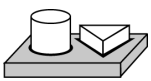


10. En utilisant l'outil Bobine, câblez les objets comme indiqué ci-dessous.



Remarque ***Pour déplacer les objets sur le diagramme, cliquez sur l'outil Flèche dans la palette*** **Outils.**

11. Sélectionnez **Fichier»Enregistrer** et enregistrez le VI sous le nom `Temp & Vol.vi` dans le répertoire `LabVIEW\Activity`.
12. Exécutez le VI à partir de la face-avant en cliquant sur le bouton **Exécuter**. Notez que les valeurs du volume et de la température sont affichées sur la face-avant.
13. Fermez le VI en sélectionnant **Fichier»Fermer.**



Fin de l'exercice 2-1.

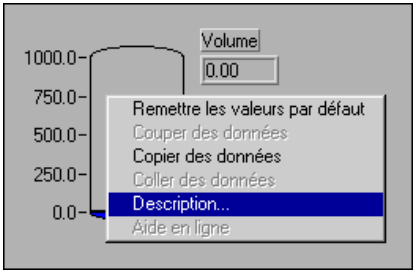
Documentation du VI

Vous pouvez documenter un VI en choisissant l'option **Fenêtres»Infos sur le VI…** Entrez la description du VI dans la boîte de dialogue Informations sur le VI. Vous pouvez ensuite rappeler la description en sélectionnant de nouveau **Fenêtres»Infos sur le VI…**

Vous pouvez éditer les descriptions des objets de la face-avant (ou leurs terminaux respectifs sur le diagramme) en ouvrant le menu local de l'objet et en choisissant **Opérations sur les données»Description…**

☞ **Remarque** ***Vous ne pouvez pas changer la description d'un VI ou des objets de sa face-avant lorsque le VI est en cours d'exécution.***

L'illustration suivante est un exemple de menu local qui apparaît pendant l'exécution d'un VI. Vous ne pouvez pas changer ou étoffer la description pendant l'exécution du VI, mais vous pouvez afficher toutes les informations entrées précédemment.

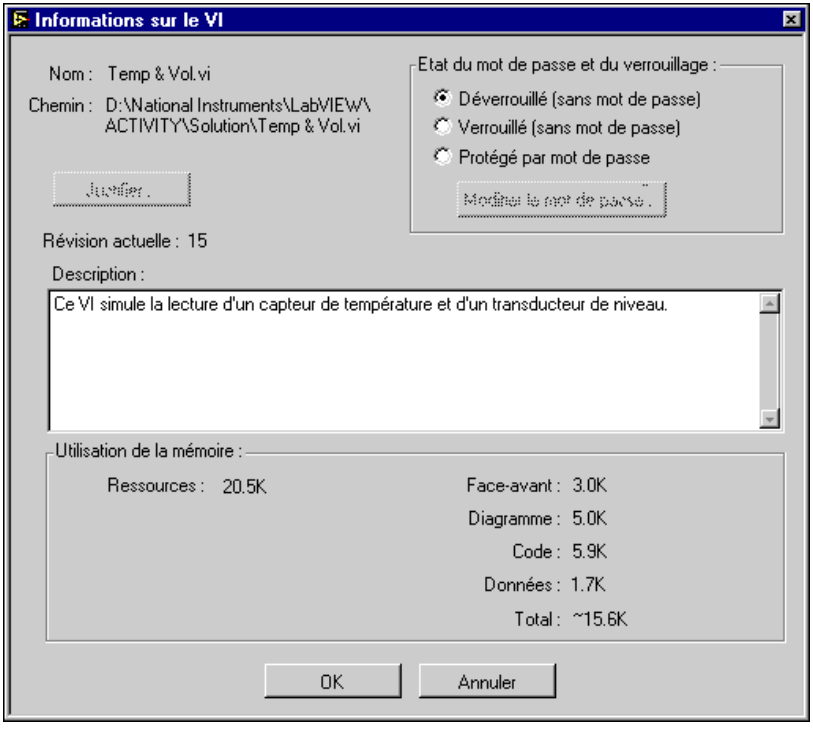


Vous pouvez également afficher la description d'un objet de la face-avant en faisant apparaître la fenêtre d'aide (**Aide»Visualiser l'aide**) et en déplaçant le curseur sur l'objet.

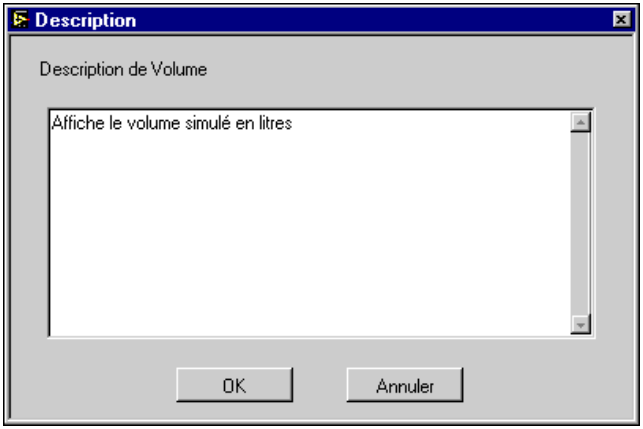
Exercice 2-2. Documenter un VI

Votre objectif est de documenter un VI que vous avez créé.

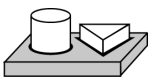
1. Ouvrez le VI `Temp & Vol.vi` créé dans l'exercice 2-1 dans le répertoire `LabVIEW\Activity`.
2. Sélectionnez **Fenêtres»Infos sur le VI…** Entrez la description du VI, comme décrit dans l'illustration suivante, puis cliquez sur **OK.**



3. Ouvrez le menu local du réservoir et choisissez **Opérations sur les données»Description…** Entrez la description de l'indicateur, comme indiqué dans l'illustration suivante, puis cliquez sur **OK**.



4. Ouvrez le menu local du thermomètre et choisissez **Opérations sur les données»Description…** Entrez la description : `Affichage de la mesure de température simulée (deg F)`. Cliquez sur **OK**.
5. Sélectionnez **Visualiser l'aide** dans le menu **Aide**. Placez le curseur sur les indicateurs Volume et Température. Vous pouvez voir dans la fenêtre d'aide les descriptions que vous avez tapées.
6. Enregistrez et fermez le VI.



Fin de l'exercice 2-2.

Qu'est-ce qu'un sous-VI ?

Un *sous-VI* est très semblable à un sous-programme dans les langages de programmation textuels. Il s'agit d'un VI utilisé dans le diagramme d'un autre VI.

Vous pouvez utiliser tout VI possédant une icône et un connecteur comme sous-VI dans un autre VI. Dans le diagramme, sélectionnez des VIs à utiliser comme des sous-VIs grâce à la palette **Fonctions»Sélectionner un VI…** Si vous choisissez cette option, une boîte de dialogue de fichier apparaît, à partir de laquelle vous pouvez sélectionner un VI quelconque dans le système. Si vous ouvrez un VI ne possédant pas d'icône ni de connecteur, une boîte carrée vide apparaît dans le diagramme du VI appelant. Vous ne pouvez pas câbler de fils de liaison à ce nœud. Pour plus d'informations sur les icônes et les connecteurs, consultez le *Tutorial en ligne LabVIEW*, auquel vous pouvez accéder depuis la boîte de dialogue de démarrage.

Un sous-VI est semblable à un sous-programme. Un nœud de sous-VI est analogue à un appel de sous-programme. Le nœud d'un sous-VI n'est pas le sous-VI lui-même, tout comme une déclaration d'appel de sous-programme dans un programme n'est pas le sous-programme lui-même. Un diagramme contenant plusieurs nœuds d'un sous-VI identique appelle plusieurs fois le même sous-VI.

Fenêtre Hiérarchie

La fenêtre Hiérarchie affiche une représentation graphique de la hiérarchie appelante pour tous les VIs en mémoire, y compris les définitions de types et les variables globales. Utilisez la fenêtre Hiérarchie (**Projet»Visualiser la hiérarchie des VIs**) pour afficher les dépendances des VIs en fournissant des informations sur les VIs appelants et sur les sous-VIs. Cette fenêtre contient une barre d'outils que vous pouvez utiliser pour configurer plusieurs types de paramètres des éléments affichés. L'illustration suivante montre un exemple de la barre d'outils de la hiérarchie des VIs.

Vous pouvez utiliser les boutons de la barre d'outils de la fenêtre Hiérarchie ou le menu Afficher, ou bien ouvrir le menu local sur un espace vide dans la fenêtre pour accéder aux options suivantes. Pour plus d'informations sur la fenêtre Hiérarchie, consultez la section *Utilisation de la fenêtre Hiérarchie* au chapitre 3, *Utilisation de sous-VIs*, du *Manuel de référence de programmation en G*.

Refaire le schéma : permet de réorganiser les nœuds après des opérations successives sur des nœuds de hiérarchie, si vous devez minimiser des croisements de lignes et favoriser la symétrie. Si un nœud principal existe, parcourez la fenêtre de sorte que la première racine montrant les sous-VIs soit visible.

Présentation verticale : arrange les nœuds de haut en bas, en plaçant les racines en haut.

Présentation horizontale : arrange les nœuds de gauche à droite, en plaçant les racines sur le côté gauche.

Inclure/Exclure les VIs de VI.lib : fait basculer le graphe de hiérarchie pour inclure les bibliothèques de VIs, ou exclure les VIs des bibliothèques de VIs.

Inclure/Exclure les variables globales : fait basculer le graphe de hiérarchie pour inclure ou exclure des variables globales. Des variables globales enregistrent des données utilisées par plusieurs VIs.

Inclure/Exclure des définitions de types : fait basculer le graphe de hiérarchie pour inclure ou exclure des définitions de types. Une définition de type est une copie maîtresse d'une commande personnalisée, pouvant être utilisée par plusieurs VIs.

En outre, le menu Afficher et les menus locaux incluent les options **Visualiser tous les VIs** et **Chemin complet du VI** dans l'**étiquette**, auxquelles vous ne pouvez pas accéder à partir de la barre d'outils.

Lorsque vous déplacez l'outil Doigt sur les objets de la fenêtre Hiérarchie, LabVIEW affiche le nom du VI au-dessous de l'icône VI.

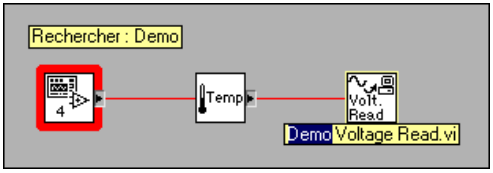
Utilisez la touche de tabulation pour basculer entre les outils de fenêtre Flèche et Défilement. Cette fonction est utile pour déplacer les nœuds de la fenêtre Hiérarchie sur le diagramme.

Vous pouvez faire glisser un nœud de VI ou de sous-VI sur le diagramme ou le copier dans le presse-papiers en cliquant sur le nœud. <Maj>-cliquez sur un nœud de VI ou de sous-VIs pour sélectionner plusieurs objets à copier sur d'autres diagrammes ou d'autres faces-avant. Si vous double-cliquez sur le nœud d'un VI ou d'un sous-VI, vous ouvrez sa face-avant.

Tous les VIs possédant des sous-VIs ont un bouton en forme de flèche près du VI. Vous pouvez utiliser ce bouton pour afficher ou masquer les sous-VIs. Si vous cliquez sur le bouton en forme de flèche rouge ou si vous double-cliquez sur le VI lui-même, vous affichez les sous-VIs de ce VI. Un bouton en forme de flèche noire sur le nœud d'un VI signifie que tous les sous-VIs sont affichés. Vous pouvez également ouvrir le menu local du nœud d'un VI ou d'un sous-VI pour accéder à un menu avec des options, comme celles permettant d'afficher ou de masquer des sous-VIs, d'ouvrir la face-avant du VI ou du sous-VI, de modifier l'icône du VI, etc.

Recherche parmi la hiérarchie

Vous pouvez également rechercher par leur nom des nœuds visibles dans la fenêtre Hiérarchie. Commencez la recherche en tapant le nom du nœud, n'importe où sur la fenêtre. Lorsque vous tapez le texte, une chaîne de recherche apparaît, affichant le texte à mesure que vous le tapez, tout en procédant à une recherche parmi la hiérarchie. L'illustration suivante montre la recherche parmi la hiérarchie.

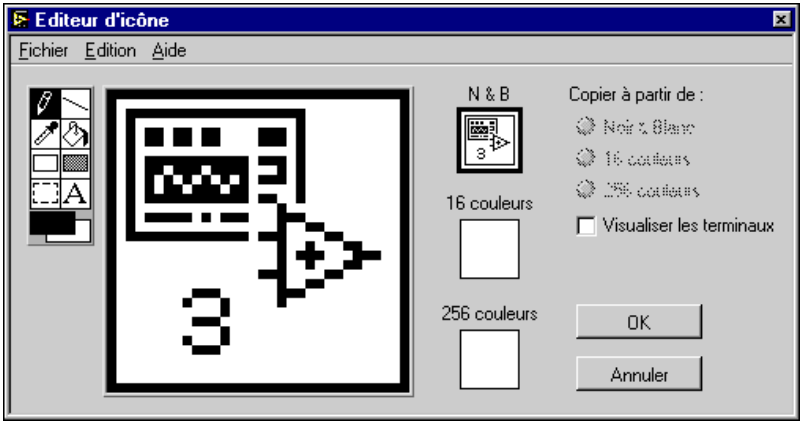


Une fois que le nœud correct a été trouvé, vous pouvez appuyer sur <Entrée> pour rechercher le nœud suivant correspondant à la chaîne de recherche, ou vous pouvez appuyer sur <Maj-Entrée> pour trouver le nœud précédent correspondant à la chaîne de recherche.

Icône et connecteur

Chaque VI possède une icône par défaut affichée en haut à droite de la face-avant et du diagramme. Pour les VIs, l'icône de VI LabVIEW s'affiche par défaut, avec une valeur numérique indiquant le nombre de nouveaux VIs que vous avez ouverts depuis le lancement de LabVIEW. Utilisez l'Editeur d'icônes pour personnaliser l'icône en activant ou désactivant des pixels séparément. Pour activer l'Editeur d'icônes, ouvrez le menu local sur l'icône par défaut en haut à droite de la fenêtre de la face-avant, puis sélectionnez **Editer l'icône**.

L'illustration suivante montre la fenêtre de l'Editeur d'icône. Utilisez les outils situés sur la gauche pour concevoir l'icône dans la zone d'édition des pixels. Une image de la taille réelle de l'icône apparaît dans l'une des boîtes sur la droite de la zone d'édition.



Les outils situés à gauche de la zone d'édition représentent les fonctions suivantes :

Outil Crayon : trace et efface pixel par pixel.

Outil Ligne : trace des lignes droites. Appuyez sur <Maj>, puis faites glisser cet outil pour tracer des lignes horizontales, verticales et diagonales.

Outil Pipette : copie la couleur du premier plan d'un élément dans l'icône.

Outil Pot de peinture : remplit la zone indiquée avec la couleur du premier plan.

Outil Rectangle : trace une bordure rectangulaire dans la couleur du premier plan. Double-cliquez sur cet outil pour cadrer l'icône dans la couleur du premier plan.

Outil Rectangle plein : trace un rectangle dont les bordures ont la couleur du premier plan mais qui est rempli avec la couleur de l'arrière-plan. Double-cliquez pour entourer l'icône d'un cadre de la couleur du premier plan et pour la remplir avec la couleur de l'arrière-plan.

Outil Marquise : sélectionne une zone de l'icône à déplacer, copier ou modifier.

Outil Texte : insère du texte dans le dessin de l'icône.

Outil Plans : affiche les couleurs actuelles du premier plan et de l'arrière-plan. Cliquez sur chaque plan pour obtenir une palette de couleurs à partir de laquelle vous pouvez choisir de nouvelles couleurs.

Les boutons à droite de l'écran d'édition permettent d'effectuer les fonctions suivantes :

- **OK** : enregistre votre dessin comme l'icône du VI et vous ramène à la face-avant.
- **Annuler** : vous ramène à la face-avant sans enregistrer les changements.

Selon le type de moniteur que vous utilisez, vous pouvez concevoir et enregistrer une icône différente pour les modes monochrome, 16 couleurs et 256 couleurs. L'éditeur passe par défaut sur **Noir & Blanc**, mais vous pouvez cliquer sur les autres options de couleur pour changer de mode.

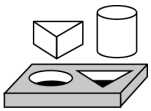
☞ **Remarque** ***Si vous ne concevez qu'une icône en couleur, elle ne pourra pas apparaître dans une sous-palette de la palette* Fonctions *si vous placez le VI dans le répertoire* `*.lib`. *De plus, l'icône ne sera ni imprimée ni affichée sur un moniteur noir et blanc.***

Le *connecteur* est l'interface de programmation avec un VI. Si vous utilisez les commandes ou les indicateurs de la face-avant pour transmettre des données vers ou en provenance de sous-VIs, ces commandes ou indicateurs ont besoin de terminaux sur le cadre connecteur. Vous pouvez définir les connexions en choisissant le nombre de terminaux que vous souhaitez pour le VI et en assignant une commande ou un indicateur de la face-avant à chacun de ces terminaux.

Pour définir un connecteur, sélectionnez **Visualiser le connecteur** dans le menu local de l'icône située dans l'angle supérieur droit de la fenêtre Face-avant.

L'icône du connecteur remplace l'icône situeé en haut à droite de la fenêtre. LabVIEW sélectionne un terminal correspondant à votre VI avec des terminaux pour les commandes sur le côté gauche du cadre connecteur, et des terminaux pour les indicateurs sur le côté droit. Le nombre de terminaux sélectionnés dépend du nombre de commandes et d'indicateurs se trouvant sur votre face-avant.

Chaque rectangle sur le connecteur représente la zone d'un terminal, et vous pouvez utiliser les rectangles comme une entrée ou une sortie du VI. Si nécessaire, vous pouvez sélectionner un terminal différent pour votre VI. Pour cela, ouvrez le menu local de l'icône, sélectionnez **Visualiser le connecteur**, ouvrez de nouveau le menu local, puis sélectionnez **Modèles**.



Exercice 2-3. Créer une icône et un connecteur

Votre objectif est de créer une icône et un connecteur pour un VI.

Pour utiliser un VI comme sous-VI, vous devez créer une icône pour le représenter lorsque vous le placerez dans le diagramme d'un autre VI, et un cadre connecteur grâce auquel vous pourrez connecter des entrées et des sorties. LabVIEW fournit plusieurs outils avec lesquels vous pouvez créer ou éditer une icône pour vos VIs.

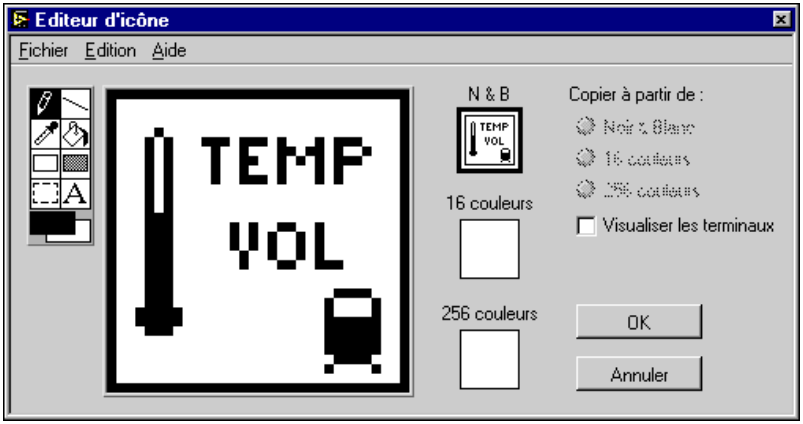
L'icône d'un VI représente celui-ci comme un sous-VI dans le diagramme d'autres VIs. Il peut s'agir de la représentation graphique de l'objectif du VI, ou d'un texte décrivant le VI.

1. Ouvrez le VI `Temp & Vol.vi` dans le répertoire `LabVIEW\Activity\Solution`.
2. Dans la face-avant, ouvrez le menu local de l'icône dans le coin supérieur droit et sélectionnez **Editer l'icône…** Vous pouvez également double-cliquer sur l'icône pour appeler l'Editeur d'icônes.

☞ **Remarque** ***Vous ne pouvez accéder à l'icône/connecteur d'un VI qu'à partir de la face-avant.***

3. Effacez l'icône par défaut. Avec l'outil Marquise, qui apparaît sous la forme d'un rectangle en pointillés, cliquez et faites glisser le curseur sur la section que vous souhaitez supprimer, puis appuyez sur la touche <Supprimer>. Vous pouvez également double-cliquer sur le rectangle plein dans la boîte à outils pour effacer l'icône.
4. Dessinez un thermomètre avec l'outil Crayon.

5. Créez le texte avec l'outil Texte. Pour changer la police, double-cliquez sur cet outil. Votre icône doit ressembler à celle de l'illustration suivante.



6. Fermez l'Editeur d'icônes en cliquant sur **OK**. La nouvelle icône apparaît dans le cadre à la place de l'ancienne.

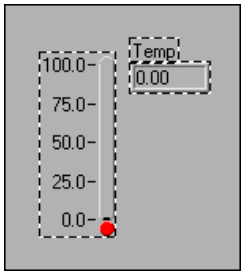
7. Définissez le terminal du connecteur en ouvrant le menu local dans le cadre icône sur la face-avant et en choisissant **Visualiser le connecteur**. Par défaut, LabVIEW sélectionne un terminal basé sur le nombre de commandes et d'indicateurs de la face-avant. Puisqu'il y a deux objets sur la face-avant, le connecteur possède deux terminaux, comme indiqué à gauche.

8. Ouvrez le menu local sur le cadre connecteur et sélectionnez **Rotation de 90 degrés**. Remarquez la façon dont le cadre connecteur change, comme indiqué à gauche.

9. Assignez les terminaux à Température et à Volume.

   a. Cliquez à l'intérieur du terminal supérieur du connecteur. Le curseur se transforme automatiquement en outil Bobine, et le terminal devient noir.

   b. Cliquez sur l'indicateur Temp. Une ligne discontinue mobile encadre l'indicateur, comme indiqué dans l'illustration suivante. Le terminal sélectionné apparaît alors dans une couleur correspondant au type de données de la commande ou de l'indicateur sélectionné.



Si vous cliquez dans une zone ouverte de la face-avant, la ligne discontinue disparaît et le terminal sélectionné s'assombrit, indiquant que vous avez associé l'indicateur à ce terminal. Si le terminal est de couleur blanche, vous n'avez pas effectué correctement la connexion.

c. Répétez les étapes a et b pour associer le terminal du bas avec l'indicateur Volume.

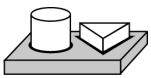
d. Ouvrez le menu local sur le connecteur et sélectionnez **Visualiser l'icône…**

10. Enregistrez le VI en choisissant **Fichier»Enregistrer**.

Ce VI est désormais achevé et prêt à être utilisé comme sous-VI dans d'autres VIs. L'icône représente le VI dans le diagramme du VI appelant. Le connecteur (avec deux terminaux) retourne la température et le volume.

☞ **Remarque** ***Le connecteur précise les entrées et sorties d'un VI lorsque vous l'utilisez comme un sous-VI. Gardez à l'esprit que vous ne pouvez utiliser les* commandes *de la face-avant que comme des entrées ; de même, vous ne pouvez utiliser les* indicateurs *de la face-avant que comme des sorties.***

11. Fermez le VI en choisissant **Fichier»Fermer**.

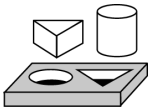


Fin de l'exercice 2-3.

Ouverture, utilisation et modification de sous-VIs

Vous pouvez ouvrir un VI utilisé comme un sous-VI à partir du diagramme du VI appelant en double-cliquant sur l'icône du sous-VI ou en sélectionnant **Projet»Sous-VIs appelés par ce VI**. Une palette contenant tous les sous-VIs du VI appelant s'affiche alors. Sélectionnez le sous-VI que vous souhaitez ouvrir.

Tous les changements que vous effectuez dans un sous-VI n'affectent que la version en mémoire tant que vous ne l'avez pas enregistré. Les changements affectent toutes les instances du sous-VI, et pas uniquement le nœud utilisé pour éditer le VI.



Exercice 2-4. Appeler un sous-VI

Votre objectif est de construire un VI en utilisant `le VI Temp & Vol` ***comme sous-VI.***

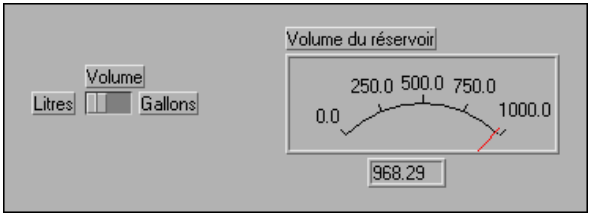
Le VI Temp & Vol construit dans l'exercice 2-1 retourne une température et un volume. Vous devez prendre une lecture de volume et la convertir en gallons lorsqu'un interrupteur est enfoncé.

Face-avant

1. Ouvrez une nouvelle face-avant en sélectionnant **Fichier»Nouveau**.

2. Sélectionnez un Interrupteur horizontal dans la palette **Commandes»Booléen** et nommez-le `volume`. Placez des étiquettes libres sur la face-avant pour y indiquer `Litres` et `Gallons` avec l'outil Texte.

3. Sélectionnez un vu-mètre dans **Commandes»Numérique** et placez-le sur la face-avant. Appelez-le `Volume du réservoir`.

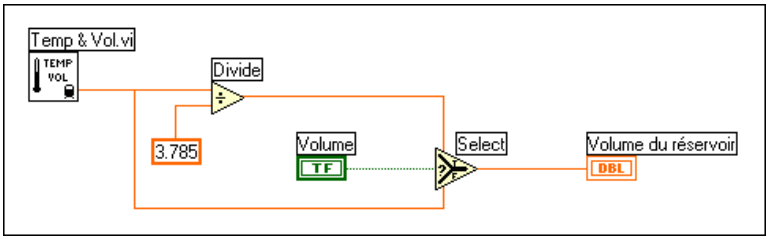


4. Changez la gamme du vu-mètre pour accepter des valeurs s'inscrivant entre 0,0 et 1000,0. Avec l'outil Doigt, double-cliquez sur la limite supérieure et entrez la valeur 1000,0. Basculez sur l'outil Flèche et ajustez la dimension du vu-mètre en faisant glisser l'un des coins et en étendant la commande.

Diagramme

5. Passez au diagramme en sélectionnant **Fenêtres»Diagramme**.

6. Ouvrez le menu local dans une zone disponible du diagramme et choisissez **Fonctions»Sélectionner un VI…** Une boîte de dialogue apparaît. Sélectionnez le VI Temp & Vol.vi dans le répertoire `LabVIEW\Activity`. Cliquez sur Ouvrir dans la boîte de dialogue LabVIEW pour placer le VI “Temp & Vol.vi” dans le diagramme.
7. Ajoutez les autres objets sur le diagramme comme indiqué dans l’illustration suivante.



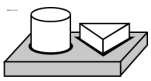
Constante numérique (**Fonctions»Numérique**) : permet d’ajouter une constante numérique dans le diagramme. Assignez la valeur `3,785` à la constante avec l’outil Texte. Il s’agit du facteur de conversion pour passer des litres aux gallons.

Fonction Sélectionner (**Fonction»Comparaison**) : retourne la valeur liée à l’entrée Vrai (TRUE) ou Faux (FALSE), selon la valeur de l’entrée booléenne.

Fonction Diviser (**Fonctions»Numérique**) : divise la valeur en litres par 3,785 pour la convertir en gallons.

8. Câblez les objets du diagramme comme indiqué.

9. Revenez à la face-avant et cliquez sur le bouton **Exécuter** dans la barre d’outils. Le vu-mètre représente la valeur en litres.
10. Cliquez sur l’interrupteur pour sélectionner `Gallons` et cliquez sur le bouton **Exécuter**. Le vu-mètre affiche alors la valeur en gallons.
11. Enregistrez le VI sous le nom `Utilisation de Temp & Vol.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l’exercice 2-4.

Comment mettre au point un VI ?

Vous ne pouvez pas compiler ni exécuter un VI invalide. Normalement, le VI est invalide lorsque vous le créez ou l'éditez, tant que vous n'avez pas câblé toutes les icônes dans le diagramme. Si le VI reste invalide lorsque vous avez terminé, essayez de sélectionner **Supprimer les fils incorrects** dans le menu **Edition**. Ceci permet souvent de corriger les erreurs d'un VI invalide.

Lorsque votre VI n'est pas exécutable, une flèche *brisée* apparaît à la place du bouton **Exécuter**. Pour afficher la liste des erreurs, cliquez sur ce bouton. Cliquez sur l'une des erreurs de la liste, puis cliquez sur **Rechercher** pour mettre en surbrillance l'objet ou le terminal ayant généré l'erreur.

Vous pouvez animer l'exécution du diagramme du VI en cliquant sur le bouton **Animer l'exécution**. L'exécution en mode Animation est souvent utilisée avec le mode pas à pas pour visualiser le flux des données dans un diagramme.

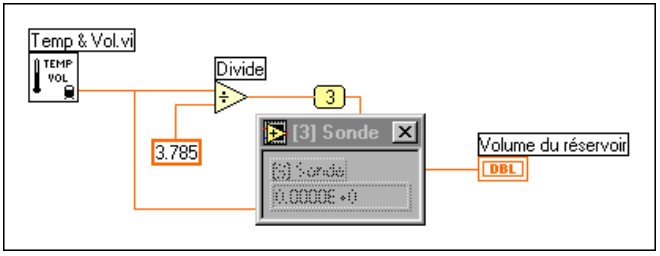
Dans l'optique d'une mise au point, vous souhaiterez peut-être exécuter un diagramme nœud par nœud, selon le mode pas à pas. Pour activer ce mode, cliquez sur le bouton **Exécution détaillée** ou sur le bouton **Exécuter sans détailler**. Ceci fait clignoter le premier nœud, indiquant que celui-ci est prêt pour l'exécution. Vous pouvez alors cliquer de nouveau sur le bouton **Exécution détaillée** ou **Exécuter sans détailler** pour exécuter le nœud, puis passer au nœud suivant. Si le nœud est une structure ou un VI, vous pouvez sélectionner le bouton **Exécuter sans détailler** pour *exécuter* le nœud mais *sans activer le mode pas à pas* à l'intérieur de ce nœud. Par exemple, si le nœud est un sous-VI et si vous cliquez sur le bouton **Exécuter sans détailler**, vous exécutez le sous-VI et passez au nœud suivant, mais vous ne pouvez pas voir la façon dont les nœuds du sous-VI s'exécutent. Pour activer le mode pas à pas au sein d'une structure ou d'un sous-VI, sélectionnez le bouton **Exécution détaillée**.

Cliquez sur le bouton **Sortie** pour terminer l'exécution des nœuds du diagramme et/ou sortir du mode pas à pas. Pour plus d'informations sur la mise au point, consultez le chapitre 4, *Exécution et mise au point des VIs et sous-VIs*, dans le *Manuel de référence de programmation en G*.

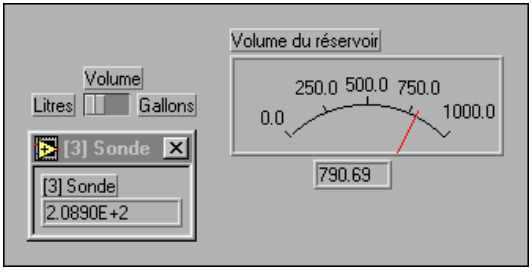
Exercice 2-5. Mettre au point un VI dans LabVIEW

Votre objectif est d'utiliser l'outil Sonde et la fenêtre Sonde, et d'examiner le flux des données dans le diagramme en utilisant la fonction d'exécution en mode Animation.

1. Ouvrez `Utilisation de Temp & Vol.vi` dans le répertoire `LabVIEW\Activity`.
2. Sélectionnez l'option **Fenêtres»Diagramme**.
3. Si la palette Outils n'est pas ouverte, sélectionnez l'option **Fenêtres»Palette d'outils**.
4. Sélectionnez l'outil Sonde dans la palette **Outils**. Cliquez avec l'outil Sonde sur le fil de liaison connecté à la sortie de la fonction Diviser. Une fenêtre Sonde apparaît sur l'écran, intitulée `Sonde 1`, avec un glyphe jaune portant le numéro de la sonde, comme indiqué dans l'illustration suivante. La fenêtre Sonde reste ouverte, même si vous basculez sur la face-avant.



5. Revenez sur la face-avant. Déplacez la fenêtre Sonde afin de voir à la fois la sonde et les valeurs du volume, comme indiqué dans l'illustration suivante. Exécutez le VI. Le volume en gallons apparaît dans la fenêtre Sonde, tandis que le `Volume du réservoir` affiche la valeur en litres.

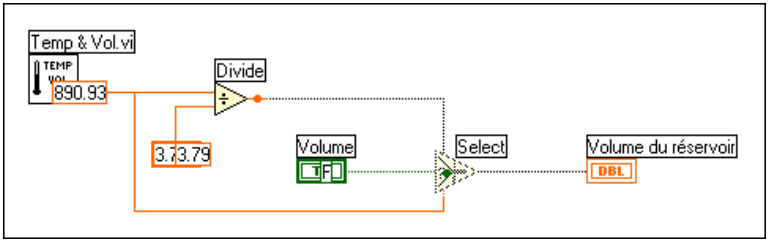


Remarque ***Les valeurs de volume qui apparaissent sur votre écran peuvent différer de celles affichées dans cette illustration. Reportez-vous à la section Conversion numérique du chapitre 3, Boucles et graphes déroulants, pour plus d'informations.***

6. Fermez la fenêtre Sonde en cliquant sur l'icône de fermeture en haut à gauche de la barre de titre de la fenêtre Sonde.

Pour une mise au point efficace, vous pouvez également examiner le flux des données dans le diagramme avec la fonction d'exécution en mode Animation.

7. Revenez sur le diagramme du VI.
8. Commencez l'exécution en mode Animation en cliquant sur le bouton **Animer l'exécution**, dans la barre d'outils. Le bouton **Animer l'exécution** se transforme en une ampoule allumée.
9. Cliquez sur le bouton **Exécuter** pour exécuter le VI. Les bulles qui se déplacent représentent le flux de données dans le VI. Notez que les valeurs des données apparaissent au niveau des fils de liaison ; il s'agit des valeurs transportées par les fils à cet instant donné, comme indiqué dans le diagramme suivant, comme si vous sondiez le fil de liaison.

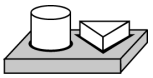


Vous pouvez également utiliser les boutons du mode pas à pas si vous souhaitez vous déplacer le long du code graphique, une étape à la fois.

10. Débutez le mode pas à pas en cliquant sur le bouton **Exécuter sans détailler** dans la barre d'outils.

11. Exécutez de façon détaillée le sous-VI `Temp & Vol.vi` en cliquant sur le bouton **Exécution détaillée** dans la barre d'outils. Si vous cliquez sur ce bouton, vous ouvrez la face-avant et le diagramme de votre sous-VI `Temp & Vol.vi`. Cliquez sur le bouton **Exécuter sans détailler** jusqu'à la fin de l'exécution du VI.

12. Terminez l'exécution du diagramme en cliquant sur le bouton **Sortie** dans la barre d'outils. Si vous cliquez sur ce bouton, vous achevez toutes les séquences qui restent dans le diagramme.



Fin de l'exercice 2-5.

3

Boucles et graphes déroulants

Ce chapitre introduit les structures et explique les concepts de base concernant les graphes déroulants, la boucle While et la boucle For. Ce chapitre propose également des exercices qui expliquent comment réaliser les tâches suivantes :

- Comprendre les différents modes d'affichage
- Utiliser une boucle While et un graphe déroulant
- Modifier l'action mécanique d'un interrupteur booléen
- Assurer le cadencement d'exécution de la boucle de contrôle
- Utiliser un registre à décalage
- Créer un graphe déroulant multicourbes
- Utiliser une boucle For

Qu'est-ce qu'une structure ?

Une *structure* est un élément de contrôle d'un programme. Les structures contrôlent le flux des données dans un VI. Le G possède cinq structures : la boucle While, la boucle For, la structure Condition, la structure Séquence et la boîte de calcul. Ce chapitre introduit les structures correspondant à la boucle While et à la boucle For, ainsi que le graphe déroulant et le registre à décalage. La structure Condition, la structure Séquence et la boîte de calcul sont détaillées au chapitre 4, *Structure Condition, structure Séquence et boîte de calcul*.

Les boucles While et For étant des structures de base pour la programmation en G, vous les trouverez dans la plupart des exemples en G, ainsi que dans les exercices figurant dans ce manuel. Vous pouvez obtenir plus d'informations sur les boucles au chapitre 19, *Structures*, dans le *Manuel de référence de programmation en G*.

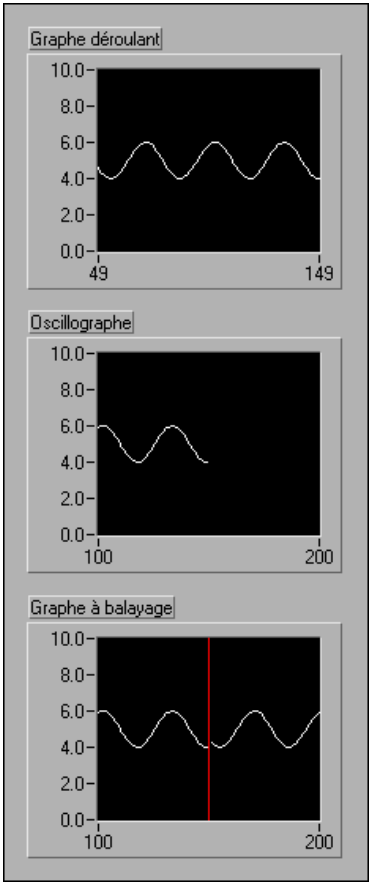
Pour des exemples de structures, reportez-vous à la bibliothèque `Examples\General\structs.llb`. Pour des exemples de graphes déroulants, consultez `Examples\General\Graphs\charts.llb`.

Graphes déroulants

Un *graphe déroulant* est un indicateur permettant de tracer des données numériques ; il est périodiquement mis à jour avec de nouvelles données. Il existe deux types de graphes déroulants dans la palette **Commandes» Graphe** : le graphe déroulant et le graphe déroulant d'intensité. Vous pouvez personnaliser les graphes déroulants pour les faire correspondre à vos critères d'affichage de données ou pour afficher davantage d'informations. Les caractéristiques disponibles des graphes déroulants incluent les éléments suivants : une barre de défilement, une légende, une palette, un afficheur numérique et la représentation des échelles en fonction du temps. Pour plus d'informations sur les graphes déroulants, consultez le chapitre 15, *Commandes et indicateurs de graphe et graphe déroulant*, dans votre *Manuel de référence de programmation en G.*

Modes d'affichage

L'illustration suivante montre les trois options d'affichage d'un graphe déroulant disponibles dans le sous-menu local **Opérations sur les données»Mode de mise à jour** : **Graphe déroulant, Oscillographe et Graphe à balayage**. Le mode par défaut est Graphe déroulant.

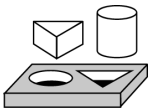


Mises à jour plus rapides de graphe déroulant

Vous pouvez transmettre un tableau à plusieurs valeurs multiples à un graphe déroulant. Le graphe déroulant traite ces entrées comme de nouvelles données pour un tracé unique. Reportez-vous à l'exemple `Charts.vi` situé dans `Examples\General\Graphs\charts.llb`.

Tracés superposés et tracés empilés

Vous pouvez afficher plusieurs tracés multiples sur un graphe déroulant grâce à une échelle verticale unique, pour des "tracés superposés", ou grâce à plusieurs échelles verticales, pour des "tracés empilés". Reportez-vous à l'exemple `Charts.vi` situé dans `Examples\General\Graphs\charts.llb`.



Exercice 3-1. Essais avec les différents modes d'affichage

Votre objectif est d'utiliser un graphe déroulant en mode graphe déroulant, mode oscillographe et mode graphe à balayage, pendant l'exécution du VI.

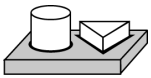
1. Ouvrez le VI "Graphes déroulants" (`Charts.vi`)situé dans la bibliothèque `LabVIEW\Examples\General\Graphs\charts.11b`.
2. Exécutez le VI.

Le mode graphe déroulant est doté d'un affichage de mise à l'échelle similaire à un enregistreur de graphe déroulant à bande perforée. A la réception de chaque nouvelle valeur, celle-ci est représentée sur la marge de droite tandis que les anciennes valeurs glissent sur la gauche.

Le mode oscillographe possède une fonction d'affichage similaire à un oscilloscope. Chaque fois que le VI reçoit une nouvelle valeur, il trace la valeur à la droite de la dernière valeur. Lorsque le tracé atteint la bordure droite de la zone de traçage, le VI efface le tracé et commence à retracer depuis la bordure gauche. L'oscillographe est considérablement plus rapide que le graphe déroulant car il ne souffre pas du délai de traitement supplémentaire impliqué dans le défilement.

Le mode graphe à balayage se comporte beaucoup comme l'oscillographe, mais ne s'efface pas lorsque les données atteignent la bordure droite. A la place, une ligne verticale mobile marque le début des nouvelles données et se déplace sur l'écran tandis que le VI ajoute de nouvelles données.

3. Alors que le VI est toujours en exécution, ouvrez le menu local n'importe où sur le graphe déroulant, sélectionnez **Mode de mise à jour**, puis faites passer le mode courant sur le mode d'un autre graphe déroulant. Notez la différence entre les divers graphes déroulants et les modes.
4. Arrêtez et fermez le VI.

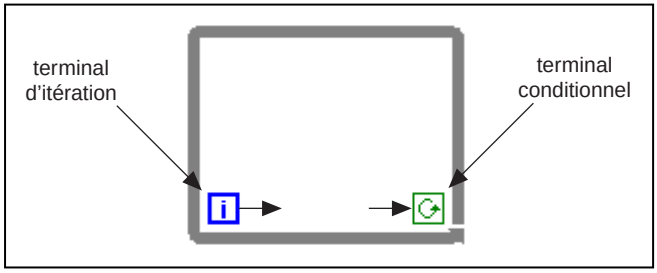


Fin de l'exercice 3-1.

Boucles While

Une boucle While est une structure répétant une section de code jusqu'à ce qu'une condition soit remplie. Cette boucle est comparable à une boucle Do ou à une boucle Repeat-Until dans un langage de programmation traditionnel.

La boucle While, présentée dans l'illustration suivante, est une boîte dont les dimensions sont réglables. Utilisez cette boucle pour exécuter le diagramme se trouvant à l'intérieur, jusqu'à ce que la valeur booléenne transmise au terminal conditionnel (qui est un terminal d'entrée) soit Faux (FALSE). Le VI vérifie le terminal conditionnel à la fin de chaque itération, c'est pourquoi la boucle While s'exécute toujours au moins une fois. Le terminal d'itération est un terminal numérique de sortie qui fournit le nombre de fois que la boucle a été exécutée. Le compte de l'itération commence toujours à zéro, donc si la boucle ne s'exécute qu'une fois, le terminal d'itération génère la valeur 0.



La boucle While est équivalente au pseudo-code suivant :

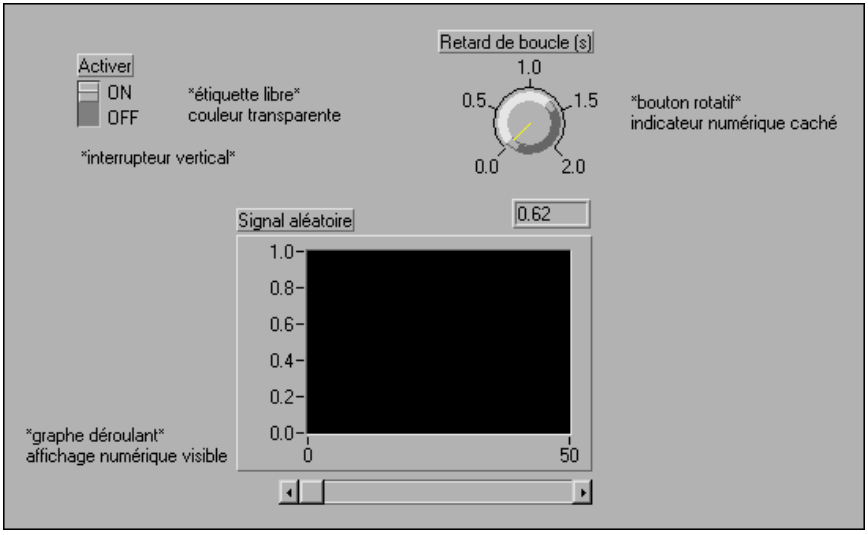
```
Faire
Exécuter le diagramme à l'intérieur de la boucle (qui
définit la condition)
Jusqu'à ce que Condition soit VRAIE
```

Exercice 3-2. Utiliser une boucle While et un graphe déroulant

Votre objectif est d'utiliser une boucle While et un graphe déroulant pour acquérir et afficher les données en temps réel.

Vous allez construire un VI qui génère des données aléatoires et les affiche dans un graphe déroulant. Un bouton rotatif sur la face-avant ajuste la fréquence de la boucle entre 0 et 2 secondes, tandis qu'un interrupteur arrête le VI. Vous devez modifier l'action mécanique de l'interrupteur pour éliminer le besoin d'actionner l'interrupteur chaque fois que vous exécutez le VI. Utilisez la face-avant de l'illustration suivante pour démarrer.

Face-avant



1. Ouvrez une nouvelle face-avant en sélectionnant **Fichier»Nouveau**.

2. Placez un interrupteur vertical (**Commandes»Booléen**) sur la face-avant. Libellez l'interrupteur `Activer`.

3. Utilisez l'outil Texte pour créer du texte libre pour les étiquettes `MARCHE` et `ARRET`. Sélectionnez l'outil Texte, puis tapez le texte de l'étiquette. Avec l'outil Pinceau ci-contre, rendez transparente la bordure du texte libre en sélectionnant le T dans le coin inférieur gauche de la palette **Couleur**.

4. Placez un graphe déroulant (**Commandes»Graphe**) sur la face-avant. Libellez le graphe déroulant `Signal aléatoire`. Le graphe déroulant affiche des données aléatoires en temps réel.

Remarque ***Assurez-vous de sélectionner un graphe déroulant et non un graphe. Dans la palette* Graphe*, le graphe déroulant se trouve le plus à gauche.***

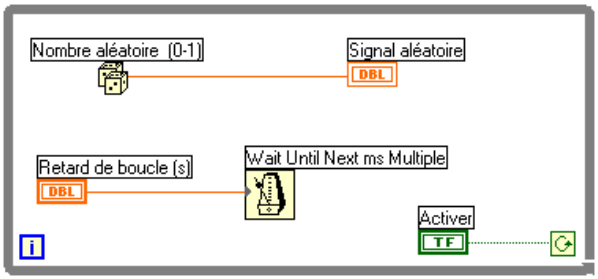
5. Ouvrez le menu local sur le graphe déroulant et choisissez **Visualiser»Palette**, puis **Visualiser»Légende** pour masquer la palette et la légende. L'afficheur numérique affiche la valeur la plus récente. Ouvrez ensuite le menu local du graphe déroulant, puis choisissez **Visualiser»Afficheur numérique** et **Visualiser»Barre de défilement**.

6. Mettez de nouveau à l'échelle le graphe déroulant pour qu'il s'inscrive entre `0,0` et `1,0`. Utilisez l'outil Texte pour remplacer la limite supérieure `10,0` par `1,0`.

7. Placez un bouton rotatif (**Commandes»Numérique**) sur la face-avant. Libellez le bouton rotatif `temporisation (sec)`. Ceci crée un bouton sur le séquencement de la boucle While. Ouvrez le menu local sur le bouton rotatif et désélectionnez **Visualiser»Afficheur numérique** pour masquer l'afficheur numérique.
8. Mettez de nouveau à l'échelle le bouton rotatif. Avec l'outil Texte, double-cliquez sur `10,0` dans l'échelle autour du bouton rotatif, puis remplacez cette valeur par `2,0`.

Diagramme

9. Ouvrez et créez le diagramme comme représenté dans l'illustration suivante.



a. Placez la boucle While dans le diagramme en la sélectionnant dans **Fonctions»Structures**. La boucle While est une boîte aux dimensions réglables qui ne se place pas immédiatement sur le diagramme. Vous pouvez ainsi placer cette boucle et ajuster ses dimensions. Pour cela, cliquez dans une zone au-dessus et à gauche de tous les terminaux. Maintenez le bouton de la souris enfoncé et faites glisser un rectangle autour des terminaux à inclure dans la boucle.

b. Sélectionnez la fonction "Nombre aléatoire (0–1)" dans **Fonctions»Numérique**.

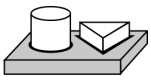
c. Câblez le diagramme comme indiqué dans le diagramme, en connectant la fonction "Nombre aléatoire (0–1)" au terminal du graphe déroulant du signal aléatoire, et en connectant l'interrupteur marche/arrêt au terminal conditionnel de la boucle While. Laissez le terminal de temporisation non câblé pour l'instant.

10. Revenez sur la face-avant et placez l'interrupteur vertical sur la position MARCHE en cliquant dessus avec l'outil Doigt.
11. Enregistrez le VI sous le nom `Signal aléatoire.vi` dans le répertoire `LabVIEW\Activity`.
12. Exécutez le VI.

La boucle While est une structure s'exécutant de façon indéfinie. Le diagramme se trouvant à l'intérieur de la boucle s'exécute aussi longtemps que la condition spécifiée est Vrai (TRUE). Dans cet exemple, tant que l'interrupteur est sur Vrai (TRUE), le diagramme continue de générer les nombres aléatoires et de les afficher sur le graphe déroulant.

13. Arrêtez le VI en cliquant sur l'interrupteur vertical. Si vous placez l'interrupteur sur la position ARRET, vous envoyez la valeur Faux (FALSE) sur le terminal conditionnel de la boucle, arrêtant ainsi la boucle.
14. Faites défiler le graphe déroulant. Cliquez et maintenez enfoncé le bouton de la souris sur l'une des flèches de la barre de défilement.
15. Effacez le buffer d'affichage et réinitialisez le graphe déroulant en ouvrant le menu local sur le graphe déroulant et en choisissant **Opérations sur les données»Effacer le graphe déroulant**.

☞ **Remarque** ***Le buffer d'affichage possède une taille par défaut de 1024 points. Vous pouvez augmenter ou diminuer cette taille en ouvrant le menu local du graphe déroulant et en choisissant* Longueur de l'historique du graphe déroulant…** ***Vous ne pouvez utiliser cette fonction que lorsque le VI n'est pas en cours d'exécution.***



Fin de l'exercice 3-2.

Action mécanique des interrupteurs booléens

Vous remarquerez peut-être que chaque fois que vous exécutez le VI, vous devez actionner l'interrupteur vertical et cliquer ensuite sur le bouton **Exécuter** dans la barre d'outils. En G, vous pouvez modifier l'action mécanique des commandes booléennes.

Vous pouvez associer six actions mécaniques à une commande booléenne :

- Commutation à l'appui
- Commutation au relâchement
- Commutation jusqu'au relâchement
- Armement à l'appui
- Armement au relâchement
- Armement jusqu'au relâchement

Les figures ci-dessous décrivent chaque interrupteur booléen et chaque action mécanique.

Commutation à l'appui : change la valeur de la commande chaque fois que vous cliquez sur la commande avec l'outil Doigt. Cette action est similaire à l'action d'un interrupteur électrique, et n'est pas affectée par le nombre de fois que le VI lit la commande.

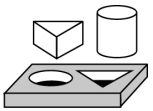
Commutation au relâchement : change la valeur de la commande uniquement après que vous ayez relâché le bouton de la souris. L'action n'est pas affectée par le nombre de fois que le VI lit la commande. Cette action est similaire à ce qui se passe lorsque vous cliquez sur la case d'une boîte de dialogue ; elle se met en surbrillance mais ne change pas tant que vous n'avez pas relâché le bouton de la souris.

Commutation jusqu'au relâchement : change la valeur de la commande lorsque vous cliquez dessus. Cette action garde la nouvelle valeur tant que vous n'avez pas relâché le bouton de la souris ; à cet instant, la commande retourne à sa valeur originale. L'action est similaire à celle d'une sonnette, et n'est pas affectée par le nombre de fois que le VI lit la commande.

Armement à l'appui : change la valeur de la commande lorsque vous cliquez dessus. Cette action garde la nouvelle valeur jusqu'à ce que le VI la lise pour la première fois ; la commande retourne alors à sa valeur originale. (Cette action se produit, que vous continuiez d'appuyer ou non sur le bouton de la souris). Cette action est similaire à celle d'un disjoncteur et s'avère utile pour arrêter les boucles While ou pour forcer le VI à accomplir une tâche une seule fois chaque fois que vous positionnez la commande.

Armement au relâchement : change la valeur de la commande uniquement après que vous avez relâché le bouton de la souris. Lorsque votre VI lit la valeur pour la première fois, la commande revient à l'ancienne valeur. Cette action garantit au moins une nouvelle valeur. Tout comme l'action de commutation au relâchement, cette action est similaire au comportement des boutons d'une boîte de dialogue ; si vous cliquez sur cette action, vous mettez en surbrillance le bouton, et le relâchement du bouton de la souris prend en compte la nouvelle valeur de la commande.

Armement jusqu'au relâchement : change la valeur de la commande lorsque vous cliquez sur la commande. Cette action retient la valeur tant que votre VI n'a pas lu la valeur une fois ou tant que vous n'avez pas relâché le bouton de la souris, selon ce qui se produit en dernier.

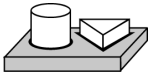


Exercice 3-3. Modifier l'action mécanique d'un interrupteur booléen

Votre objectif est de faire des essais avec les différentes actions mécaniques des interrupteurs booléens.

1. Ouvrez le VI `Signal aléatoire.vi`, comme enregistré dans l'exercice 3-2, dans le répertoire `LabVIEW\Activity`. La valeur par défaut de l'interrupteur `marche/arrêt` est Faux (FALSE).
2. Modifiez l'interrupteur vertical afin de l'utiliser uniquement pour l'arrêt du VI. Changez l'interrupteur afin d'éliminer le besoin d'actionner l'interrupteur chaque fois que vous exécutez le VI.
   a. Placez l'interrupteur vertical sur la position MARCHE grâce à l'outil Doigt.
   b. Ouvrez le menu local de l'interrupteur et choisissez **Opérations sur les données»Prendre la valeur actuelle par défaut**. Ceci met la valeur par défaut sur la position marche.
   c. Ouvrez le menu local de l'interrupteur et choisissez **Action mécanique»Armement à l'appui**.
3. Exécutez le VI. Cliquez sur l'interrupteur `marche/arrêt` pour arrêter l'acquisition. Le commutateur doit se mettre sur la position ARRET momentanément, puis revenir sur la position MARCHE.
4. Enregistrez le VI.

☞ **Remarque** ***Pour référence, LabVIEW contient un exemple illustrant ces comportements. Il s'agit du VI "Action mécanique des booléens"*** `(Mechanical action of Booleans.vi)` ***qui se trouve dans*** `Examples\General\Controls\booleans.llb`.



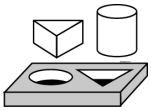
Fin de l'exercice 3-3.

Séquencement

Lorsque vous avez exécuté le VI dans l'exercice précédent, la boucle While s'est exécutée aussi rapidement que possible. Vous pouvez cependant la ralentir pour réaliser des itérations à intervalles précis avec les fonctions de la palette **Fonctions»Temps & dialogue**.

Les fonctions de séquencement expriment le temps en millisecondes (ms), mais votre système d'exploitation ne maintient peut-être pas cette précision.

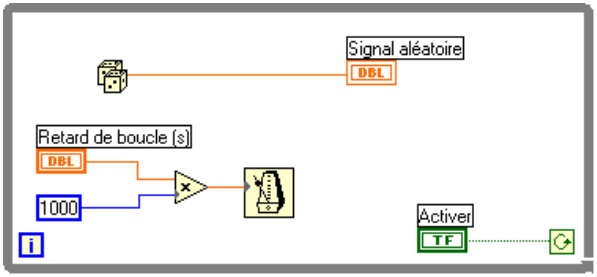
- **(Windows 95/NT)** L'horloge possède une résolution de 1 ms. Cependant, comme cela dépend du matériel, la résolution peut s'avérer inférieure sur des systèmes plus lents, comme les 80386.
- **(Windows 3.1)** L'horloge possède une résolution par défaut de 55 ms. Vous pouvez configurer LabVIEW pour avoir une résolution de 1 ms en sélectionnant **Edition»Préférences…**, en sélectionnant Performances & Disque dans le menu Chemins, et en enlevant la coche dans la case Utiliser le timer par défaut. LabVIEW n'utilise pas une résolution de 1 ms par défaut car ceci impose une charge plus lourde sur votre système d'exploitation.
- **(Macintosh)** Pour les systèmes 68K sans l'extension QuickTime, l'horloge est dotée d'une résolution de 16 2/3 ms (1/60ème de seconde). Si vous possédez un Power Macintosh ou si vous avez l'extension QuickTime installée, la résolution de l'horloge est de 1 ms.
- **(UNIX)** L'horloge a une résolution de 1 ms.



Exercice 3-4. Séquencer l'exécution d'une boucle de contrôle

Votre objectif est de séquencer l'exécution d'une boucle de contrôle et de vous assurer qu'aucune itération n'est plus courte que le nombre de millisecondes précisé.

1. Ouvrez le VI `Signal aléatoire.vi`, tel qu'il a été modifié et enregistré dans l'exercice 3-3, dans le répertoire `LabVIEW\Activity`.
2. Modifiez le VI pour générer un nouveau nombre aléatoire à un intervalle de temps précisé par le bouton rotatif, comme indiqué dans l'illustration suivante.

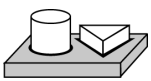


Fonction “Attendre un multiple de ms” (**Fonctions»Temps & dialogue**) : multipliez le terminal du bouton rotatif par 1000 pour convertir la valeur du bouton rotatif de secondes en millisecondes. Utilisez cette valeur comme l’entrée de la fonction “Attendre un multiple de ms”.

Fonction Multiplier (**Fonctions»Numérique**) : la fonction Multiplier multiplie la valeur du bouton rotatif par 1000 pour convertir des secondes en millisecondes.

Constante numérique (**Fonctions»Numérique**) : la constante numérique conserve la constante avec laquelle vous devez multiplier la valeur du bouton rotatif pour obtenir une quantité en millisecondes. Ainsi, si le bouton rotatif possède une valeur de 1, la boucle s’exécute une fois toutes les 1000 millisecondes (une fois par seconde).

3. Exécutez le VI. Tournez le bouton rotatif pour obtenir des valeurs différentes concernant le retard de la boucle. Notez les effets du retard de la boucle sur la mise à jour de l’affichage du VI **`Signal aléatoire.vi`**.
4. Enregistrez le VI sous le nom `Signal aléatoire avec retard.vi` dans le répertoire `LabVIEW\Activity`. Fermez le VI.



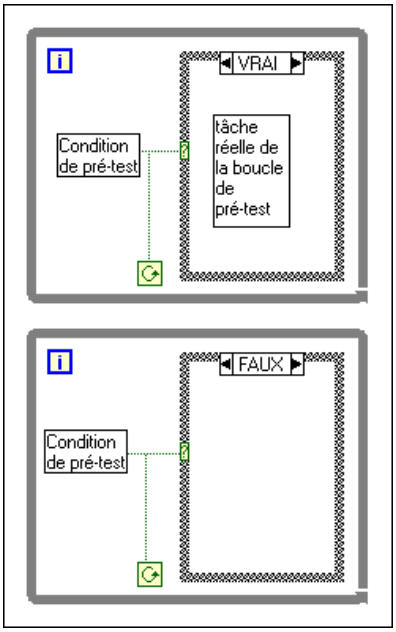
Fin de l’exercice 3-4.

Empêcher l’exécution du code dans la première itération

La boucle While s’exécute toujours au moins une fois, car le G réalise le test de la boucle pour une continuation après l’exécution du diagramme. Vous pouvez construire une boucle While qui teste à l’avance son terminal conditionnel en incluant une structure Condition dans la boucle. Câblez une entrée booléenne au terminal sélecteur de la structure Condition de sorte que le sous-diagramme de la condition FAUX s’exécute si le code dans la boucle While ne s’exécute pas. Consultez le chapitre 4, *Structure Condition, structure Séquence et boîte de calcul*, pour plus d’informations sur l’utilisation des structures Condition.

Le sous-diagramme de la condition VRAI contient les opérations devant être effectuées par la boucle While. Le test pour la continuation se produit en dehors de la structure Condition, et les résultats sont câblés au terminal conditionnel de la boucle While et au terminal sélecteur de la structure

Condition. Dans l'illustration suivante, les étiquettes représentent la condition testée à l'avance.

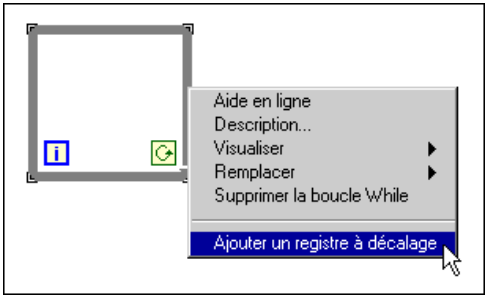


Cet exemple a le même résultat que le pseudo-code suivant :

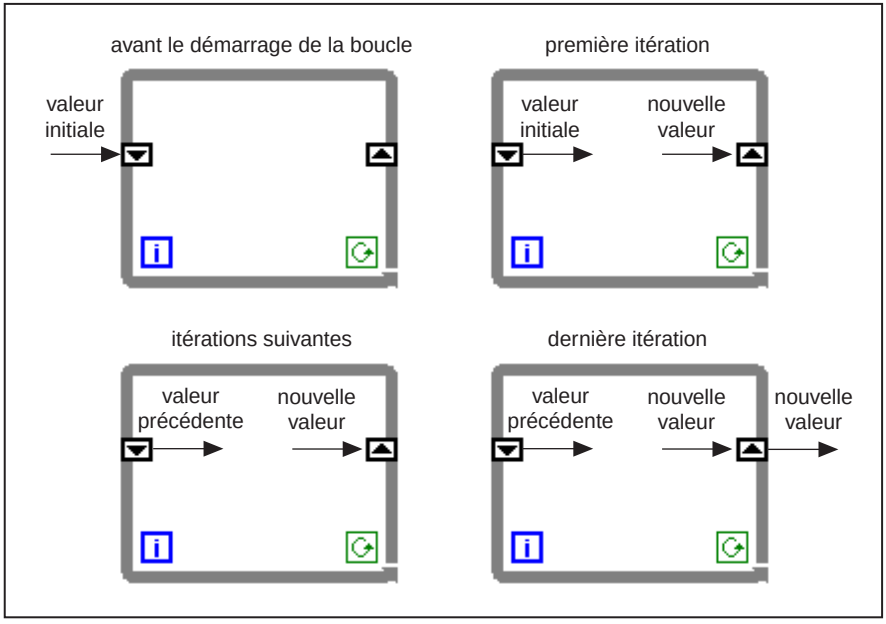
```
Tant que (condition de pré-test)
Effectuer les opérations de la boucle While
Retour au test
```

Registres à décalage

Les *registres à décalage* (disponibles pour les boucles While et les boucles For) transfèrent des valeurs d'une itération de boucle à la suivante. Vous pouvez créer un registre à décalage en ouvrant le menu local sur la bordure gauche ou droite d'une boucle et en sélectionnant **Ajouter un registre à décalage**.

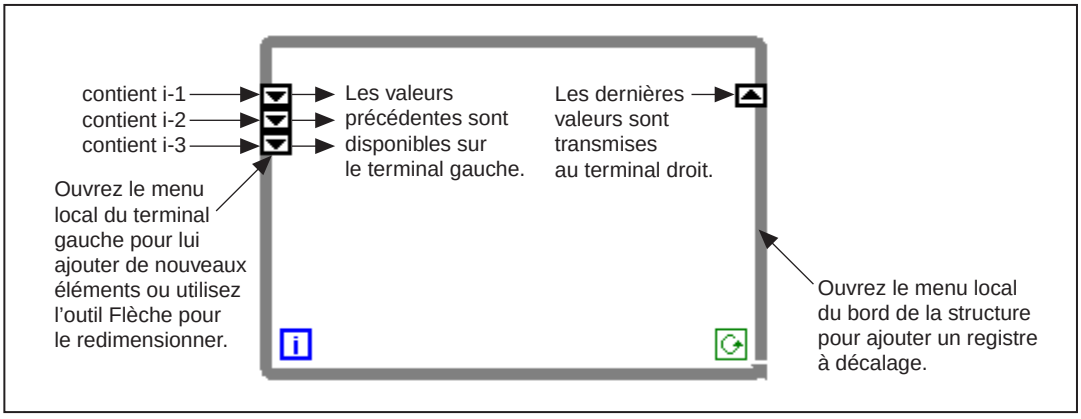


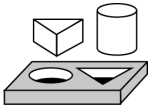
Le registre à décalage contient une paire de terminaux situés sur chacun des côtés verticaux de la bordure de la boucle. Le terminal de droite enregistre les données à la fin d'une itération. A la fin de l'itération, ces données se déplacent et apparaissent dans le terminal de gauche au début de l'itération suivante, comme indiqué dans l'illustration suivante. Un registre à décalage peut transmettre tout type de données – numériques, booléennes, chaînes de caractères, tableaux, etc. Le registre à décalage s'adapte automatiquement au type de données du premier objet que vous lui câblez.



Vous pouvez configurer le registre à décalage de façon à voir les valeurs provenant de plusieurs itérations précédentes. Cette fonction est utile pour faire la moyenne des points. Pour accéder à des valeurs provenant d'itérations précédentes, créez des terminaux supplémentaires en ouvrant

le menu local sur le terminal de gauche ou de droite et en choisissant **Ajouter un élément**. Par exemple, si un registre à décalage contient trois éléments dans le terminal de gauche, vous pouvez accéder aux valeurs provenant des trois dernières itérations, comme indiqué dans l'illustration suivante.



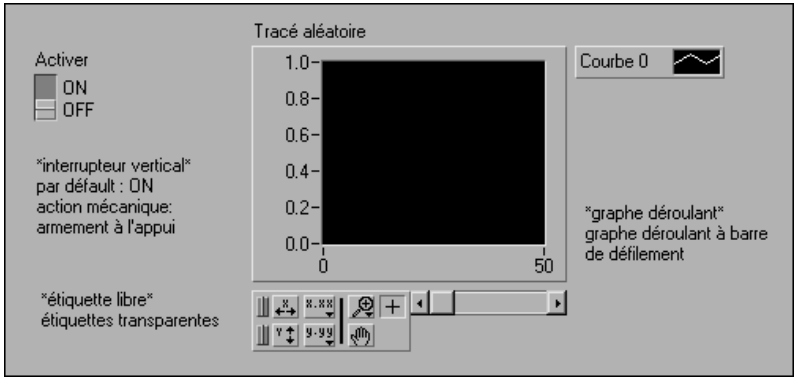


Exercice 3-5. Utiliser un registre à décalage

Votre objectif est de construire un VI affichant une moyenne d'exécution sur un graphe déroulant.

Face-avant

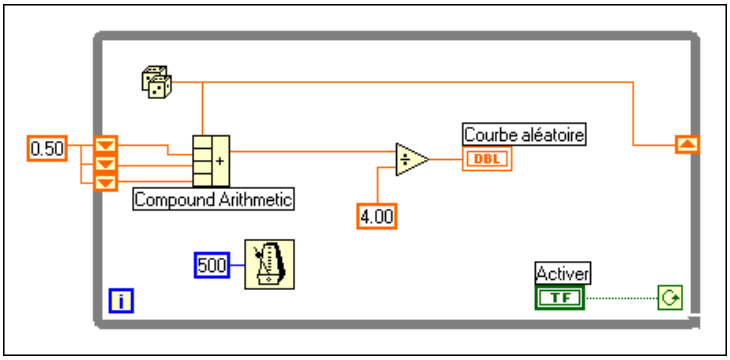
1. Ouvrez une nouvelle face-avant et créez les objets comme indiqué dans l'illustration suivante.



2. Changez l'échelle du graphe déroulant pour que sa gamme s'inscrive entre `0,0` et `2,0`.
3. Après avoir ajouté l'interrupteur vertical, ouvrez le menu local de cet interrupteur et sélectionnez **Action mécanique»Armement à l'appui**. Mettez ensuite la valeur par défaut sur l'état MARCHE en choisissant **Exécution»Prendre les valeurs actuelles par défaut**.

Diagramme

4. Construisez le diagramme de l'illustration suivante.



5. Ajoutez la boucle While (**Fonctions»Structures**) dans le diagramme, puis créez le registre à décalage.
   a. Ouvrez le menu local sur la bordure droite ou gauche de la boucle While et choisissez **Ajouter un registre à décalage**.
   b. Ajoutez un élément supplémentaire en ouvrant le menu local sur le terminal gauche du registre à décalage et en choisissant **Ajouter un élément**. Ajoutez un troisième élément de la même manière.

Fonction "Nombre aléatoire (0–1)" (**Fonctions»Numérique**) : cette fonction génère des données aléatoires s'inscrivant entre 0 et 1.

Fonction "Opérateur arithmétique" (**Fonctions»Numérique**) : dans cet exercice, la fonction "Opérateur arithmétique" retourne la somme de nombres aléatoires provenant de deux itérations. Pour ajouter plusieurs entrées, ouvrez le menu local sur une entrée, puis choisissez l'option Ajouter une entrée.

Fonction Diviser (**Fonctions»Numérique**) : dans cet exercice, la fonction Diviser retourne la moyenne de quatre nombres aléatoires.

Constante numérique (**Fonctions»Numérique**) : durant chaque itération de la boucle While, la fonction “Nombre aléatoire (0–1)” génère une valeur aléatoire. Le VI ajoute cette valeur aux trois dernières valeurs enregistrées dans les terminaux de gauche du registre à décalage. La fonction “Nombre aléatoire (0–1)” divise le résultat par quatre pour trouver la moyenne des valeurs (la valeur courante, plus les trois précédentes). La moyenne est ensuite affichée sur le graphe déroulant.

Fonction “Attendre un multiple de ms” (**Fonctions»Temps & dialogue**) : cette fonction vous assure que chaque itération de la boucle ne se produit pas plus rapidement que le nombre de millisecondes spécifié. L’entrée est égale à 500 millisecondes pour cet exercice. Si vous ouvrez le menu local sur l’icône et si vous choisissez **Visualiser»Etiquette**, l’étiquette Attendre un multiple de ms apparaît.

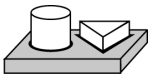
6. Ouvrez le menu local sur l’entrée de la fonction “Attendre un multiple de ms”, puis sélectionnez **Créer une constante**. Une constante numérique apparaît et est câblée automatiquement à la fonction.

7. Tapez `500` dans l’étiquette. La boucle s’exécutera alors une fois toutes les demi-secondes (500 ms).

Remarquez que le VI initialise les registres à décalage avec un nombre aléatoire. Si vous n’initialisez pas le terminal du registre à décalage, celui-ci contiendra la valeur par défaut ou la dernière valeur de l’exécution précédente, et dans ce cas les premières moyennes ne signifieront rien.

8. Exécutez le VI et observez l’opération.
9. Enregistrez ce VI sous le nom `Moyenne aléatoire.vi` dans le répertoire `LabVIEW\Activity`.

Remarque ***Pensez à initialiser les registres à décalage à l’entrée de la boucle, afin d’éviter d’incorporer des anciennes données ou les données par défaut dans vos mesures de données actuelles.***



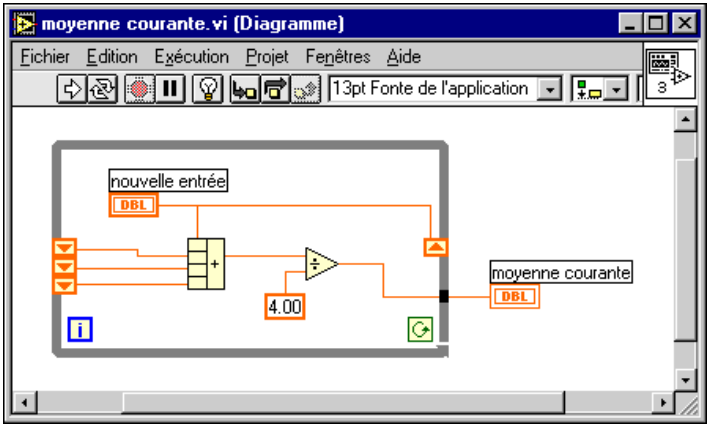
Fin de l’exercice 3-5.

Utilisation des registres à décalage non initialisés

Vous pouvez initialiser un registre à décalage en câblant une valeur se trouvant en dehors d’une boucle While ou d’une boucle For au terminal gauche du registre à décalage. Parfois cependant, vous souhaiterez peut-être exécuter un VI de façon répétée avec une boucle et un registre à décalage, de sorte qu’à chaque exécution du VI, la sortie initiale du registre à décalage soit la dernière valeur de l’exécution précédente. Pour réaliser

ceci, vous ne devez pas câbler le terminal gauche du registre à décalage à un objet situé à l'extérieur de la boucle. Si vous laissez l'entrée du terminal gauche du registre à décalage non câblée, vous préservez les informations d'état entre les exécutions ultérieures d'un VI.

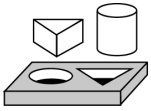
L'illustration suivante montre un exemple d'un sous-VI calculant la moyenne courante de quatre points. Le VI utilise un registre à décalage non initialisé (avec trois éléments supplémentaires) pour enregistrer les points précédents.



A chaque appel du VI, la `moyenne courante` est calculée entre la nouvelle entrée et les trois valeurs précédentes. La nouvelle valeur est alors enregistrée dans le registre à décalage, et les deux valeurs précédentes sont déplacées vers le haut dans le registre à décalage. Il n'y a pas de valeur d'entrée câblée à côté de l'entrée des registres à décalage de gauche, c'est pourquoi ces trois valeurs sont préservées pour l'exécution suivante du VI.

Ce sous-VI n’ayant aucun élément câblé au terminal de condition, il s’exécute une fois lorsqu’il est appelé. La boucle While de ce sous-VI n’est pas utilisée pour exécuter plusieurs cycles, mais plutôt pour enregistrer des valeurs dans les registres à décalage de la boucle entre les appels.

Lorsque le VI Moyenne courante.vi est chargé en mémoire, les registres à décalage non initialisés sont mis automatiquement à zéro. Si les registres à décalage sont câblés aux valeurs booléennes, la valeur initiale est Faux (FALSE).

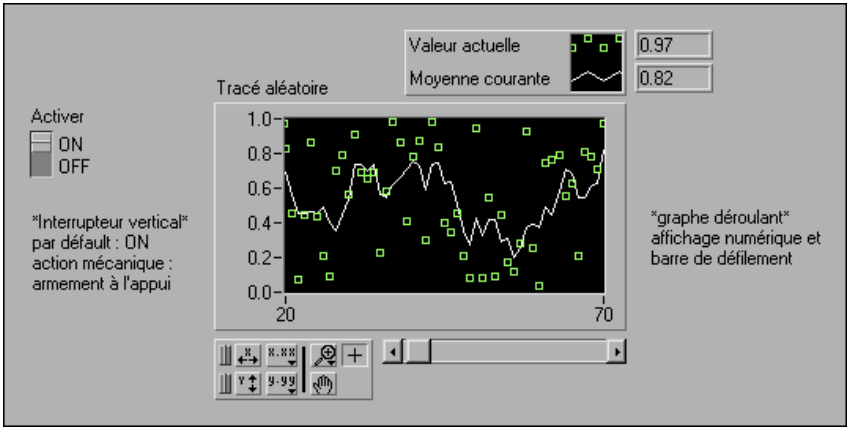


Exercice 3-6. Créer un graphe déroulant multicourbes

Votre objectif est de créer un graphe déroulant représentant plusieurs tracés.

Face-avant

1. Ouvrez le VI Moyenne courante.vi que vous avez créé dans l’exercice 3-5.
2. Modifiez la face-avant comme indiqué dans l’illustration suivante.



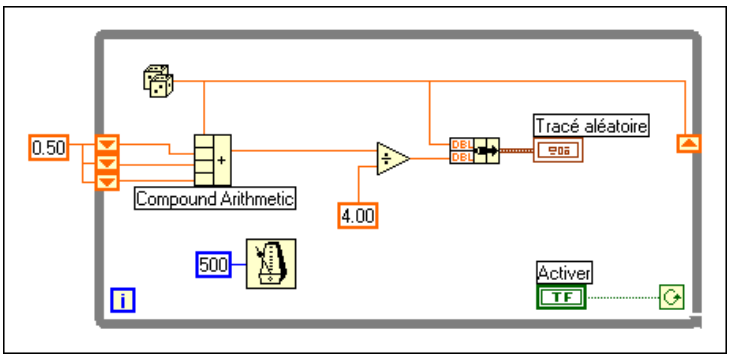
   a. Avec l’outil Flèche, tirez la légende verticalement pour inclure deux tracés.
   b. Faites apparaître l’afficheur numérique en ouvrant le menu local du graphe déroulant, et en choisissant **Visualiser»Afficheur numérique**. Déplacez la légende si c’est nécessaire.

c. Renommez le tracé 0 `Valeur actuelle` en double-cliquant sur l'étiquette avec l'outil Texte et en tapant le nouveau texte. Vous pouvez ajuster les dimensions de la zone de l'étiquette en faisant glisser l'un des coins de gauche avec l'outil Flèche. Renommez le tracé 1 `Moyenne courante`, tout comme vous l'avez fait avec le tracé 0.

d. Pour le tracé `Valeur actuelle`, changez le style d'interpolation en "non connecté", changez le style de point en carré, et optez pour la couleur verte. Pour changer le style et la couleur du tracé, ouvrez le menu local de la légende.

Diagramme

3. Modifiez le diagramme, comme indiqué dans l'illustration suivante, pour afficher la moyenne et le nombre aléatoire actuel sur le même graphe déroulant.



Fonction Assembler (**Fonctions»Cluster**) : dans cet exercice, la fonction Assembler regroupe la moyenne et la valeur actuelle du traçage sur le graphe déroulant. Le nœud assemblé apparaît comme indiqué ci-contre lorsque vous le placez dans le diagramme. Vous pouvez ajouter des éléments supplémentaires en utilisant le curseur de redimensionnement (accessible en plaçant l'outil Flèche à l'angle de la fonction) pour un agrandissement.

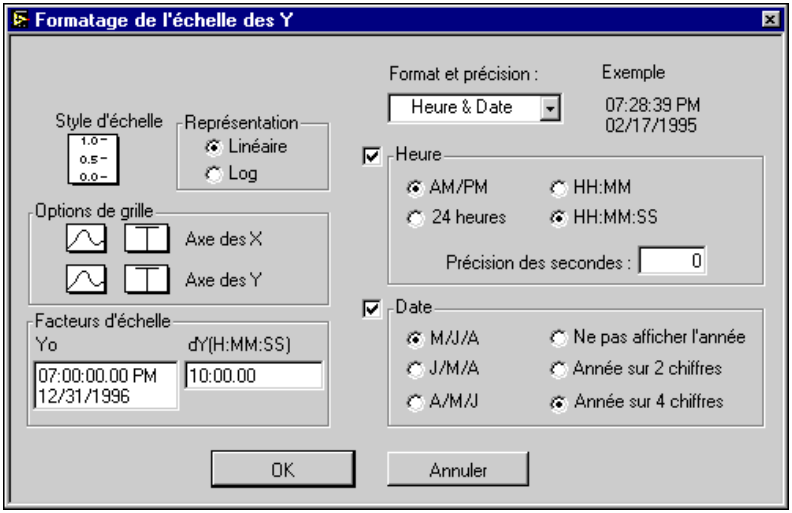
Remarque ***L'ordre des entrées de la fonction Assembler détermine l'ordre des tracés sur le graphe déroulant. Par exemple, si vous câblez les données brutes à l'entrée supérieure de la fonction Assembler et la moyenne à l'entrée inférieure, le premier tracé correspond aux données brutes et le second, à la moyenne.***

4. Exécutez le VI à partir de la face-avant. Le VI affiche deux tracés sur le graphe déroulant. Les tracés sont superposés, c'est-à-dire qu'ils partagent la même échelle verticale.
5. Exécutez le VI à partir du diagramme avec l'exécution en mode animation activée pour visualiser les données dans les registres à décalage.
6. Désactivez l'exécution en mode Animation. Exécutez le VI dans la face-avant. Lorsque le VI est en cours d'exécution, utilisez les boutons de la palette pour modifier le graphe déroulant. Vous pouvez réinitialiser le graphe déroulant, mettre à l'échelle les axes des X ou des Y, et changer le format d'affichage à tout instant. Vous pouvez également faire défiler un graphe pour voir d'autres zones ou faire un zoom sur les zones d'un graphe ou d'un graphe déroulant.

Vous pouvez utiliser les boutons **X** et **Y** pour remettre à l'échelle les axes des X et Y, respectivement. Si vous souhaitez que le graphe mette automatiquement à l'échelle l'une des échelles en continu, cliquez sur le verrou à gauche de chaque bouton pour bloquer la mise à l'échelle automatique.

Vous pouvez utiliser les autres boutons pour modifier la précision du texte des axes ou pour contrôler le mode de fonctionnement du graphe déroulant. Faites des essais avec ces boutons pour explorer leur fonction, faites défiler la zone affichée, ou faites un zoom sur des zones du graphe déroulant.

7. Formatez les échelles du graphe déroulant pour représenter un temps absolu ou relatif. Pour sélectionner le format de l'heure de l'échelle des X, ouvrez le menu local sur l'échelle des X et sélectionnez **Formatage…**
   a. Choisissez le temps absolu en sélectionnant l'option **Heure & Date** dans le menu déroulant **Format et précision**. Ceci change la boîte de dialogue en la boîte affichée ci-après. Pour que le graphe déroulant commence à un temps donné et qu'il s'incrémente à certains intervalles, vous pouvez modifier les valeurs de *Xo* et de *dX*, respectivement.

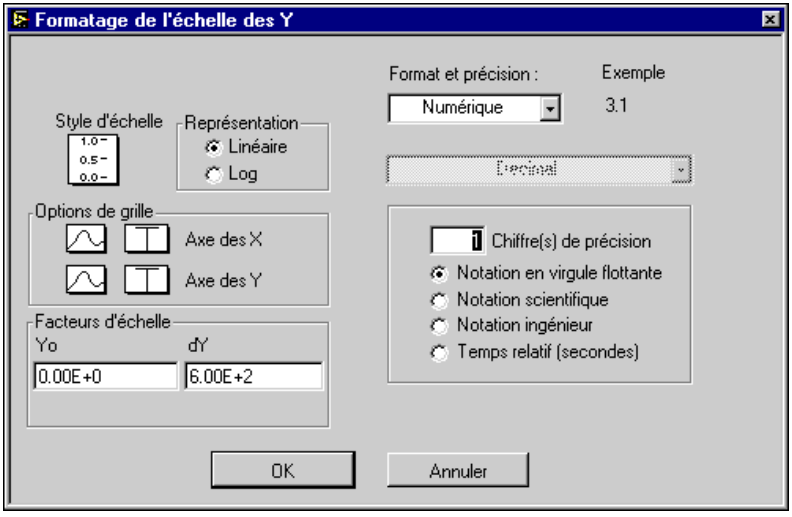


b. Formatez le graphe déroulant pour afficher les données en commençant à midi, le 24 octobre 1996, puis incrémentez toutes les 10 minutes, comme indiqué ci-dessus.

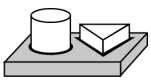
Remarque ***La modification du format du texte des axes nécessite souvent davantage d'espace physique que ce qui est originellement prévu pour l'axe. Si vous changez l'axe, le texte peut dépasser la taille maximale que présente la forme d'onde. Dans ce cas, vous pouvez utiliser le curseur de redimensionnement pour diminuer la zone d'affichage du graphe déroulant.***

8. Pour sélectionner le format du temps relatif, sélectionnez **Numérique** dans le menu déroulant **Format et précision**. Vous pouvez alors sélectionner l'option **Temps relatif (secondes)** dans la boîte de dialogue et représenter le temps en secondes. Modifiez la boîte

de dialogue, comme indiqué dans l'illustration suivante, et sélectionnez **OK**.



9. Exécutez le VI.
10. Enregistrez le VI sous le nom `Tracés aléatoires multiples.vi` dans le répertoire `LabVIEW\Activity`.

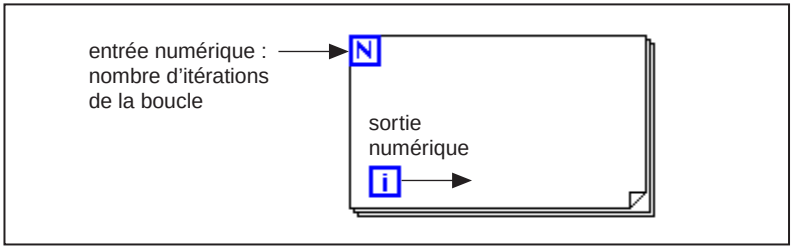


Fin de l'exercice 3-6.

Boucles For

Une boucle For exécute une section de code un nombre défini de fois. Sa taille est réglable, et comme la boucle While, elle ne se pose pas sur le diagramme immédiatement. A la place, une petite icône représentant la boucle For apparaît dans le diagramme, et vous pouvez ainsi ajuster ses dimensions et la placer où vous voulez. Pour cela, cliquez d'abord dans une zone au-dessus et à gauche de tous les terminaux. Tout en maintenant appuyé le bouton de la souris, faites glisser un rectangle incluant les terminaux que vous souhaitez placer dans la boucle For. Lorsque vous

relâchez le bouton de souris, le G crée une boucle For possédant la taille et la position sélectionnées. Placez la boucle For sur le diagramme en la sélectionnant dans **Fonctions»Structures**.



La boucle For exécute le diagramme dans sa bordure, un nombre prédéfini de fois. La boucle For possède deux terminaux, expliqués ci-dessous.

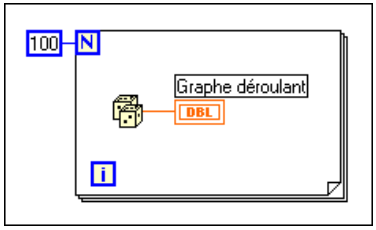
N Terminal de comptage (un terminal d'entrée) : le terminal de comptage spécifie le nombre de fois qu'il faut exécuter la boucle.

i Terminal d'itération (un terminal de sortie) : le terminal d'itération contient le nombre de fois que la boucle a été exécutée.

La boucle For est équivalente au pseudo-code suivant :

```
Pour i = 0 à N-1
Exécuter le diagramme à l'intérieur de la boucle
```

L'illustration suivante montre une boucle For générant 100 nombres aléatoires et affichant les points sur un graphe déroulant.

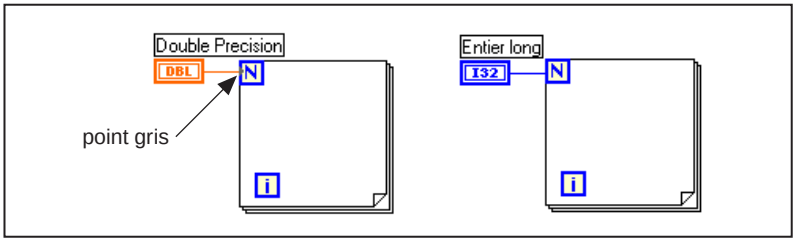


Conversion numérique

Jusqu'à présent, toutes les commandes et tous les indicateurs numériques que vous avez utilisés étaient des nombres flottants en double précision, représentés avec 32 bits. Le G peut cependant représenter des numériques comme des entiers (octets, mots ou entiers longs) ou des nombres flottants (simple précision, double précision ou étendus). La représentation par défaut d'un numérique est un nombre flottant en double précision.

Si vous câblez deux terminaux ayant différents types de données, le G convertit l'un des terminaux en la même représentation que l'autre terminal. En tant que rappel, le G place un point gris, appelé *point de coercition*, sur le terminal où a lieu la conversion.

Par exemple, considérez le terminal de comptage de la boucle For. La représentation du terminal est un entier long. Si vous câblez un nombre flottant en double précision au terminal de comptage, le G convertit le nombre en entier long. Notez le point gris dans le terminal de comptage de la première boucle For.



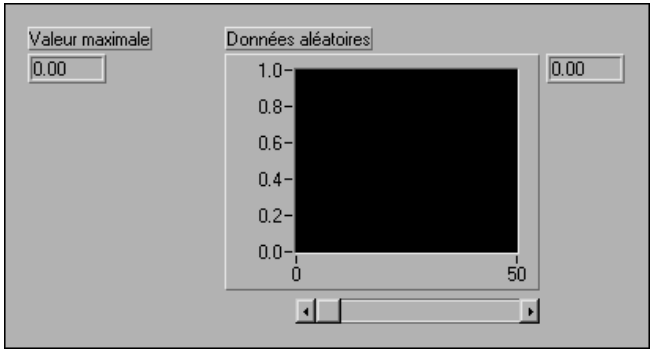
Remarque ***Lorsque le VI convertit les nombres flottants en entiers, il les arrondit à l'entier le plus proche. Si un nombre est exactement à mi-chemin entre deux entiers, il est arrondi à l'entier pair le plus proche. Par exemple, le VI arrondit 6,5 à 6 mais arrondit 7,5 à 8. Ceci est une méthode IEEE standard pour arrondir des nombres. Consultez le standard 754 IEEE pour plus de détails.***

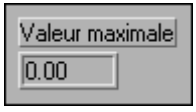
Exercice 3-7. Utiliser une boucle For

Votre objectif est d'utiliser une boucle For et des registres à décalage pour calculer la valeur maximale dans une série de nombres aléatoires.

Face-avant

1. Ouvrez une nouvelle face-avant et ajoutez les objets comme indiqué dans l'illustration suivante.





a. Placez un indicateur numérique sur la face-avant et nommez-le `Valeur maximale`.

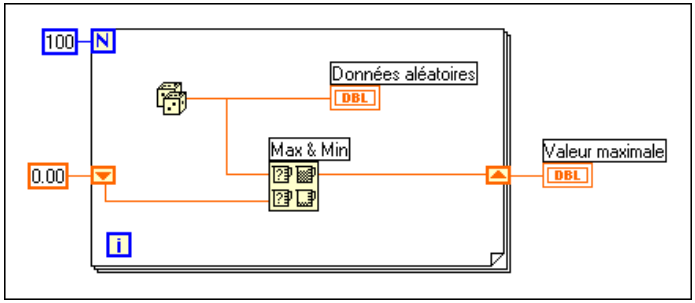
b. Placez un graphe déroulant sur la face-avant et nommez-le `Données aléatoires`. Changez l'échelle du graphe déroulant pour qu'elle s'inscrive entre `0,0` et `1,0`.

c. Ouvrez le menu local du graphe déroulant et choisissez **Visualiser»Barre de défilement** et **Visualiser»Afficheur numérique**. Ouvrez le menu local, puis masquez la palette et la légende.

d. Ajustez la taille de la barre de défilement à l'aide de l'outil Flèche.

Diagramme

2. Ouvrez le diagramme et modifiez-le comme indiqué dans l'illustration suivante.



3. Placez une boucle For (**Fonctions»Structures**) sur le diagramme.
4. Ajoutez un registre à décalage en ouvrant le menu local ou en cliquant avec le bouton droit de la souris sur la bordure droite ou gauche de la boucle For, puis en choisissant **Ajouter registre à décalage**.
5. Ajoutez les objets suivants sur le diagramme.

Fonction "Nombre aléatoire (0–1)" (**Fonctions»Numérique**) : cette fonction génère des données aléatoires.

Constante numérique (**Fonctions»Numérique**) : la boucle For a besoin de connaître le nombre d'itérations à réaliser. Dans ce cas, vous exécuterez 100 fois le diagramme situé à l'intérieur de la boucle For.

Constante numérique (**Fonctions»Numérique**) : mettez la valeur initiale du registre à décalage à zéro pour cet exercice car vous savez que la sortie du générateur des nombres aléatoires se trouve entre 0,0 et 1,0.

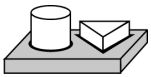
Vous devez prévoir les données que vous obtiendrez pour initialiser un registre à décalage. Par exemple, si vous initialisez le registre à décalage à 1, cette valeur est déjà supérieure à toutes les valeurs de données attendues, et va toujours être la valeur maximale. Si vous n'initialisez pas le registre à décalage, il contiendra la valeur maximale d'une exécution précédente du VI. Vous risquez ainsi d'obtenir une valeur de sortie maximale non liée à l'ensemble actuel des données recueillies.

Fonction Max & Min (**Fonctions»Comparaison**) : accepte deux entrées numériques et génère la valeur maximale des deux entrées dans l'angle supérieur droit et la valeur minimale dans l'angle inférieur droit. Comme vous n'êtes intéressé que par la valeur maximale dans cet exercice, câblez seulement la sortie maximale et ignorez la sortie minimale.

6. Câblez les terminaux comme indiqué. Si le terminal de la valeur maximale était dans la boucle For, vous devriez le voir continuellement mis à jour, mais comme il se trouve en dehors de la boucle, il ne contiendra que la dernière valeur maximale calculée.

Remarque ***Comme la mise à jour des indicateurs à chaque itération d'une boucle prend du temps, vous devez essayer de l'éviter lorsque c'est possible afin d'augmenter la vitesse d'exécution.***

7. Exécutez le VI.
8. Enregistrez le VI sous le nom `Calculer Max.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l'exercice 3-7.

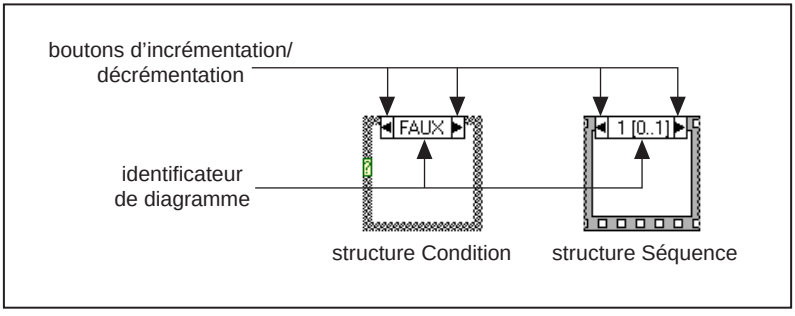
4

Structure Condition, structure Séquence et boîte de calcul

Ce chapitre introduit les concepts de base des structures Condition, Séquence et de la boîte de calcul. Les exercices présentés dans ce chapitre détaillent les points suivants :

- Comment utiliser la structure Condition
- Comment utiliser la structure Séquence
- Définition des variables locales de séquence et comment les utiliser
- Définition d'une boîte de calcul et comment l'utiliser

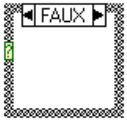
Les structures Condition et Séquence peuvent toutes deux posséder plusieurs sous-diagrammes, configurés comme un jeu de cartes, c'est-à-dire qu'un seul diagramme est visible à la fois. Au-dessus de chaque bordure de structure, se trouve la *fenêtre d'affichage du sous-diagramme*, qui contient un *identificateur de diagramme* au centre, ainsi que des boutons de décrémentation et d'incrémentation de chaque côté. L'identificateur de diagramme indique le type de sous-diagramme couramment affiché. Pour les structures Condition, un identificateur de diagramme est une liste de valeurs qui permettent de sélectionner le sous-diagramme. Pour les structures Séquence, l'identificateur de diagramme est le nombre de cadres dans la séquence (de 0 à $n - 1$). L'illustration suivante montre une structure Condition et une structure Séquence.



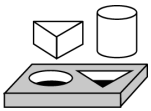
Si vous cliquez sur le bouton de décrémentation (à gauche) ou d'incrémentation (à droite), vous affichez, respectivement, le sous-diagramme précédent ou le sous-diagramme suivant. Une incrémentation à partir du dernier sous-diagramme affiche le premier sous-diagramme, tandis qu'une décrémentation à partir du premier sous-diagramme affiche le dernier sous-diagramme. Pour plus d'informations sur les structures Condition et Séquence, reportez-vous au chapitre 19, *Structures*, dans le *Manuel de référence de programmation en G*.

Structure Condition

La structure Condition possède au moins deux sous-diagrammes, ou *conditions*. Une seule de ces conditions est effectuée lorsque la structure s'exécute. Ceci dépend de l'entier, du booléen, de la chaîne de caractères, ou de la valeur d'énumération que vous câblez au côté externe du terminal de sélection ou *sélecteur*. Une structure Condition est représentée dans l'illustration suivante.



☞ **Remarque** ***Les déclarations de conditions dans les autres langages de programmation n'effectuent généralement pas de condition si la valeur sur laquelle porte la condition dépasse la gamme. En G, vous devez soit inclure une condition par défaut qui gère les valeurs hors gamme, soit répertorier clairement chaque valeur d'entrée possible.***

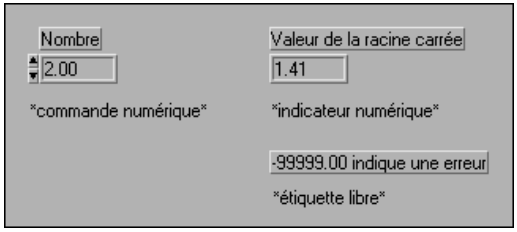


Exercice 4-1. Utiliser une structure Condition

Votre objectif est de construire un VI qui vérifie un nombre pour voir s'il est positif. Si le nombre est positif, le VI calcule sa racine carrée ; sinon, le VI retourne une erreur.

Face-avant

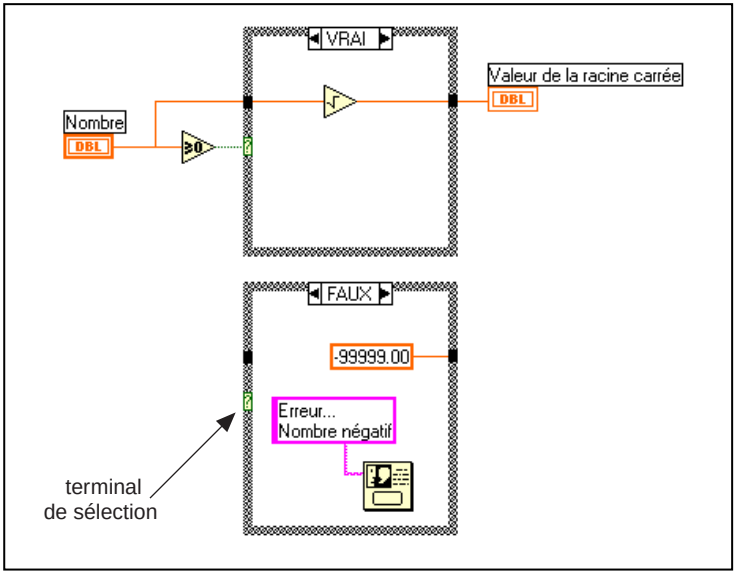
1. Ouvrez une nouvelle face-avant et créez les objets comme indiqué dans l'illustration ci-après.



La commande `Nombre` fournit le nombre. L'indicateur `Valeur de la racine carrée` affiche la racine carrée du nombre. Quant au texte libre, il peut être utile à l'utilisateur pour y insérer des commentaires.

Diagramme

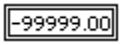
2. Construisez le diagramme comme représenté dans l'illustration suivante.



3. Placez une structure Condition dans le diagramme en la sélectionnant à partir de la palette **Fonctions»Structures**. La structure Condition est une boîte aux dimensions réglables qui ne se place pas immédiatement sur le diagramme. Vous pouvez ainsi la placer et modifier ses dimensions à votre guise. Pour cela, cliquez dans une zone au-dessus et à gauche de tous les terminaux que vous voulez mettre à l'intérieur de la structure Condition. Maintenez enfoncé le bouton de la souris et faites glisser un rectangle incluant les terminaux souhaités.

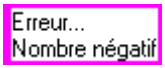
Fonction "Supérieur ou égal à 0 ?" (**Fonctions»Comparaison**) : retourne Vrai (TRUE) si le nombre entré est supérieur ou égal à 0.

Fonction "Racine carrée" (**Fonctions»Numérique**) : retourne la racine carrée du nombre entré.



Constante numérique (**Fonctions»Numérique**) : dans cet exercice, la constante indique la valeur numérique de l'erreur.

Fonction "Boîte de dialogue 1 bouton" (**Fonctions»Temps & dialogue**) : dans cet exercice, la fonction affiche une boîte de dialogue contenant le message `Erreur...Nombre négatif`.



Constante chaîne de caractères (**Fonctions»Chaîne de caractères**) : entrez du texte dans la boîte grâce à l'outil Texte.

Le VI exécute la condition VRAI ou la condition FAUX. Si le nombre est supérieur ou égal à zéro, le VI exécute la condition VRAI et renvoie la racine carrée du nombre. La condition FAUX génère –99999,00 et affiche une boîte de dialogue dotée du message `Erreur...Nombre négatif`.

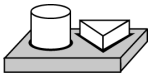
Remarque ***Vous devez définir le tunnel de sortie pour chaque condition. Lorsque vous créez un tel tunnel dans une condition, les tunnels apparaissent à la même position dans toutes les autres conditions. Les tunnels non câblés apparaissent sous la forme de carrés blancs.***

4. Revenez à la face-avant et exécutez le VI. Essayez un nombre supérieur ou égal à zéro et un nombre inférieur ou égal à zéro, en modifiant la valeur de la commande numérique que vous avez libellée `Nombre`. Remarquez que lorsque vous entrez un nombre négatif dans la commande numérique, LabVIEW affiche le message d'erreur que vous avez défini dans la condition FAUX de la structure Condition.
5. Enregistrez le VI sous le nom `Racine carrée.vi` dans le répertoire `LabVIEW\Activity`.

Logique d'un VI

Le diagramme affiché dans cet exercice a le même effet que le pseudo-code suivant dans un langage textuel :

```
Si (Nombre >= 0) alors
Valeur racine carrée = SQRT(Nombre)
Sinon
Valeur racine carrée = -99999.00
Afficher le message "Erreur...nombre négatif"
Fin de Si
```

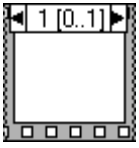


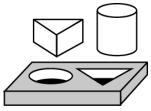
Fin de l'exercice 4-1.

Structure Séquence

La structure Séquence, qui ressemble à des cadres d'image, exécute des sous-diagrammes les uns après les autres. Dans les langages de programmation conventionnels, les déclarations du programme s'exécutent dans l'ordre dans lequel elles apparaissent. Dans la programmation en flux de données, un nœud s'exécute lorsque les données sont disponibles à toutes les entrées de nœud. Il est cependant parfois nécessaire d'exécuter un nœud avant un autre. Le G utilise la structure Séquence comme une méthode pour contrôler l'ordre dans lequel les nœuds s'exécutent. Le G exécute d'abord le diagramme situé à l'intérieur de la bordure du Cadre 0, puis il exécute le diagramme à l'intérieur de la bordure du Cadre 1, et ainsi de suite. Tout comme avec la structure Condition, un seul cadre est visible à la fois.

Une structure Séquence est représentée dans l'illustration suivante.



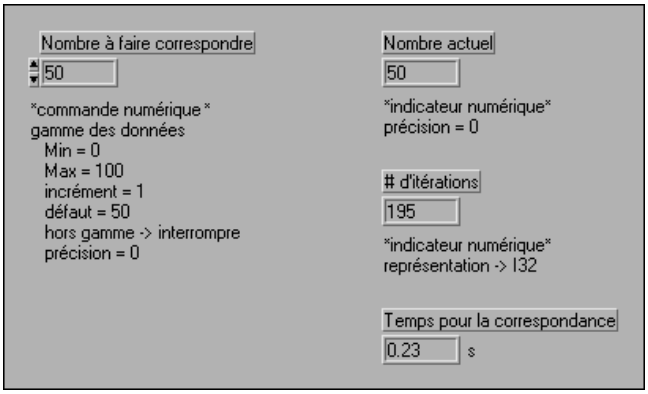


Exercice 4-2. Utiliser une structure Séquence

Votre objectif est de construire un VI qui calcule le temps nécessaire pour générer un nombre aléatoire correspondant à un nombre donné.

Face-avant

1. Ouvrez une nouvelle face-avant et construisez celle qui est représentée dans l'illustration suivante. Assurez-vous de modifier les commandes et les indicateurs comme il décrit dans le texte de l'illustration suivante.



La commande `Nombre à trouver` contient le nombre que vous souhaitez faire correspondre. L'indicateur `Nombre actuel` affiche le nombre aléatoire courant. L'indicateur `# d'itérations` affiche le nombre d'itérations avant l'obtention d'une correspondance. L'indicateur `Temps pour la correspondance` indique le nombre de secondes qui se sont écoulées avant l'obtention du nombre correspondant.

Modification du format numérique

LabVIEW affiche par défaut des valeurs dans les commandes numériques en notation décimale, avec deux chiffres après la virgule (par exemple, 3,14). Vous pouvez utiliser l'option **Format & Précision…** du menu local d'une commande ou d'un indicateur pour changer la précision ou pour afficher les commandes et indicateurs numériques en notation scientifique ou ingénieur. Vous pouvez également utiliser l'option **Format & Précision…** pour changer les formats d'heure et de date des commandes numériques.

2. Ouvrez le menu local de l'indicateur numérique Temps écoulé et choisissez **Format & Précision…** La face-avant doit être la fenêtre active pour que vous puissiez accéder au menu.

3. Entrez `3` dans la case “Chiffre(s) de précision” et cliquez sur **OK**.

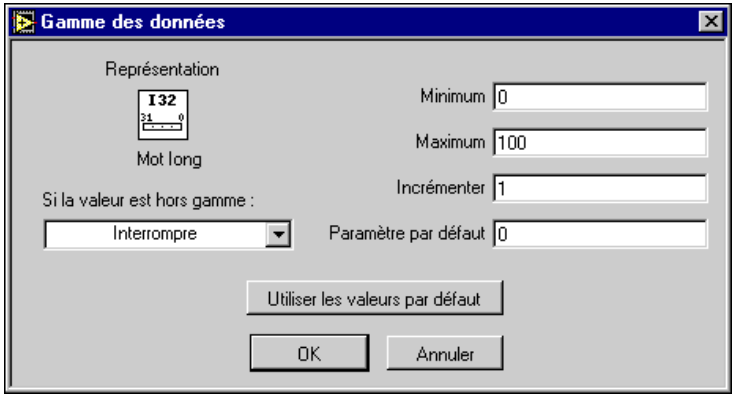
4. Ouvrez le menu local du nombre pour la commande `Nombre à faire correspondondre` et choisissez **Représentation»I32**.
5. Répétez l’étape 4 pour le nombre actuel et le nombre d’itérations des indicateurs numériques.

Définition de la gamme des données

Avec l’option Gamme des données…, vous pouvez empêcher un utilisateur de définir la valeur d’une commande ou d’un indicateur en dehors d’une gamme ou d’un incrément prédéfini. Vous avez trois options : ignorer la valeur, la forcer à rentrer dans la gamme, ou suspendre l’exécution. Le symbole de dépassement de gamme apparaît à la place du bouton d’exécution dans la barre d’outils lorsqu’une erreur de dépassement fait suspendre l’exécution. De plus, une bordure sombre et unie encadre la commande qui se trouve hors gamme.

6. Ouvrez le menu local de l’indicateur Nombre à trouver et choisissez **Gamme des données…**

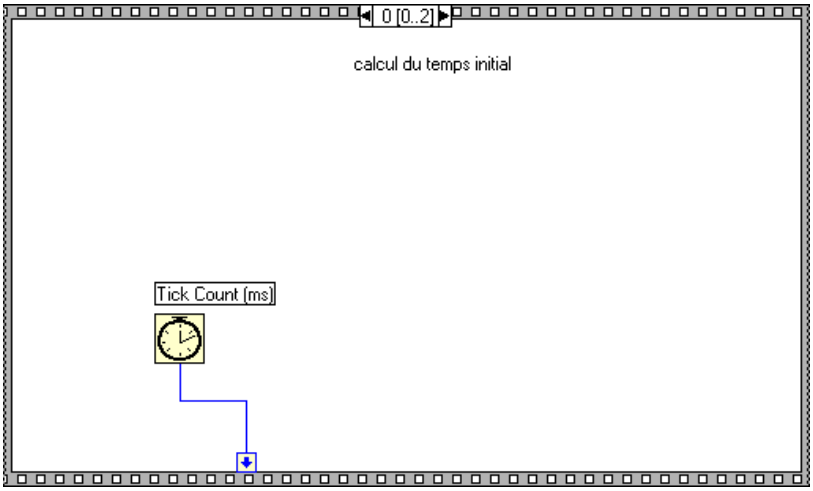
7. Remplissez la boîte de dialogue comme présenté dans l'illustration suivante et cliquez sur **OK**.

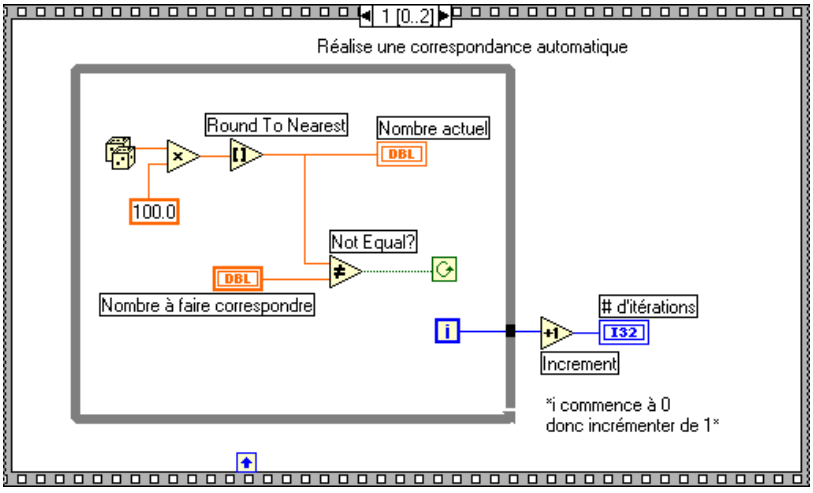


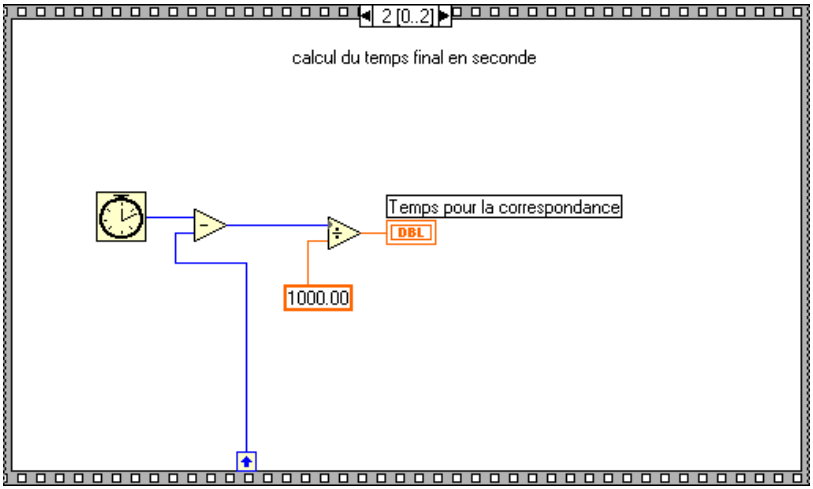
Diagramme

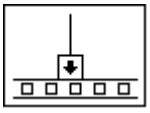
8. Ouvrez le diagramme.
9. Placez la structure Séquence (**Fonctions»Structures**) dans le diagramme.
10. Agrandissez la structure en faisant glisser un coin avec le curseur de redimensionnement.
11. Créez un nouveau cadre en ouvrant le menu local à partir de la bordure du cadre et choisissez **Ajouter une étape après**. Répétez cette étape pour créer le cadre 2.

12. Construisez le diagramme affiché dans les illustrations suivantes.









Le cadre 0 de l'illustration précédente contient une petite boîte avec une flèche à l'intérieur. Cette boîte est une *variable locale de séquence* qui transmet des données entre les cadres d'une structure Séquence. Vous pouvez créer des variables locales de séquence à partir de la bordure d'un cadre. Les données câblées à une variable locale de séquence de cadre sont alors disponibles dans les nouveaux cadres. Vous ne pouvez cependant pas accéder aux données dans les cadres précédant le cadre dans lequel vous avez créé la variable locale de séquence.

13. Créez la variable locale de séquence en ouvrant le menu local sur la bordure inférieure du cadre 0 et en choisissant **Ajouter une variable locale de séquence**.

La variable locale de séquence apparaît sous la forme d'un carré vide. La flèche à l'intérieur du carré apparaît automatiquement lorsque vous lui câblez une sortie ou une entrée de fonction.

14. Terminez le diagramme comme indiqué dans la première illustration de la section *Diagramme* de cet exercice.

Fonction "Compteur d'impulsions d'horloge (ms)" (**Fonctions»Temps & dialogue**) : retourne le nombre de millisecondes écoulées depuis la mise en marche. Pour cet exercice, vous avez besoin de deux fonctions "Compteur d'impulsions d'horloge".

Fonction "Nombre aléatoire (0–1)" (**Fonctions»Numérique**) : retourne un nombre aléatoire entre 0 et 1.

Fonction Multiplier (**Fonctions»Numérique**) : dans cet exercice, la fonction multiplie le nombre aléatoire par 100.

Constante numérique (**Fonctions»Numérique**) : dans cet exercice, la constante numérique représente le nombre maximum pouvant être multiplié.

Fonction “Arrondir à l’entier le plus proche” (**Fonctions»Numérique**) : dans cet exercice, la fonction arrondit le nombre aléatoire compris entre 0 et 100 à l’entier le plus proche.

Fonction “Non égaux ?” (**Fonctions»Comparaison**) : dans cet exercice, la fonction compare le nombre aléatoire au nombre spécifié dans la face-avant et retourne Vrai (TRUE) si les nombres ne sont pas égaux. Sinon, cette fonction retourne Faux (FALSE).

Fonction Incrément (**Fonctions»Numérique**) : dans cet exercice, la fonction incrémente de 1 le terminal d’itération de la boucle While.

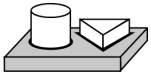
Fonction Soustraire (**Fonctions»Numérique**) : dans cet exercice, la fonction retourne le temps (en millisecondes) écoulé entre l’étape 2 et l’étape 0.

Fonction Diviser (**Fonctions»Numérique**) : dans cet exercice, la fonction divise le nombre de millisecondes écoulées par 1000 pour convertir le nombre en secondes.

Constante numérique (**Fonctions»Numérique**) : nombre utilisé pour la conversion du temps écoulé en secondes.

Dans l’étape 0, la fonction “Compteur d’impulsions d’horloge (ms)” retourne le temps actuel en millisecondes. Cette valeur est câblée à la variable locale de séquence : cette valeur est donc disponible dans les étapes suivantes. Dans l’étape 1, le VI exécute la boucle While tant que le nombre spécifié ne correspond pas au nombre retourné par la fonction “Nombre aléatoire (0–1)”. Dans l’étape 2, la fonction “Compteur d’impulsions d’horloge (ms)” retourne un nouveau temps en millisecondes. Le VI soustrait l’ancien temps (de l’étape 0 à la variable locale de séquence) au nouveau temps pour calculer le temps écoulé.

15. Revenez à la face-avant et entrez un nombre dans la commande `Nombre à trouver`, puis exécutez le VI.
16. Enregistrez le VI sous le nom `Temps écoulé.vi` dans le répertoire `LabVIEW\Activity`.



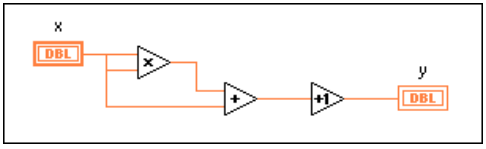
Fin de l'exercice 4-2.

Boîte de calcul

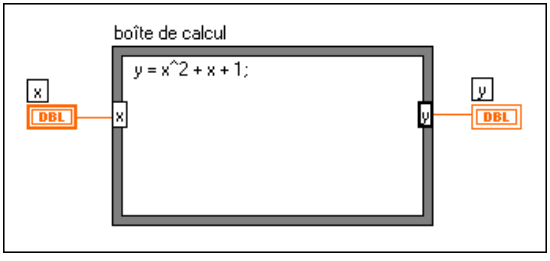
Vous pouvez régler la taille d'une boîte de calcul. Cette boîte permet d'entrer des formules directement dans un diagramme. Placez la boîte de calcul sur le diagramme en la sélectionnant dans **Fonctions»Structures**. Cette fonction est utile lorsqu'une équation possède de nombreuses variables ou s'avère complexe. Par exemple, considérez l'équation ci-dessous :

$y = x^2 + x + 1$

Si vous implémentez cette équation avec des fonctions arithmétiques en G classiques, le diagramme ressemble à celui de l'illustration suivante.



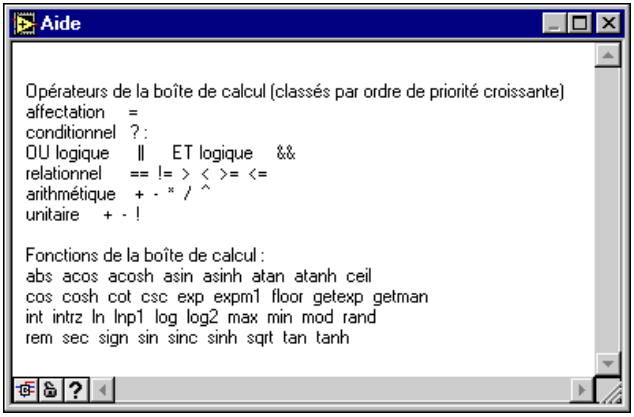
Vous pouvez implémenter la même équation avec une boîte de calcul, comme présenté dans l'illustration suivante :



Grâce aux boîtes de calcul, vous pouvez entrer directement une ou plusieurs formules compliquées, au lieu de créer des sous-sections du diagramme. Entrez des formules en utilisant l'outil Texte. Pour créer des terminaux

d'entrée et de sortie de la boîte de calcul, ouvrez le menu local sur la bordure du nœud et choisissez Ajouter une entrée (Ajouter une sortie). Entrez le nom de la variable dans la boîte. Les variables font la différence entre majuscules et minuscules. Entrez la ou les formules dans la boîte. Chaque déclaration de formule doit se terminer par un point-virgule (;).

Les opérateurs et les fonctions disponibles dans la boîte de calcul sont répertoriés dans la fenêtre d'aide de la boîte de calcul, comme présenté dans l'illustration suivante.

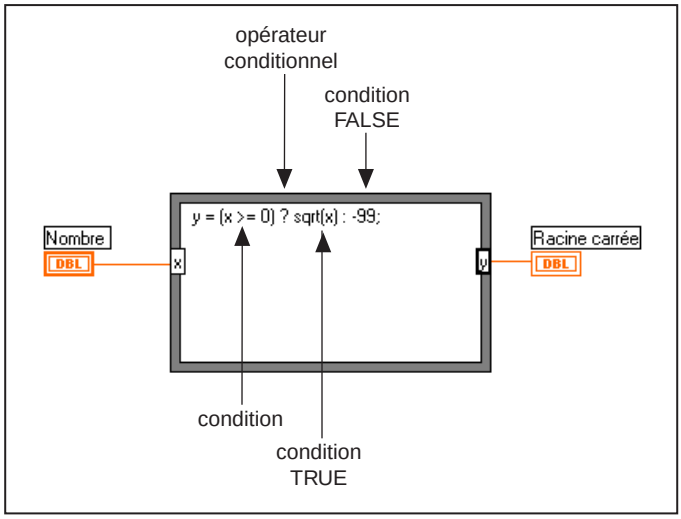


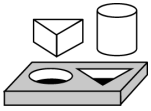
L'exemple suivant montre la façon dont vous pouvez créer un élément relié conditionnel dans une boîte de calcul.

Considérez le fragment de code suivant qui calcule la racine carrée de x si x est positif, puis affectez le résultat à y. Si x est négatif, le code affecte -99 à y.

```
Si (x >= 0) alors
y = sqrt(x)
Sinon
y = -99
Fin de Si
```

Vous pouvez implémenter le fragment de code dans une boîte de calcul, comme montré dans l'illustration suivante.





Exercice 4-3. Utiliser une boîte de calcul

Votre objectif est de construire un VI qui utilise une boîte de calcul pour calculer les équations suivantes.

$y1 = x^3 - x^2 + 5$

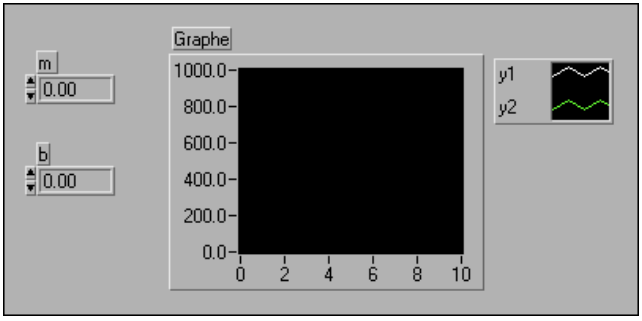
$y2 = m * x + b$

où x va de 0 à 10.

Vous n'utiliserez qu'une boîte de calcul pour les deux équations, et vous afficherez les résultats dans le même graphe. Pour plus d'informations sur les graphes, consultez le chapitre 5, *Tableaux, clusters et graphes*.

Face-avant

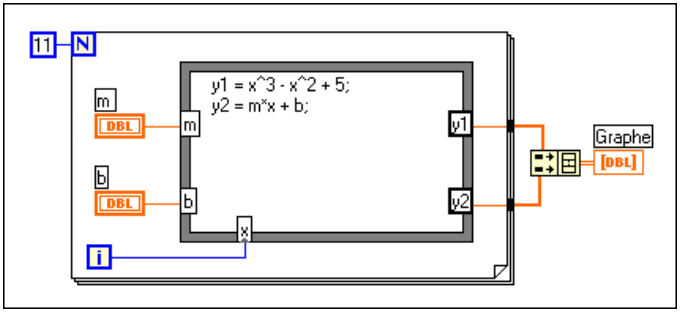
1. Ouvrez une nouvelle face-avant et construisez la face-avant représentée dans l'illustration suivante. L'indicateur du graphe affiche la courbe de l'équation. Le VI utilise les deux commandes numériques pour entrer les valeurs de `m` et `b`.

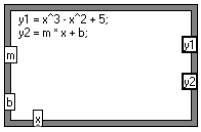


2. Créez la légende du graphe comme présenté dans l'illustration suivante en sélectionnant **Visualiser»Légende**. Utilisez le curseur de redimensionnement pour étendre la légende vers le bas afin d'afficher les deux tracés. Utilisez l'outil Texte pour renommer les tracés. Vous pouvez définir le style de la ligne de chaque tracé dans le menu local de la légende. Vous pouvez également changer la couleur de chaque tracé en utilisant l'outil Pinceau sur la légende des tracés.

Diagramme

3. Construisez le diagramme représenté dans l'illustration suivante.





Boîte de calcul (**Fonctions»Structures**). Créez les trois terminaux d'entrée en ouvrant le menu local sur la bordure et en choisissant **Ajouter une entrée**. Pour créer le terminal de sortie, choisissez **Ajouter une sortie** dans le menu local.

Lorsque vous créez un terminal d'entrée ou de sortie, vous devez lui donner un nom de variable. Le nom de variable doit correspondre exactement à la variable utilisée dans la formule. Le système fait la différence entre les majuscules et les minuscules. Ainsi, si vous utilisez la lettre minuscule a pour nommer le terminal, vous devez utiliser la même minuscule a dans la formule. Vous pouvez entrer les noms de variables et les formules à l'aide de l'outil Texte.

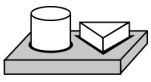
Remarque ***Bien que les noms de variables ne soient pas limités en longueur, soyez conscient que des noms longs occupent un espace considérable dans le diagramme. Un point-virgule (;) doit terminer la déclaration des formules.***

Constante numérique (**Fonctions»Numérique**). Vous pouvez également ouvrir le menu local sur le terminal de comptage, puis sélectionnez **Créer une constante** pour créer et câbler automatiquement la constante numérique. La constante numérique spécifie le nombre d'itérations que doit effectuer la boucle For. Si x s'inscrit entre 0 et 10 (10 inclus), vous devez câbler 11 au terminal de comptage.

Comme le terminal d'itération compte de 0 à 10, utilisez-le pour contrôler la valeur x dans la boîte de calcul.

La fonction "Construire un tableau" (**Fonctions»Tableau**) met deux entrées de type tableau sous la forme d'un tableau à deux dimensions pour un affichage dans un graphe multicourbes. Créez les deux terminaux d'entrée avec le curseur de redimensionnement en faisant glisser l'un des coins. Pour plus d'informations sur les tableaux, consultez le chapitre 5, *Tableaux, clusters et graphes*.

4. Revenez sur la face-avant et exécutez le VI avec des valeurs différentes pour `m` et `b`.
5. Enregistrez le VI sous le nom `Equations.vi` dans le répertoire `LabVIEW/Activity`.



Fin de l'exercice 4-3.

Dépendance artificielle des données

Les nœuds non connectés par un fil de liaison peuvent s'exécuter dans n'importe quel ordre. Les nœuds ne s'exécutent pas nécessairement de gauche à droite ou de haut en bas. Une structure Séquence représente une façon de contrôler l'ordre d'exécution lorsqu'il n'existe pas une dépendance naturelle des données.

Une autre façon de contrôler l'ordre d'exécution consiste à créer une dépendance artificielle des données, auquel cas l'exécution d'un objet est amorcée par l'arrivée des données plutôt que par leur valeur. Le récepteur n'utilise peut-être pas réellement les données de façon interne. L'avantage de la dépendance artificielle est que tous les nœuds sont visibles à un niveau (cependant, dans certaines conditions, la confusion créée par des liens artificiels entre les nœuds peut s'avérer gênante).

Vous pouvez ouvrir le VI "Modèle de séquencement" [`Timing Template (data dep).vi`] dans `Examples\General\structs.llb` pour voir la façon dont le `modèle de séquencement` a été modifié, afin d'utiliser la dépendance artificielle des données à la place d'une structure Séquence.

5

Tableaux, clusters et graphes

Ce chapitre introduit les concepts de base concernant le polymorphisme, les tableaux, les clusters et les graphes. Il présente également des exercices expliquant l'auto-indexation ainsi que les VIs de graphe et d'analyse.

Tableaux

Un tableau regroupe les éléments de données de même type. Un tableau possède une ou plusieurs dimensions, avec jusqu'à $2^3 - 1$ éléments par dimension (si la mémoire le permet). Vous accédez à chaque élément du tableau grâce à son indice. L'indice s'inscrit entre 0 et $n - 1$, où n représente le nombre d'éléments dans le tableau. Le tableau 1D de valeurs numériques suivant illustre cette structure. Notez que le premier élément possède l'indice 0, le second élément possède l'indice 1, et ainsi de suite.

indice	0	1	2	3	4	5	6	7	8	9
tableau de 10 éléments	1.2	3.2	8.2	8.0	4.8	5.1	6.0	1.0	2.5	1.7

Comment créer et initialiser des tableaux ?

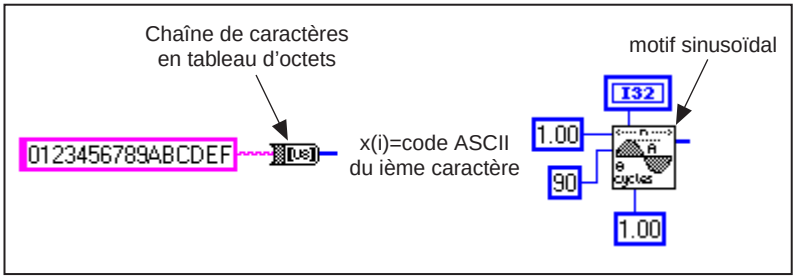
Si vous avez besoin d'un tableau comme source de données dans votre diagramme, vous pouvez choisir la constante **Tableau** dans la palette **Fonctions**, puis sélectionner et placer le tableau vierge dans votre diagramme. Avec l'outil Doigt, vous pouvez choisir une constante numérique, une constante booléenne, ou une constante chaîne de caractères à placer dans le tableau vide. L'illustration suivante montre un exemple de tableau vierge avec une constante numérique insérée dans le tableau.

Pour créer un tableau sur la face-avant, sélectionnez **Tableaux et Clusters** dans la palette **Commandes**, puis placez le tableau vierge sur votre face-avant. Sélectionnez ensuite un objet (numérique, par exemple) et placez-le dans le tableau vierge. Ceci permet de créer un tableau de numériques.

Remarque ***Vous pouvez également créer un tableau et sa commande correspondante sur la face-avant, puis copier ou faire glisser la commande du tableau sur le diagramme afin de créer une constante du même type.***

Pour plus d'informations sur la façon de créer des commandes et des indicateurs de type tableau sur la face-avant, consultez le chapitre 14, *Commandes et indicateurs de tableaux et de clusters*, dans le *Manuel de référence de programmation en G*.

Il y a plusieurs façons de créer et d'initialiser des tableaux dans le diagramme. Certaines fonctions du diagramme produisent également des tableaux, comme le montre l'illustration suivante.



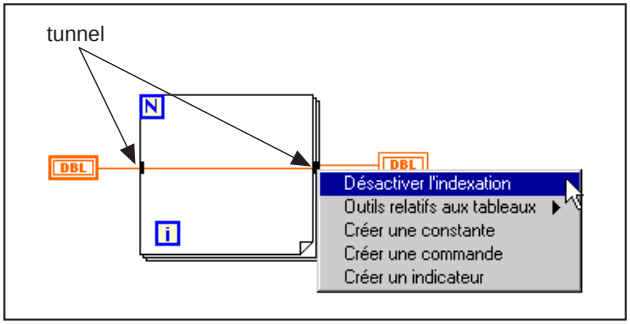
Commandes, constantes et indicateurs de tableau

Créez des commandes, constantes et indicateurs de tableau sur la face-avant ou le diagramme en combinant un tableau vierge avec un numérique, un booléen, une chaîne de caractères ou un cluster. Un élément de tableau ne peut pas être un autre tableau ou un graphe. Pour des exemples de tableaux, reportez-vous à `Examples\General\arrays.llb`.

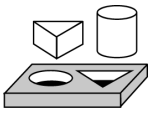
Auto-indexation

Les structures de boucle For et de boucle While peuvent indexer et accumuler automatiquement des tableaux. Ceci s'appelle *l'auto-indexation*. Lorsque vous activez l'auto-indexation et que vous câblez un tableau de dimension quelconque d'un nœud externe à un tunnel d'entrée sur la bordure de la boucle, les composantes de ce tableau entrent dans la boucle, les unes après les autres, en commençant par la première composante. La boucle indexe les éléments scalaires des tableaux 1D, les tableaux 1D des tableaux 2D, et ainsi de suite. L'action opposée se produit au niveau des tunnels de sortie : les éléments s'accumulent en séquence dans les tableaux 1D, tandis que les tableaux 1D s'accumulent dans les tableaux 2D, et ainsi de suite.

Remarque ***L'auto-indexation est une option par défaut pour chaque tableau câblé à une boucle For. Vous pouvez désactiver l'auto-indexation en ouvrant le menu local sur le tunnel (point d'entrée du tableau d'entrée) et en sélectionnant*** **Désactiver l'indexation.**



Par défaut, l'auto-indexation est désactivée pour chaque tableau câblé à une boucle While. Ouvrez le menu local du tunnel d'un tableau d'une boucle While pour activer l'auto-indexation.



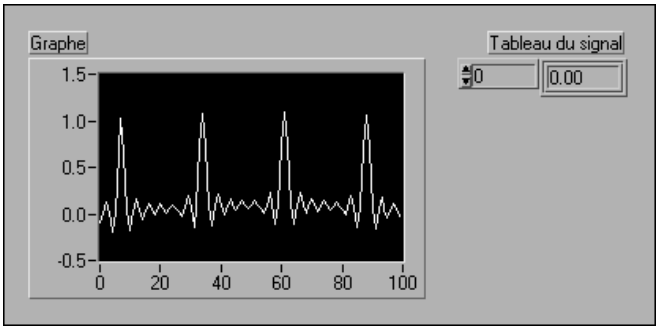
Exercice 5-1. Créer un tableau par auto-indexation

Votre objectif est de créer un tableau en utilisant la fonction d'auto-indexation d'une boucle For et de tracer ce tableau dans un graphe.

Vous allez construire un VI qui génère un tableau grâce au VI "Générer un signal" (Generate Waveform.vi). Ce nouveau VI trace le tableau dans un graphe. Vous allez également modifier le VI pour tracer plusieurs courbes.

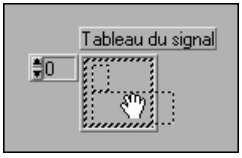
Face-avant

1. Ouvrez une nouvelle face-avant.



2. Placez un tableau vierge provenant de la sous-palette **Commandes» Tableaux et Clusters** dans la face-avant. Nommez-le `Tableau du signal`.

3. Placez un indicateur numérique provenant de la palette **Commandes» Numérique** à l'intérieur de l'emplacement du premier élément du tableau vierge, comme l'indique l'illustration suivante. Cet indicateur affiche le contenu du tableau.



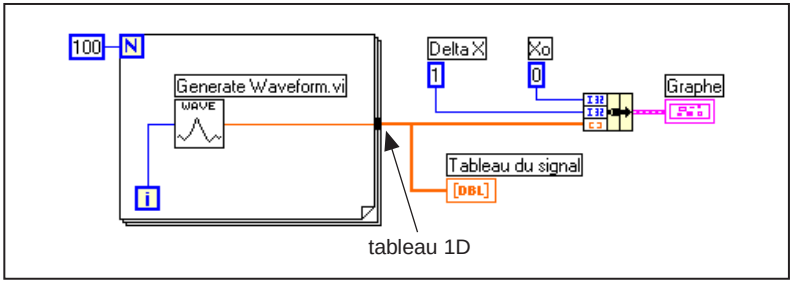
4. Placez un graphe provenant de la palette **Commandes»Graphe** dans la face-avant. Nommez-le `graphe`.

5. Agrandissez le graphe en faisant glisser l'un des coins avec le curseur de redimensionnement.
6. Masquez la légende et la palette.
7. Désactivez la mise à l'échelle automatique en ouvrant le menu local du graphe et en désélectionnant **Echelle des Y»Mise à l'échelle automatique des Y**.

8. Utilisez l'outil Texte pour remettre à l'échelle l'axe des Y afin qu'il s'inscrive entre `-0.5` et `1.5`.

Diagramme

9. Construisez le diagramme comme présenté dans l'illustration suivante.



VI "Générer un signal" (Generate Waveform.vi) (**Fonctions»Sélectionner un VI…** à partir du répertoire `LabVIEW\Activity`) : retourne un point d'un signal. Comme le VI requiert une entrée d'indice scalaire, câblez le terminal d'itération de boucle à cette entrée.

Remarquez que le fil de liaison de ce VI devient plus épais lorsqu'il se transforme en un tableau sur la bordure de la boucle.

La boucle For accumule automatiquement les tableaux à sa limite. Ceci s'appelle l'auto-indexation. Dans un tel cas, la constante numérique connectée à l'entrée numérique de comptage de boucle (N) force la boucle For à créer un tableau à 100 éléments (indexation de 0 à 99).

Fonction Assembler (**Fonctions»Cluster**) : assemble les composantes du tracé dans un cluster. Vous devez modifier la taille de l'icône de la fonction Assembler avant de pouvoir la câbler convenablement. Placez l'outil Flèche sur le coin inférieur gauche de l'icône. L'outil se transforme en curseur de redimensionnement (voir ci-contre). Lorsque l'outil se transforme, cliquez dessus et faites-le glisser vers le bas jusqu'à l'apparition d'un troisième terminal d'entrée. Vous pouvez désormais continuer à câbler votre diagramme comme montré dans l'illustration précédente.

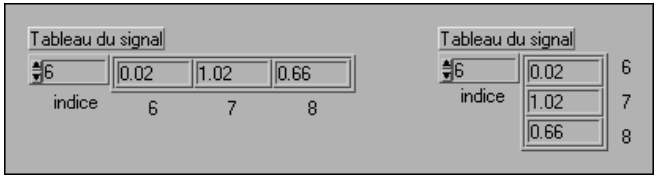
Constante numérique (**Fonctions»Numérique**) : trois constantes numériques définissent le nombre d'itérations de la boucle For, la valeur initiale de X et la valeur de delta X. Remarquez que vous pouvez ouvrir le menu local du terminal de comptage de la boucle For, présenté ci-contre, et sélectionner "Créer une constante" pour ajouter et câbler automatiquement une constante numérique à ce terminal.

10. Exécutez le VI à partir de la face-avant. Le VI trace le tableau du signal auto-indexé sur le graphe. La valeur initiale de *X* est 0 tandis que la valeur de delta *X* est 1.
11. Faites passer la valeur de delta *X* sur 0,5 et la valeur initiale de *X* sur 20. Exécutez de nouveau le VI.

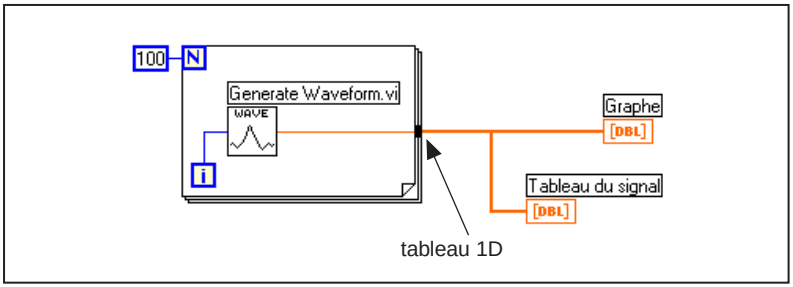
Remarquez que le graphe affiche désormais les mêmes 100 points, avec une valeur de départ de 20 et un delta *X* de 0,5 pour chaque point (voir l'axe des X). Dans un test chronométré, ce graphe peut correspondre à 50 secondes d'acquisition de données, en commençant à 20 secondes.

12. Pour voir un élément quelconque dans le tableau du signal, entrez l'indice de cet élément dans l'affichage de l'indice. Si vous entrez un nombre supérieur à la taille du tableau, l'écran s'assombrit, indiquant ainsi que vous n'avez pas d'élément bien défini pour cet indice.

Si vous souhaitez voir plusieurs éléments à la fois, vous pouvez réajuster la taille de l'indicateur du tableau. Pour cela, placez l'outil Flèche sur le coin inférieur droit du tableau. L'outil se transforme en curseur de redimensionnement (comme présenté ci-contre). Faites-le alors glisser vers la droite ou tout droit vers le bas. Le tableau affiche désormais plusieurs éléments selon un ordre d'indice ascendant, en commençant par l'élément correspondant à l'indice spécifié, comme le montre l'illustration suivante.



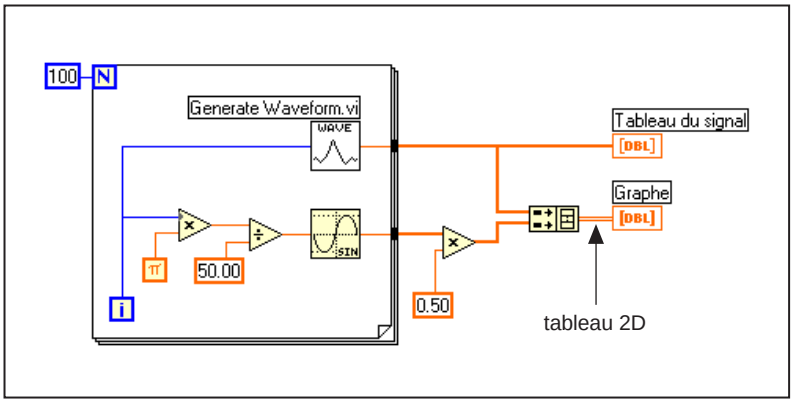
Dans le diagramme précédent, vous avez spécifié un *X* initial et une valeur de delta *X* pour le signal. La valeur initiale de *X* est zéro tandis que la valeur de delta *X* est 1. Vous pouvez donc câbler le tableau de la courbe directement au terminal du graphe sans le *X* initial et le delta *X* spécifiés, comme le montre l'illustration suivante.



13. Revenez au diagramme. Supprimez la fonction Assembler et les constantes numériques qui lui sont liées. Pour effacer la fonction et les constantes, sélectionnez-les avec l'outil Flèche, puis appuyez sur la touche <Supprimer>. Sélectionnez **Edition»Supprimer les fils incorrects**. Terminez de câbler le diagramme comme l'indique l'illustration précédente.
14. Exécutez le VI. Notez que le VI trace la forme d'onde avec une valeur initiale de *X* égale à 0 et une valeur de delta *X* égale à 1.

Graphes multicourbes

Vous pouvez créer des graphes multicourbes en construisant un tableau du type de données normalement transmises sur un graphe à tracé unique.



15. Continuez la construction de votre diagramme comme le montre le diagramme précédent.

Fonction Sinus (**Fonctions»Numérique»Trigonométrique**) : dans cet exercice, vous utilisez la fonction dans une boucle For pour construire un tableau de points représentant une période d'un signal sinusoïdal.

Fonction “Construire un tableau” (**Fonctions»Tableau)** : dans cet exercice, vous utilisez cette fonction pour créer la structure correcte (dans ce cas, un tableau 2D) des données permettant de représenter deux tableaux dans un graphe. Agrandissez la fonction “Construire un tableau” pour créer deux entrées en faisant glisser l’un des coins avec l’outil Flèche.

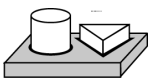
Constante Pi (**Fonctions»Numérique»Constantes numériques supplémentaires**) : notez que vous pouvez trouver les fonctions Multiplier et Diviser dans **Fonctions»Numérique**.

16. Basculez sur la face-avant. Exécutez le VI.

Notez que les deux formes d’onde apparaissent sur le même graphe. Pour les deux ensembles de données, la valeur initiale de *X* est par défaut 0, tandis qu*e* la valeur de delta *X* est par défaut 1.

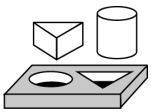
Remarque ***Vous pouvez changer l’apparence d’un tracé sur le graphe en ouvrant le menu local à partir de la légende d’un tracé. Par exemple, vous pouvez passer d’un graphe linéaire à un graphe à barres en choisissant*** **Tracés communs»Tracé à barres.**

17. Enregistrez le VI sous le nom `Tableaux et graphe.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l’exercice 5-1.

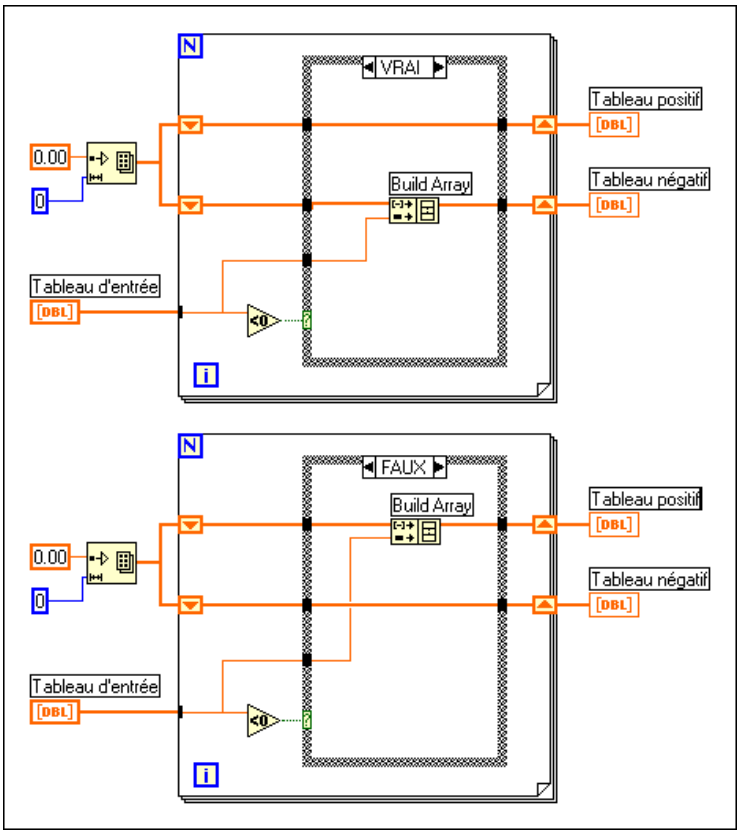
Dans l’exemple précédent, la boucle For a été exécutée 100 fois parce qu’une constante de `100` a été câblée au terminal de comptage. L’exercice suivant illustre une autre façon de déterminer le nombre de fois qu’une boucle s’exécutera.



Exercice 5-2. Utiliser l’auto-indexation pour les tableaux d’entrée

Votre objectif est d’ouvrir et de faire fonctionner un VI qui utilise l’auto-indexation dans une boucle For pour traiter un tableau.

1. Ouvrez le VI “Séparer les valeurs du tableau” (Separate Array Values.vi) en sélectionnant **Fichier»Ouvrir…** Le VI se trouve dans `Examples\General\arrays.llb`.
2. Ouvrez le diagramme. L’illustration suivante montre le diagramme avec les deux conditions VRAI et FAUX visibles.



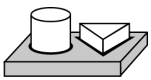
Notez que le fil de liaison du `tableau d'entrée` change d'apparence : en dehors de la boucle For, le fil de liaison est épais pour indiquer que le type de données transportées est un tableau, tandis qu'à l'intérieur de la boucle, le fil de liaison est fin pour indiquer un élément unique. Le $i^{ème}$ élément du tableau est indexé automatiquement au tableau durant chaque itération.

Utilisation de l'auto-indexation pour établir le comptage de la boucle For

N

Notez que le terminal de comptage reste non câblé. Lorsque vous utilisez l'auto-indexation sur un tableau entrant dans une boucle For, la boucle s'exécute conformément à la taille du tableau, éliminant ainsi le besoin de câbler une valeur au terminal de comptage. Si vous utilisez l'auto-indexation pour plusieurs tableaux, ou si vous fixez la valeur de comptage en plus d'auto-indexer un tableau, le nombre d'itérations réel est le plus petit des deux nombres.

3. Exécutez le VI. Des huit valeurs d'entrée, vous en voyez quatre dans le tableau des valeurs positives et quatre dans le tableau des valeurs négatives.
4. A partir du diagramme, câblez une constante de 5 au terminal de comptage de la boucle For. Exécutez le VI. Vous voyez trois valeurs dans le tableau des valeurs positives et deux dans le tableau des valeurs négatives, même si le tableau d'entrée possède encore huit éléments. Ceci démontre que si N est défini et si vous utilisez l'auto-indexation, le G compare la valeur de N et la taille du tableau : le plus petit de ces deux nombres est utilisé comme le nombre réel d'itérations de la boucle.
5. Fermez le VI et n'enregistrez pas les changements.

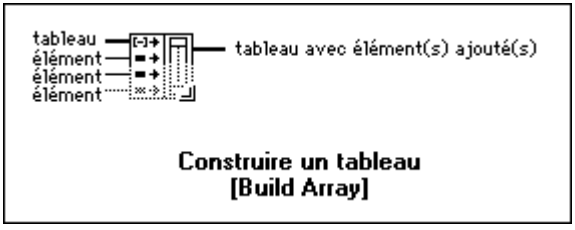


Fin de l'exercice 5-2.

Utilisation des fonctions du tableau

Le G possède de nombreuses fonctions pour manipuler les tableaux situés dans la palette **Fonctions»Tableau**. Ces fonctions sont notamment : "Remplacer un élément d'un tableau", "Rechercher dans un tableau 1D", "Classer un tableau 1D", "Renverser un tableau 1D" et "Multiplier les éléments du tableau". Pour plus d'informations sur les tableaux et les fonctions de tableau disponibles, reportez-vous au chapitre 14, *Commandes et indicateurs de tableaux et de clusters*, dans le *Manuel de référence de programmation en G* ou à **Référence en ligne»Fonction et Référence du VI**.

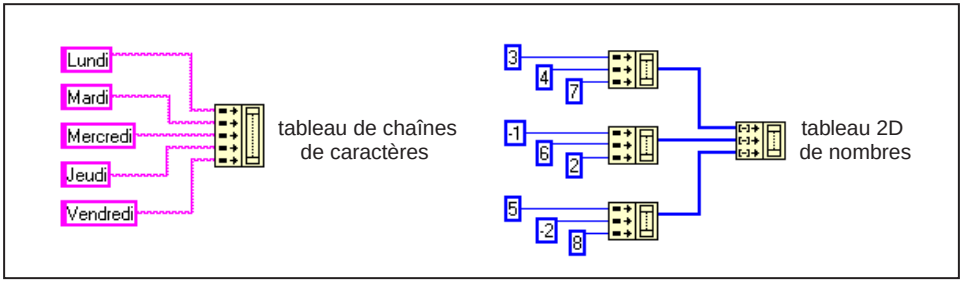
Construire un tableau



Fonction “Construire un tableau” (**Fonctions»Tableau**) : vous pouvez l’utiliser pour créer un tableau à partir de valeurs scalaires ou à partir d’autres tableaux. Initialement, la fonction “Construire un tableau” apparaît avec une entrée scalaire.

Vous pouvez ajouter autant d’entrées dont vous avez besoin pour la fonction “Construire un tableau”, et chaque entrée peut être un scalaire ou un tableau. Pour ajouter plusieurs entrées, ouvrez le menu local sur le côté gauche de la fonction, puis sélectionnez **Ajouter une entrée de type élément** ou **Ajouter une entrée de type tableau**. Vous pouvez également agrandir le nœud à l’aide du curseur de redimensionnement (placez l’outil Flèche sur le coin d’un objet pour le transformer en curseur de redimensionnement). Vous pouvez enlever les entrées en rétrécissant le nœud avec le curseur de redimensionnement, ou en sélectionnant **Supprimer une entrée**.

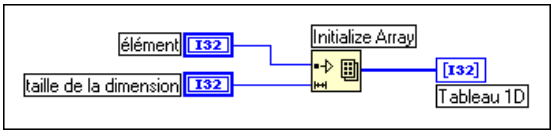
L’illustration suivante montre les deux façons de créer et d’initialiser des tableaux avec les valeurs provenant des constantes du diagramme. Sur la gauche, cinq constantes de type chaîne de caractères sont construites dans un tableau 1D de chaînes de caractères. Sur la droite, trois groupes de constantes numériques sont construits dans trois tableaux numériques 1D. Les trois tableaux sont ensuite combinés dans un tableau numérique 2D. Le résultat est un tableau 3 x 3 avec les rangées 3, 4, 7 ; –1, 6, 2 ; et 5, –2, 8.



Vous pouvez également créer un tableau en combinant d’autres tableaux avec des éléments scalaires. Par exemple, supposez que vous possédez deux tableaux et trois éléments scalaires que vous souhaitez combiner dans un nouveau tableau dans l’ordre suivant : tableau 1, scalaire 1, scalaire 2, tableau 2 et scalaire 3.

Initialiser un tableau

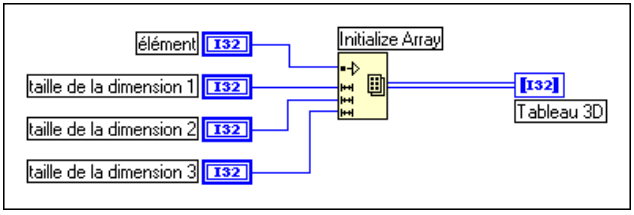
Utilisez cette fonction pour créer un tableau dont les éléments possèdent la même valeur. Dans l'illustration suivante, cette fonction crée un tableau 1D.



L'entrée "élément" détermine le type des données et la valeur de chaque élément. L'entrée "taille de la dimension" détermine la longueur du tableau. Par exemple, si `élément` est un entier long ayant pour valeur 5 et si `taille de la dimension` a une valeur de 100, le résultat est un tableau 1D de 100 entiers longs, tous égaux à 5. Vous pouvez câbler les entrées des terminaux de commande de la face-avant, comme montré dans l'illustration précédente, depuis les constantes du diagramme, ou depuis les calculs sur d'autres parties de votre diagramme.

Pour créer et initialiser un tableau possédant plusieurs dimensions, ouvrez le menu local sur le côté inférieur gauche de la fonction et sélectionnez **Ajouter une dimension**. Vous pouvez également utiliser le curseur de redimensionnement pour agrandir le nœud "Initialiser un tableau" et pour ajouter davantage d'entrées de type taille de la dimension (une entrée pour chaque dimension supplémentaire). Vous pouvez enlever des dimensions en rétrécissant le nœud ; pour cela, sélectionnez Supprimer une dimension dans le menu local de la fonction ou utilisez le curseur de redimensionnement.

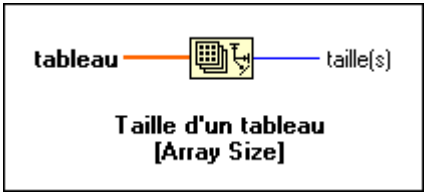
Le diagramme suivant montre la façon d'initialiser un tableau 3D.

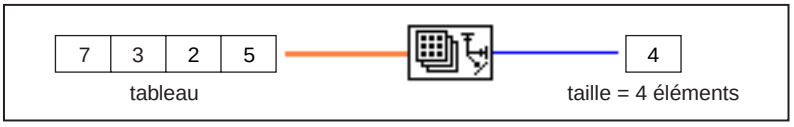


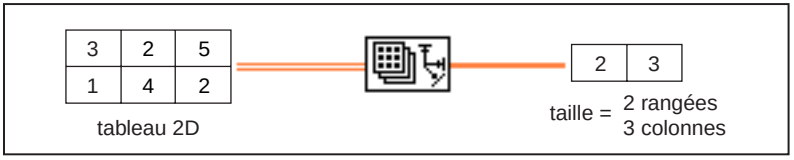
Si toutes les entrées "taille de la dimension" sont égales à zéro, la fonction crée un tableau vide du type et de la dimension spécifiés.

Taille d’un tableau

La fonction “Taille d’un tableau” retourne le nombre d’éléments dans le tableau d’entrée.

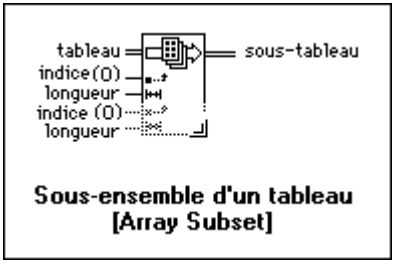






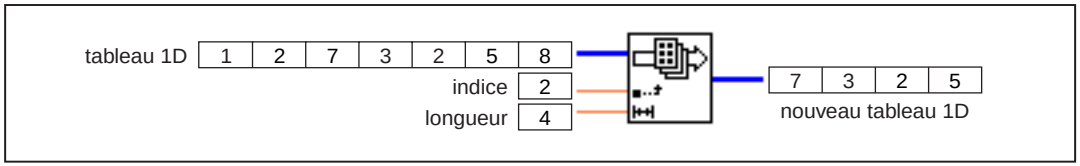
Sous-ensemble d’un tableau

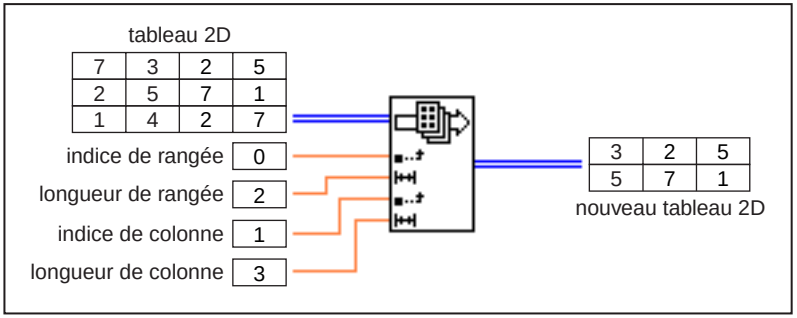
Vous pouvez utiliser cette fonction pour extraire une portion d’un tableau ou d’une matrice.



La fonction “Sous-ensemble d’un tableau” retourne une portion d’un tableau commençant à un indice donné et contenant une “longueur” d’éléments. Les illustrations suivantes montrent des exemples d’utilisation

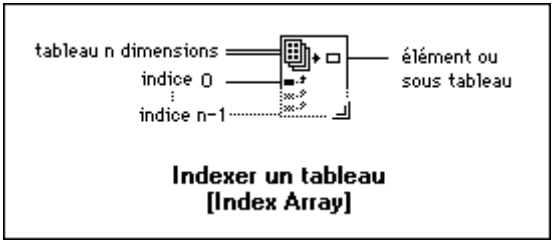
de la fonction “Sous-ensembles d’un tableau”. Notez que l’indice de tableau commence par 0.



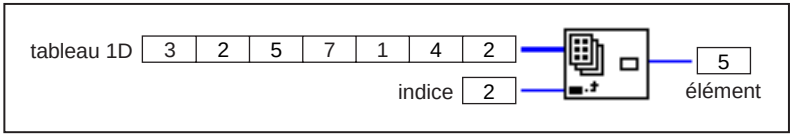


Indexer un tableau

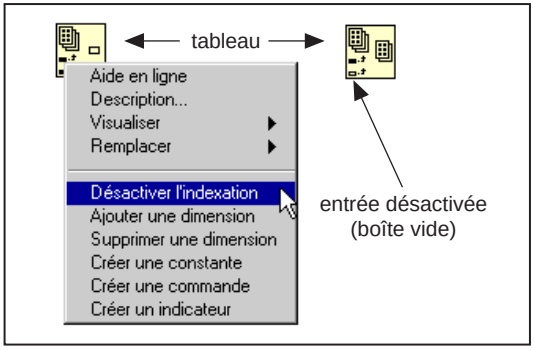
La fonction “Indexer un tableau” permet d’accéder à un élément d’un tableau.



L’illustration suivante montre un exemple d’une fonction “Indexer un tableau” accédant au troisième élément d’un tableau. Remarquez que l’indice du troisième élément est 2 parce que le premier élément possède l’indice 0.

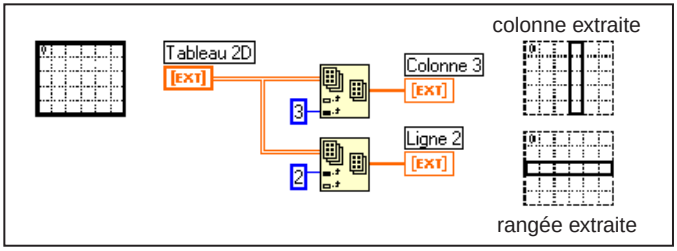


Vous pouvez également utiliser cette fonction pour découper une ou plusieurs dimensions d'un tableau multi-dimensionnel afin de créer un sous-tableau de l'original. Pour cela, étendez la fonction "Indexer un tableau" pour inclure deux entrées d'indice, puis sélectionnez la commande **Désactiver l'indexation** dans le menu local du second terminal d'indice, comme présenté dans l'illustration suivante. Vous avez désormais désactivé l'accès à une colonne spécifique du tableau. En lui donnant un indice de rangée, le résultat est un tableau dont les éléments sont les éléments de la rangée spécifiée du tableau 2D. Vous pouvez également désactiver l'indexation sur le terminal de la rangée.

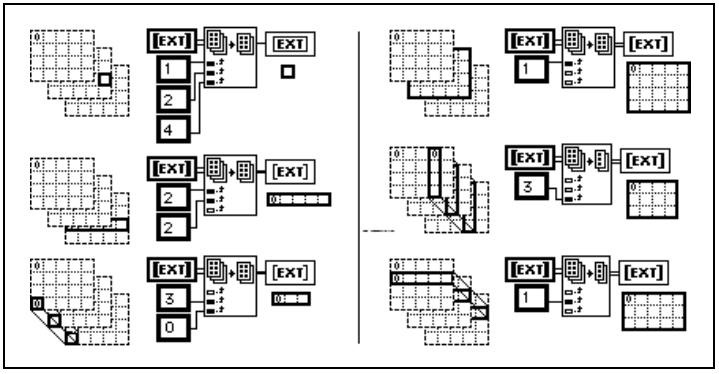


Notez que le symbole du terminal d'indice change, pour passer d'un rectangle plein à une boîte vide lorsque vous désactivez l'indexation. Pour restaurer un indice désactivé, utilisez la commande **Activer l'indexation** du même menu.

Vous pouvez extraire des sous-tableaux avec toute combinaison possible des dimensions. L'illustration suivante montre la façon d'extraire un tableau de colonnes ou de rangées 1D depuis un tableau 2D.



D’un tableau 3D, vous pouvez extraire un tableau 2D en désactivant deux terminaux d’indices, ou un tableau 1D en désactivant un seul terminal d’indice. La figure suivante montre plusieurs façons de découper un tableau 3D.



Les règles d’utilisation de la fonction “Indexer un tableau” pour découper des tableaux sont les suivantes :

- La dimension de l’objet de sortie doit être égale au nombre de terminaux d’indice désactivés. Par exemple :
  - aucun terminal désactivé = élément scalaire
  - un terminal désactivé = composante 1D
  - deux terminaux désactivés = composante 2D
- Les valeurs câblées aux terminaux activés doivent identifier les éléments de sortie.

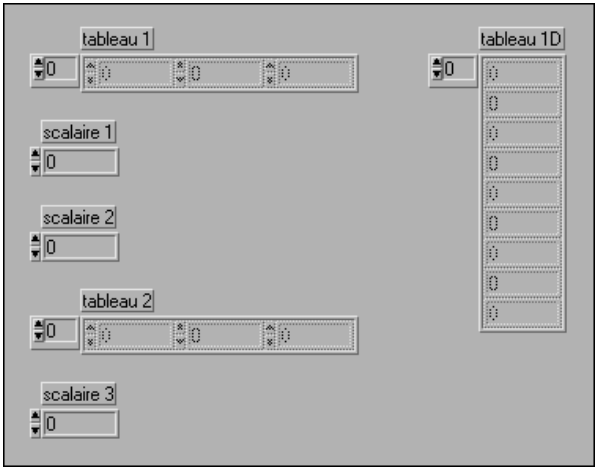
Vous pouvez ainsi interpréter l’exemple inférieur gauche précédent comme une commande pour générer un tableau 1D de tous les éléments à la colonne 0 et la rangée 3. Vous pouvez interpréter l’exemple supérieur droit comme une commande pour générer le tableau 2D de la page 1. Le nouveau 0ème élément est le plus proche de l’original, comme montré dans l’illustration précédente.

Exercice 5-3. Utiliser la fonction “Construire un tableau”

Votre objectif est d’utiliser la fonction “Construire un tableau” pour combiner les éléments et les tableaux dans un tableau plus grand.

Face-avant

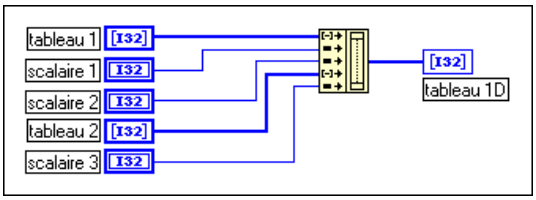
1. Créez une nouvelle face-avant, comme présenté dans l’illustration suivante.



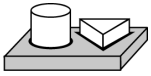
2. Placez une commande numérique dans la palette **Commandes» Numérique** et nommez-la `scalaire 1`. Faites passer sa représentation sur `I32`.
3. Copiez la commande et collez-la pour créer deux autres commandes numériques que vous nommerez `scalaire 2 et scalaire 3`.
4. Créez un tableau de commandes numériques et nommez-le `tableau 1`. Copiez et collez-le, puis nommez le nouveau tableau `tableau 2`.
5. Agrandissez les tableaux et entrez les valeurs 1 à 9 dans `tableau 1`, `scalaire 1, scalaire 2, tableau 2` et `scalaire 3`, comme montré dans l’illustration ci-dessus.
6. Copiez le tableau et collez-le, puis transformez-le en indicateur. Nommez celui-ci `tableau 1D`. Agrandissez-le pour afficher neuf valeurs.

Diagramme

7. Placez une fonction “Construire un tableau” (**Fonctions»Tableau**) sur le diagramme. Agrandissez la fonction avec l’outil Flèche pour avoir cinq entrées.
8. Ouvrez le menu local de la première entrée dans le nœud “Construire un tableau”, puis sélectionnez **Changer en tableau**. Faites de même pour la quatrième entrée.
9. Câblez les tableaux et les scalaires au nœud. Le tableau de sortie est un tableau 1D composé des éléments du `tableau 1` suivis par `scalaire 1, scalaire 2`, les éléments du `tableau 2` et `scalaire 3`, comme l’indique l’illustration suivante.



10. Exécutez le VI. Vous pouvez voir les valeurs de `scalaire 1, scalaire 2, scalaire 3, tableau 1` et `tableau 2` dans un même tableau 1D.
11. Enregistrez le VI sous le nom `Construire un tableau.vi` dans le répertoire `LabVIEW\Activity`.



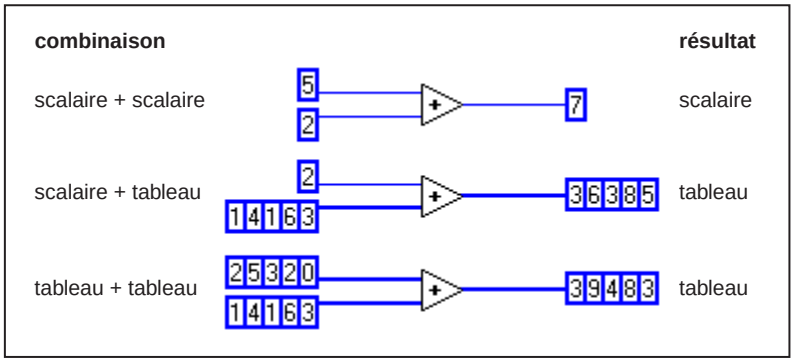
Fin de l’exercice 5-3.

Utilisation efficace de la mémoire : réduction des copies de données

Pour économiser l’espace mémoire, vous pouvez utiliser des tableaux d’éléments simple précision au lieu de tableaux en double précision. Pour des informations sur la façon dont la mémoire est affectée, consultez la section *Gestion de la mémoire* au chapitre 28, *Performance*, dans le *Manuel de référence de programmation en G*.

Qu’est-ce que le polymorphisme ?

Le *polymorphisme* est la capacité d’une fonction à adapter des données d’entrée de différents types, dimensions ou représentations. La plupart des fonctions en G sont polymorphes. Par exemple, les illustrations suivantes montrent les combinaisons polymorphes de la fonction Ajouter.



Dans la première combinaison, les deux nombres scalaires sont ajoutés : le résultat est un scalaire. Dans la seconde combinaison, le scalaire est ajouté à chaque élément du tableau : le résultat est un tableau. Un tableau est un regroupement de données. Dans la troisième combinaison, chaque élément d’un tableau est ajouté à l’élément correspondant de l’autre tableau. Vous pouvez également utiliser d’autres combinaisons, telles que des clusters de numériques ou des tableaux de clusters.

Vous pouvez appliquer ces principes à d’autres fonctions du G et à d’autres types de données. Les fonctions en G sont polymorphes à des degrés différents. Certaines fonctions peuvent accepter des entrées numériques et booléennes, tandis que d’autres peuvent accepter une combinaison de tout autre type de données. Pour plus d’informations sur le polymorphisme, reportez-vous à **Référence en ligne»Fonction et Référence du VI**.

Clusters

Un cluster est un type de données pouvant contenir des éléments de données de différents types. Le cluster du diagramme que vous construirez dans l'exercice 5-4 regroupe les éléments de données connexes provenant de divers endroits du diagramme, réduisant ainsi l'encombrement des fils de liaison. Lorsque vous utilisez des clusters, vos sous-VIs requièrent moins de terminaux de connexion. Un cluster est semblable à un "record" en Pascal ou à une structure en C. Vous pouvez considérer un cluster comme un faisceau de fils, comme dans le cas d'un câble téléphonique. Chaque fil du câble représenterait un élément différent du cluster. Les composantes incluent la valeur initiale de *X* (0), la valeur de delta *X* (1), et le tableau Y (données du signal, fournies par les constantes numériques du diagramme). Dans le G, utilisez la fonction Assembler pour créer un cluster. Pour plus d'informations sur les clusters, reportez-vous au chapitre 14, *Commandes et indicateurs de tableaux et de clusters*, dans le *Manuel de référence de programmation en G*.

Graphes

Un *graphe* est une représentation graphique à deux dimensions d'un ou de plusieurs tableaux de données appelés tracés. Les trois types de graphes disponibles dans la palette **Commandes»Graphe** sont indiqués ci-après :

- Graphe XY
- Graphe
- Graphe d'intensité

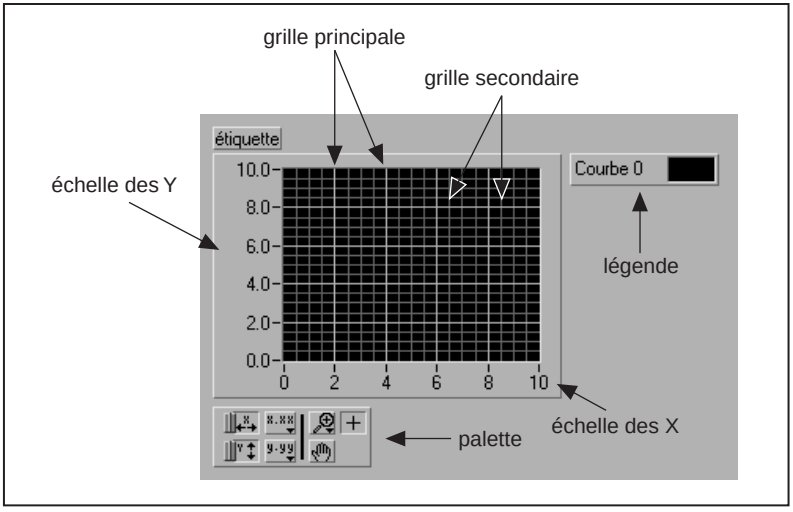
La différence entre un graphe et un graphe déroulant est qu'un graphe trace les données comme un bloc, alors qu'un graphe déroulant trace les données point par point, ou tableau par tableau.

Pour des exemples de VIs de graphes, reportez-vous au répertoire `Examples\General\Graphs`.

Personnalisation de graphes

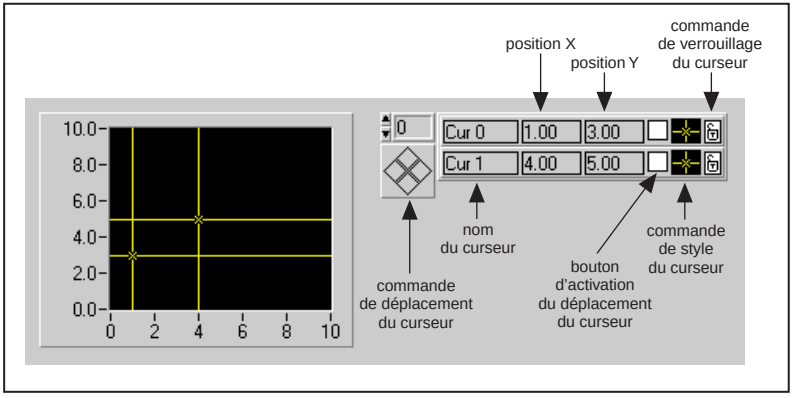
Les graphes et les graphes XY possèdent des éléments optionnels que vous pouvez afficher ou masquer grâce au sous-menu **Visualiser** du menu local du graphe. Les options incluent une légende, avec laquelle vous pouvez définir la couleur et le style d'un tracé donné, une palette permettant de changer les options de mise à l'échelle et de format lors de

l'exécutiondu VI, et l'affichage du curseur. L'illustration suivante présente toutes les composantes en option pour un graphe, à l'exception de l'affichage du curseur.



Curseurs des graphes

Vous pouvez placer des curseurs et un affichage du curseur sur tous les graphes, et vous pouvez nommer le curseur sur le tracé. Vous pouvez verrouiller un curseur sur un tracé, et vous pouvez déplacer plusieurs curseurs en même temps. Le nombre de curseurs que peut accepter un graphe est illimité. L'illustration suivante montre un graphe avec l'affichage du curseur.



Pour des informations plus détaillées concernant la personnalisation des graphes, consultez le chapitre 15, *Commandes et indicateurs de graphe et graphe déroulant*, dans le *Manuel de référence de programmation en G*.

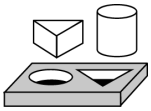
Reportez-vous au VI “Zoom sur un graphe” (ZoomGraph.vi) dans `Examples\General\Graphs\zoom.llb` pour un exemple de lecture des valeurs du curseur, et de programmation de zoom-avant et de zoom-arrière dans un graphe utilisant les curseurs.

Axes des graphes

Vous pouvez formater les échelles d’un graphe pour représenter un temps absolu ou relatif. Utilisez le format de temps absolu pour afficher le temps, la date ou les deux sur votre échelle. Si vous ne souhaitez pas que le G définisse lui-même une date, utilisez le format de temps relatif.
Pour sélectionner le format de temps relatif ou absolu, ouvrez le menu local du graphe déroulant et sélectionnez l’échelle que vous souhaitez modifier. Sélectionnez **Formatage…** Ceci active la boîte de dialogue **Formatage**, que vous pouvez utiliser pour spécifier différents attributs du graphe déroulant.

Tableau d’acquisition de données

Les données retournées depuis la carte d’acquisition grâce aux VIs d’acquisition de données peuvent se présenter sous la forme d’une valeur unique, d’un tableau 1D ou d’un tableau 2D. Vous pouvez trouver plusieurs exemples de graphes dans le répertoire `Examples\General\Graphs`, contenant des VIs permettant d’effectuer différentes fonctions en utilisant les tableaux et les graphes.

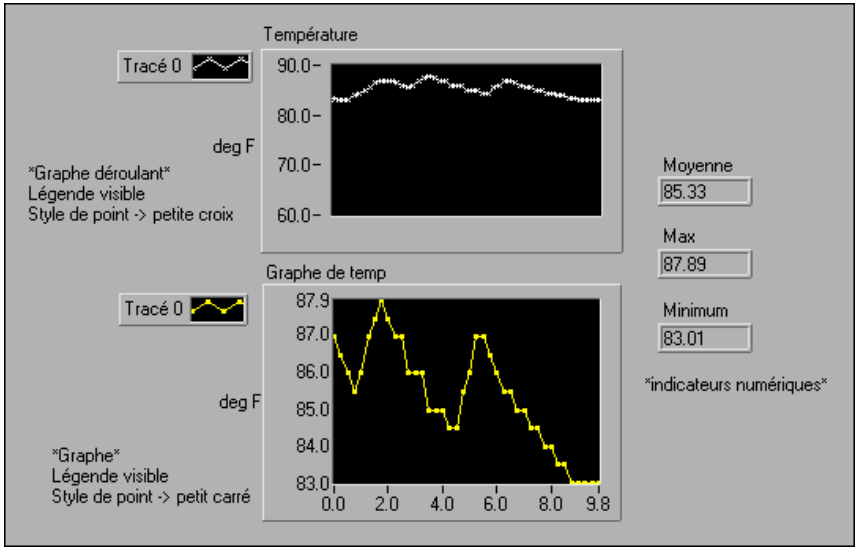


Exercice 5-4. Utiliser les VIs Graphe et Analyse

Votre objectif est de construire un VI qui mesure la température et affiche les valeurs en temps réel. Il doit également afficher les températures moyenne, maximale et minimale.

Face-avant

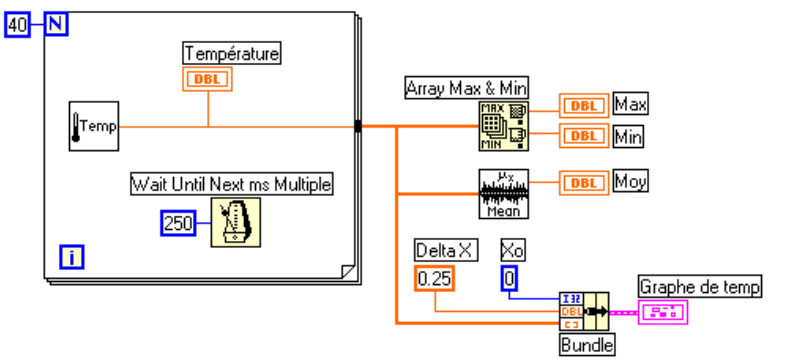
1. Créez une nouvelle face-avant comme montré dans l’illustration suivante. Vous pouvez modifier les styles de points du graphe déroulant et du graphe en ouvrant le menu local sur leur légende. Mettez à l’échelle les graphes déroulants comme il convient.



Le graphe Température représente la température à mesure qu'elle est reçue. Après acquisition, le VI trace les données dans le `Graphe de température`. Les indicateurs numériques `Moyenne`, `Max` et `Min` affichent les températures moyenne, maximale et minimale.

Diagramme

2. Construisez le diagramme comme indiqué dans l'illustration suivante.



VI "Thermomètre numérique" (Digital Thermometer.vi) (**Fonctions» Sélectionner un VI** dans le répertoire `LabVIEW\Activity`) : retourne une mesure de la température.

Fonction “Attendre un multiple de ms” (**Fonctions»Temps & dialogue**) : dans cet exercice, cette fonction force la boucle For à s’exécuter toutes les 0,25 secondes (250 millisecondes).

Constante numérique (**Fonctions»Numérique**) : vous pouvez également ouvrir le menu local de la fonction “Attendre un multiple de ms” et sélectionner **Créer une constante** pour créer et câbler automatiquement une constante numérique.

Fonction “Max. & Min. d’un tableau” (**Fonctions»Tableau**) : dans cet exercice, cette fonction retourne les températures maximale et minimale mesurées durant l’acquisition.

VI Moyenne (Mean.vi) (**Fonctions»Analyse»Probabilités et statistiques ou Fonctions»Analyse de base pour les utilisateurs du Package de base de LabVIEW**) : retourne la moyenne des mesures de la température.

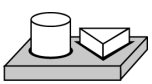
Fonction Assembler (**Fonctions»Cluster**) : assemble les composantes du tracé dans un cluster. Ces composantes incluent la valeur initiale de *X* (0), la valeur de delta *X* (0,25), et le tableau Y (données de température). Utilisez l’outil Flèche pour modifier la dimension de la fonction en faisant glisser l’un des angles.

La boucle For s’exécute 40 fois. La fonction “Attendre un multiple de ms” provoque une itération toutes les 250 millisecondes. Le VI enregistre les mesures de température dans un tableau créé à la bordure de la boucle For (auto-indexation). Après la fin de l’exécution de la boucle For, le tableau est transmis aux sous-VIs et au `Graphe de la température`.

La fonction “Max. & Min. d’un tableau” retourne les températures maximale et minimale. Le VI Moyenne retourne la moyenne des mesures de température.

Pour finir, votre VI assemble le tableau des données avec une valeur initiale de X de 0 et une valeur de delta X de 0,25. Le VI requiert une valeur de delta X de 0,25 afin que le VI représente les points du tableau de température toutes les 0,25 secondes sur le graphe.

3. Revenez à la face-avant et exécutez le VI.
4. Enregistrez le VI sous le nom `Analyse de température.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l’exercice 5-4.

Graphes d'intensité

LabVIEW propose deux méthodes pour afficher des données à trois dimensions : le graphe déroulant d'intensité et le graphe d'intensité. Les deux tracés d'intensité acceptent des tableaux de nombres à deux dimensions, où chaque nombre correspond à une couleur. Vous pouvez définir la correspondance des couleurs de deux façons : de manière interactive, en utilisant une échelle de rampe de couleur en option, ou par programmation, en utilisant un attribute node du graphe déroulant. Pour des exemples utilisant le graphe déroulant d'intensité et le graphe d'intensité, reportez-vous à la bibliothèque `intgraph.llb` dans le répertoire `Examples\General\Graphs`.

6

Chaînes de caractères et E/S sur fichiers

Ce chapitre introduit les commandes et les indicateurs de type chaîne de caractères et les opérations d'entrée et sortie de fichier. Il présente également des exercices illustrant la façon d'accomplir les tâches suivantes :

- Créer des commandes et des indicateurs de chaînes de caractères
- Utiliser des fonctions de chaînes de caractères
- Réaliser des opérations d'entrée et sortie de fichier
- Enregistrer des données sur des fichiers au format tableur
- Ecrire des données et lire des données de fichiers texte

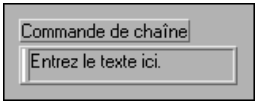
Chaînes de caractères

Une chaîne de caractères est un regroupement de caractères ASCII. Dans une commande d'instrument, vous pouvez transférer des données numériques comme des chaînes de caractères, puis convertir ces chaînes de caractères en nombres. L'enregistrement de données numériques sur un disque peut également impliquer des chaînes de caractères. Pour enregistrer des nombres dans un fichier ASCII, vous devez d'abord convertir les nombres en chaînes de caractères avant d'écrire les nombres dans un fichier sur le disque.

Pour des exemples d'utilisation de chaînes de caractères, reportez-vous à la bibliothèque `Examples\General\strings.llb`.

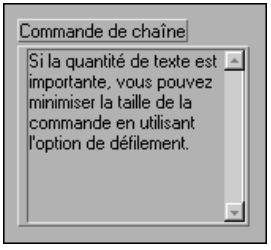
Création de commandes et d'indicateurs de type chaîne de caractères

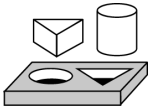
Vous pouvez trouver la commande et l'indicateur de chaîne de caractères, comme présenté ci-contre, dans **Commandes»Chaîne de caractères & table**. Vous pouvez entrer ou changer le texte dans une commande de chaîne de caractères avec l'outil Doigt ou l'outil Texte. Vous pouvez changer la dimension des commandes et des indicateurs en faisant glisser l'un des angles avec l'outil Flèche.



Chaînes de caractères et E/S sur fichiers

Si vous souhaitez réduire l'espace occupé par une commande ou un indicateur de chaîne de caractères sur la face-avant, sélectionnez **Visualiser»Barre de défilement**. Si cette option est grisée, vous devez augmenter la taille verticale de la fenêtre pour la rendre disponible.



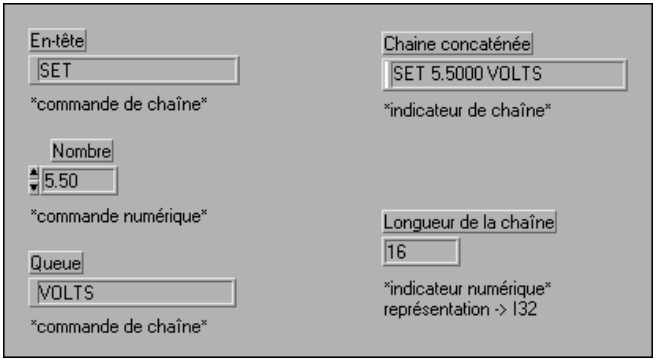


Exercice 6-1. Concaténer une chaîne de caractères

LabVIEW possède de nombreuses fonctions pour manipuler des chaînes de caractères. Pour cet exercice, votre objectif est d'utiliser certaines des fonctions de chaîne de caractères pour convertir un nombre en une chaîne de caractères. Vous allez également concaténer cette chaîne avec d'autres chaînes pour former une chaîne de caractères de sortie unique.

1. Ouvrez une nouvelle face-avant et placez les objets présentés dans l'illustration suivante. Assurez-vous de modifier les commandes et les indicateurs comme il convient.

Face-avant

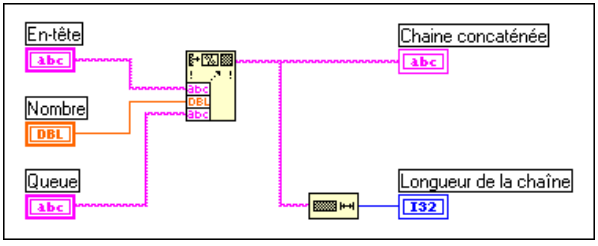


Les deux commandes de type chaîne de caractères et la commande numérique peuvent être combinées en une seule chaîne de caractères de sortie et affichées dans l'indicateur de chaîne de caractères. L'indicateur numérique affiche la longueur de la chaîne de caractères.

La sortie chaîne de caractères combinée possède, dans cet exemple, un format similaire aux commandes utilisées pour communiquer avec des instruments GPIB (IEEE 488) et série (RS-232 ou RS-422). Reportez-vous à la Partie II, *Interfaces d'E/S*, dans ce manuel, pour en apprendre davantage sur les chaînes de caractères utilisées pour les commandes d'instruments.

Diagramme

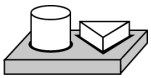
2. Construisez le diagramme présenté dans l'illustration suivante.



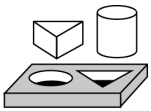
La fonction "Formater en chaîne de caractères" (**Fonctions»Chaîne de caractères**) permet de concaténer et de formater des nombres et des chaînes de caractères en une seule chaîne de caractères de sortie. Redimensionnez l'icône pour ajouter trois entrées d'argument.

La fonction “Longueur d’une chaîne de caractères” (**Fonctions»Chaîne de caractères**) retourne le nombre de caractères de la chaîne concaténée.

3. Exécutez le VI. Notez que la fonction “Formater en chaîne de caractères” permet de concaténer les deux commandes de chaîne de caractères et la commande numérique en une seule chaîne de caractères de sortie.
4. Enregistrez le VI sous le nom `Construire une chaîne de caractères.vi`. Vous utiliserez ce VI dans l’exercice suivant.



Fin de l’exercice 6-1.



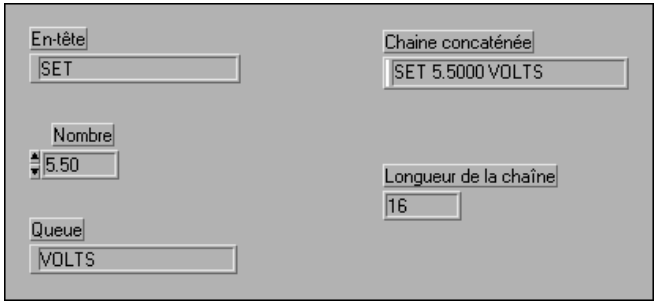
Exercice 6-2. Utiliser des chaînes de format

Votre objectif est de créer une chaîne de caractères suivant le format précisé.

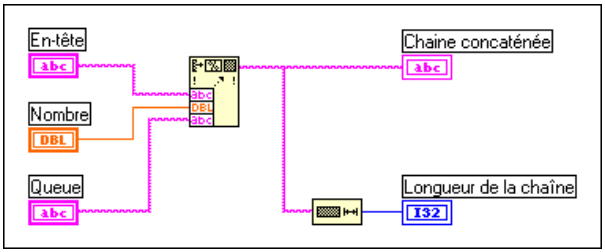
Utilisez le VI “Construire une chaîne de caractères” que vous avez créé dans l’exercice 6-1 pour créer une chaîne de format. Grâce aux chaînes de format, vous pouvez spécifier le format des arguments, y compris la largeur de champ, la base (hexadécimale, octale, etc.), et tout texte séparant les arguments.

Face-avant

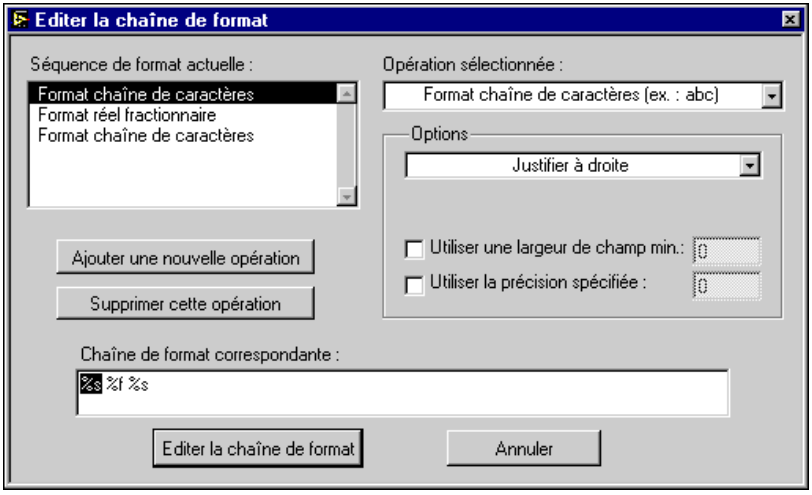
1. Ouvrez le VI “Construire une chaîne de caractères” que vous avez créé dans l’exercice 6-1.



Diagramme



2. Ouvrez le menu local de la fonction “Formater en chaîne de caractères”, puis sélectionnez **Editer la chaîne de format**. La boîte de dialogue suivante apparaît.

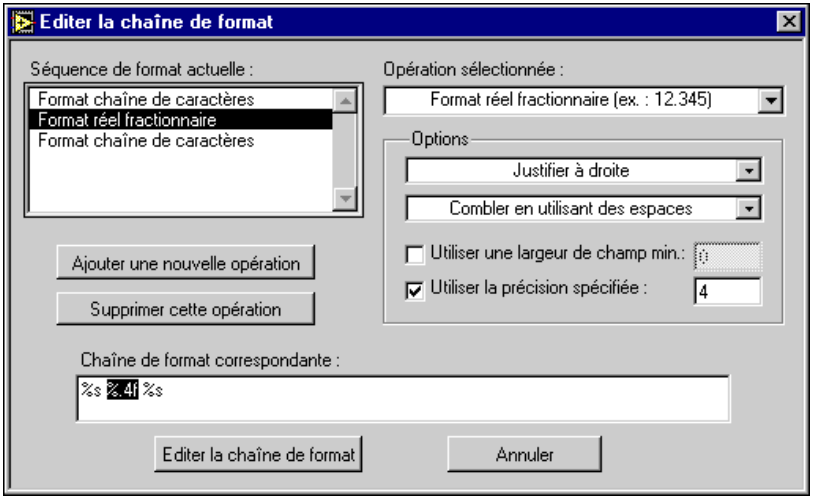


☞ **Remarque** ***Vous pouvez également double-cliquer sur l’icône pour accéder directement à la boîte de dialogue Editer la chaîne de format.***

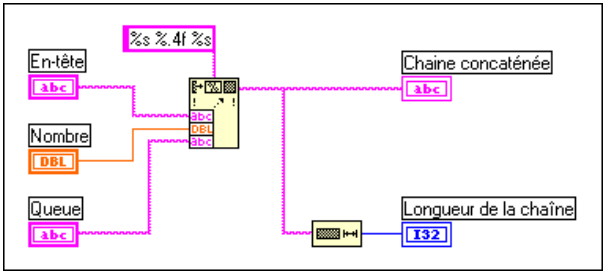
Notez que la case “Séquence de format actuelle” contient les types d’argument, dans l’ordre dans lequel vous les avez câblés.

3. Réglez la précision du numérique sur 4.
   a. Mettez en surbrillance `Format réel fractionnaire` dans la liste Séquence de format actuelle.
   b. Cliquez dans la case à cocher “Utiliser la précision spécifiée”.
   c. Mettez en surbrillance l’indicateur numérique à côté de la case à cocher “Utiliser la précision spécifiée”, tapez 4, puis appuyez sur

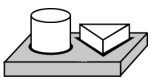
<Enter> **(Windows)** ; sur <return> (**Macintosh**) ; sur <Return> (**Sun**) ; ou sur <Enter> (**HP-UX**). L'illustration suivante montre les options sélectionnées pour définir la précision d'un nombre.



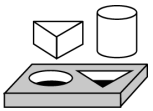
4. Appuyez sur le bouton **Editer la chaîne de format**. Ceci permet d'insérer automatiquement les informations correctes sur la chaîne de format et de connecter une chaîne de format à la fonction, comme montré dans l'illustration suivante.



5. Revenez à la face-avant et entrez du texte dans les deux commandes de chaîne de caractères et un nombre dans la commande numérique. Exécutez le VI.
6. Enregistrez et fermez le VI. Appelez-le `Chaîne de format.vi`.



Fin de l'exercice 6-2.

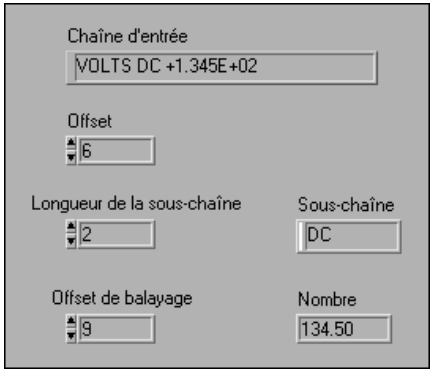


Exercice 6-3. Sous-ensembles d'une chaîne de caractères et extraction de nombres

Votre objectif est de prendre un sous-ensemble d'une chaîne de caractères correspondant à la représentation en chaîne de caractères d'un nombre et de le convertir en une valeur numérique.

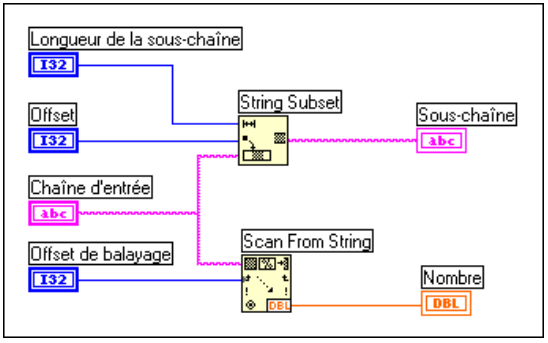
Face-avant

1. Ouvrez le VI "Analyse de chaine" (`Parse String.vi`) dans la bibliothèque `Examples\General\Strings.llb`. Exécutez le VI en utilisant les entrées par défaut. Notez que le sous-ensemble d'une chaîne de caractères de *DC* est choisi pour la chaîne d'entrée. Remarquez également que la partie numérique de la chaîne de caractères a été analysée et convertie en nombre. Vous pouvez essayer différentes valeurs de commande (souvenez-vous que les chaînes de caractères, comme les tableaux, sont indexées à partir de zéro), ou vous pouvez afficher le diagramme pour voir la façon dont les composantes de la chaîne d'entrée sont traitées.



Diagramme

2. Ouvrez le diagramme du VI “Analyse de chaine” (Parse String.vi). Ce diagramme est représenté dans l’illustration suivante.



LabVIEW utilise les fonctions “Sous-ensemble d’une chaîne de caractères” et “Balayer une chaîne de caractères” pour analyser la chaîne d’entrée.

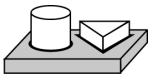
La fonction “Sous-ensemble d’une chaîne de caractères” (**Fonctions» Chaîne de caractères**) retourne la sous-chaîne de caractères commençant par **offset** et contenant le nombre de caractères **longueur**. Le premier offset de caractère est zéro.

Dans de nombreux cas, vous devez convertir les chaînes de caractères en nombres : par exemple, vous devez convertir une chaîne de caractères de données reçue d’un instrument en des valeurs numériques.

La fonction “Balayer une chaîne de caractères” (**Fonctions»Chaîne de caractères**) permet de balayer une chaîne de caractères et de convertir des caractères numériques valides (0 à 9, +, –, e, E, et point) en nombres. Si vous câblez l’entrée chaîne de format, la fonction “Balayer une chaîne de caractères” fait les conversions conformément au format précisé. Si vous ne câblez pas l’entrée chaîne de format, la fonction “Balayer une chaîne de caractères” réalise des conversions par défaut pour chaque terminal d’entrée par défaut dans la fonction. Cette fonction commence par balayer la **chaîne de caractères** à partir d’**offset**. L’offset du premier caractère est zéro.

La fonction “Balayer une chaîne de caractères” est utile lorsque vous connaissez la longueur de l’en-tête (`VOLTS DC` dans l’exemple montré ici), ou lorsque la chaîne de caractères ne contient que des caractères numériques valides.

3. Fermez le VI en sélectionnant **Fichier»Fermer**. N’enregistrez pas le VI.



Fin de l’exercice 6-3.

E/S sur fichiers

Les fonctions “E/S sur fichiers” du G (**Fonctions»E/S sur fichiers**) sont des outils puissants et souples destinés à utiliser des fichiers. Outre la lecture et l’écriture des données, les fonctions “E/S sur fichiers” de LabVIEW déplacent et renomment les fichiers et les répertoires, créent des fichiers (type tableur) de texte lisible ASCII, et écrivent des données sous la forme binaire pour augmenter la vitesse et la compacité.

Vous pouvez enregistrer ou récupérer des données de fichiers sous trois formats différent.

- Structure d’octet ASCII : vous devriez enregistrer vos données sous le format ASCII lorsque vous souhaitez y accéder à partir d’une autre application, comme un traitement de texte ou un tableur. Pour enregistrer des données de cette manière, vous devez convertir toutes les données en chaînes de caractères ASCII.
- Fichiers d’enregistrement de données : ces fichiers sont au format binaire (seul format auquel le G peut accéder). Les fichiers d’enregistrement de données sont similaires aux fichiers de base de données parce que vous pouvez enregistrer plusieurs types de données différents en un seul enregistrement d’un fichier.
- Structure d’octet binaire : il s’agit de la méthode la plus compacte et rapide pour enregistrer des données. Vous devez convertir les données au format chaîne binaire et vous devez connaître exactement les types de données que vous utilisez pour enregistrer et récupérer les données dans les fichiers.

Cette section traite des fichiers ASCII de structure d’octet car il s’agit du format de fichier de données le plus courant. Pour des exemples d’E/S sur fichiers, reportez-vous aux VIs du répertoire `Examples\File`.

Fonctions “E/S sur fichiers”

La plupart des opérations d’E/S sur fichiers comportent trois étapes fondamentales : l’ouverture d’un fichier existant ou la création d’un nouveau fichier ; l’écriture dans le fichier ou la lecture de celui-ci ; et la fermeture du fichier. LabVIEW contient ainsi de nombreux VIs utilitaires dans **Fonctions»E/S sur fichiers**. Cette section décrit les neuf utilitaires de niveau supérieur. Ces fonctions Utilitaire sont construites sur des VIs de niveau intermédiaire qui incorporent la gestion d’erreur avec les fonctions “E/S sur fichiers”.

Le VI “Ecrire des caractères dans un fichier” (Write Characters To File.vi) permet d’écrire une chaîne de caractères sur un nouveau fichier de structure d’octet ou d’ajouter la chaîne de caractères à un fichier existant. Ce VI ouvre ou crée le fichier, écrit les données, puis ferme le fichier.

Le VI “Lire des caractères dans un fichier” (Read Characters From File.vi) lit un nombre défini de caractères dans un fichier de structure d’octet, en commençant à un offset de caractère spécifié. Ce VI ouvre le fichier avant la lecture et le referme ensuite.

Le VI “Lire des lignes dans un fichier” (Read Lines From File.vi) lit un nombre défini de lignes dans un fichier de structure d’octet, en commençant à un offset de caractère spécifié. Ce VI ouvre le fichier avant la lecture et le referme ensuite.

Le VI “Ecrire dans un fichier tableur” (Write To Spreadsheet File.vi) convertit un tableau 1D ou 2D de nombres simple précision en une chaîne de caractères, puis écrit la chaîne de caractères dans un nouveau fichier de structure d’octet ou l’ajoute à un fichier existant. Si vous le souhaitez, vous pouvez aussi transposer les données. Ce VI ouvre ou crée d’abord le fichier et le referme ensuite. Vous pouvez utiliser ce VI pour créer des fichiers texte lisibles par la plupart des tableurs.

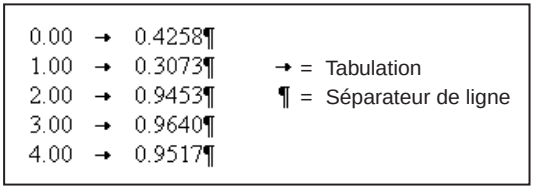
Le VI “Lire dans un fichier tableur” (Read From Spreadsheet File.vi) lit un nombre défini de lignes ou de rangées dans un fichier texte numérique, en commençant à un offset de caractère spécifié, puis convertit les données en un tableau 2D de nombres en simple précision. Si vous le désirez, vous pouvez transposer le tableau. Ce VI commence par ouvrir le fichier et le referme ensuite. Vous pouvez utiliser ce VI pour lire les fichiers tableurs enregistrés sous un format texte.

Pour des fonctions “E/S sur fichiers” supplémentaires, sélectionnez la sous-palette **Fonction»E/S sur fichiers»VIs de fichiers binaires** ou **Fonction»E/S sur fichiers»Fonctions de fichiers avancées**.

Ecriture dans un fichier tableur

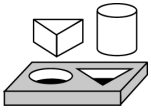
Une application très courante d'enregistrement des données dans un fichier est le formatage du fichier texte afin de pouvoir l'ouvrir dans un tableur.

Dans la plupart des tableurs, des tabulations séparent les colonnes et des caractères fin de ligne séparent les rangées, comme montré dans la figure suivante.



Si vous ouvrez le fichier avec un tableur, la table suivante apparaît.

	A	B	C
1	0	0.4258	
2	1	0.373	
3	2	0.9453	
4	3	0.964	
5	4	0.9517	
6			



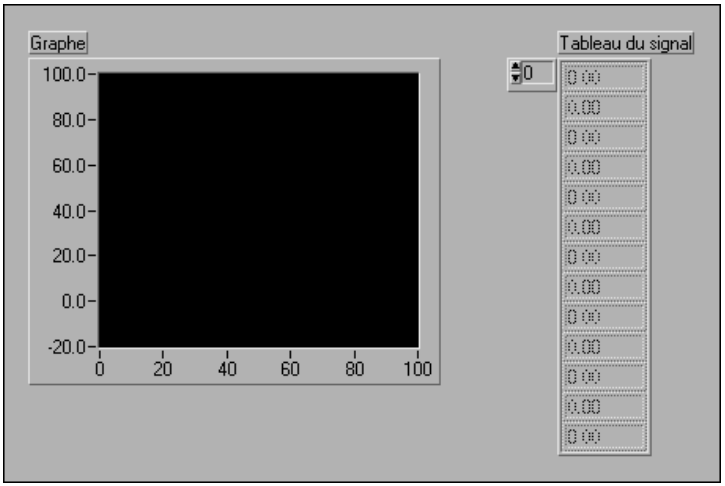
Exercice 6-4. Ecrire dans un fichier tableur

Votre objectif est de modifier un VI existant pour utiliser une fonction "E/S sur fichiers", afin de pouvoir enregistrer les données dans un nouveau fichier au format ASCII. Vous pourrez accéder ultérieurement à ce fichier à partir d'une application de type tableur.

Face-avant

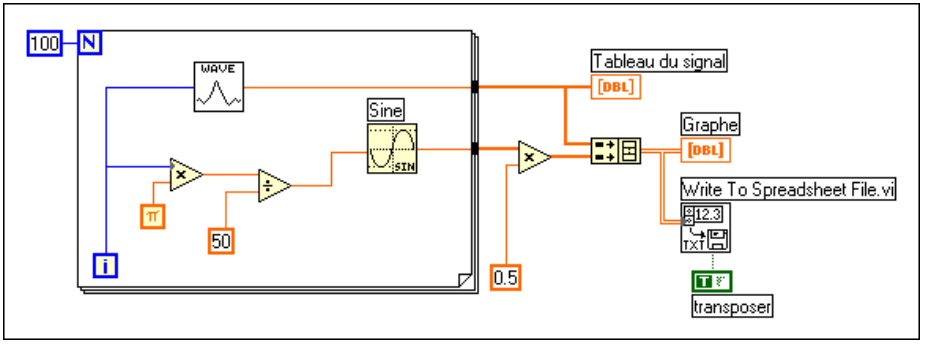
1. Ouvrez le VI `Tableaux et graphe.vi` que vous avez construit dans l'exercice 5-1. Comme vous le savez, ce VI génère deux tableaux de données et les représente sur un graphe. Vous devez modifier ce VI

pour écrire les deux tableaux dans un même fichier, où chaque colonne contient un tableau de données.



Diagramme

2. Ouvrez le diagramme du VI `Tableaux et graphe.vi` et modifiez-e en ajoutant les fonctions du diagramme qui se trouvent en bas à droite de l'illustration suivante.



Le VI "Ecrire dans un fichier tableur" (Write To Spreadsheet File.vi) (**Fonctions»E/S sur fichiers**) convertit le tableau 2D en une chaîne de caractères au format tableur et l'écrit sur un fichier. Si vous n'avez pas précisé le chemin, une boîte de dialogue de fichier apparaît, vous invitant à entrer un nom de fichier. Le VI "Ecrire dans un fichier tableur" (Write To Spreadsheet File.vi) écrit un tableau 1D ou 2D dans le fichier. Comme vous possédez un tableau de données 2D dans cet exemple, vous ne devez pas

câbler l'entrée 1D. Avec ce VI, vous pouvez utiliser un séparateur de type tableur ou des séparateurs de type chaîne de caractères, comme des tabulations ou des virgules dans vos données.

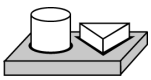
Constante booléenne (**Fonctions»Booléen**) contrôle la transposition ou non du tableau 2D avant de l'écrire dans le fichier. Pour faire passer la valeur sur VRAI (TRUE), cliquez sur la constante avec l'outil Doigt. Dans cet exemple, les données doivent être transposées parce que le tableau de données est organisé suivant les rangées (ainsi, chaque rangée d'un tableau à deux dimensions est un tableau de données). Comme chaque colonne du fichier tableur contient un tableau de données, le tableau 2D doit d'abord être transposé.

3. Revenez à la face-avant et exécutez le VI. Après la génération des tableaux de données, une boîte de dialogue de fichier vous invite à entrer le nom du nouveau fichier à créer. Entrez un nom de fichier et cliquez sur **OK**.

Avertissement ***N'essayez pas d'écrire des données dans les bibliothèques de VIs, par exemple*** `mywork.llb`***. Vous risqueriez d'écraser votre bibliothèque et de perdre votre travail précédent.***

4. Enregistrez le VI, nommez-le `Tableaux de signaux dans un fichier.vi`, puis fermez le VI.
5. Vous pouvez désormais utiliser un tableur ou un éditeur de texte pour ouvrir et lire le fichier que vous venez de créer. Vous devez voir deux colonnes de 100 éléments.

Dans cet exemple, les données ne sont ni converties ni écrites dans un fichier tant que les tableaux de données ne sont pas acquis en totalité. Si vous obtenez des buffers de données importants ou si vous souhaitez écrire les valeurs des données sur un disque à mesure que les buffers sont générés, vous devez utiliser un VI d'E/S sur fichiers différent.



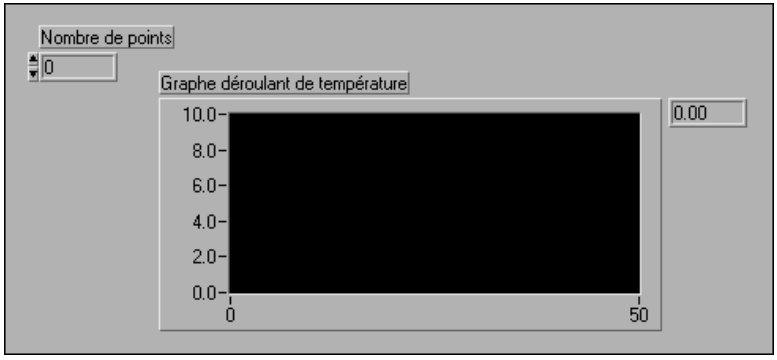
Fin de l'exercice 6-4.

Exercice 6-5. Ajouter des données à un fichier

Votre objectif est de créer un VI pour ajouter des données de température à un fichier en format ASCII. Ce VI utilise une boucle For pour générer des valeurs de température et pour les enregistrer dans un fichier. Lors de chaque itération, vous convertirez les données en une chaîne de caractères, ajouterez une virgule comme caractère de délimitation, et ajouterez la chaîne de caractères à un fichier.

Face-avant

1. Ouvrez une nouvelle face-avant et placez les objets comme indiqué dans l'illustration suivante.

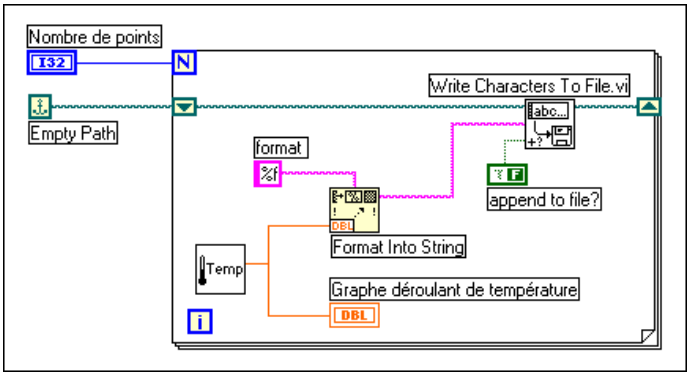


La face-avant contient une commande numérique et un graphe déroulant. Sélectionnez **Visualiser»Afficheur numérique**. La commande `nb de points` précise le nombre des valeurs de température à obtenir et à écrire dans le fichier. Le graphe déroulant représente la courbe de température. Remettez à l'échelle l'axe des Y du graphe déroulant pour la gamme 70,0 à 90,0. Remettez à l'échelle l'axe des X pour la gamme 0 à 20.

2. Ouvrez le menu local sur la commande numérique `nb de points` et choisissez **Représentation»I32**.

Diagramme

3. Ouvrez le diagramme.



4. Ajoutez la boucle For et agrandissez-la. Ce VI génère le nombre de valeurs de température précisé par la commande `nb de points`.
5. Ajoutez un registre à décalage à la boucle en ouvrant le menu local sur la bordure de la boucle. Ce registre à décalage contient le nom de chemin pour le fichier.
6. Terminez de câbler les objets.

Constante Chemin vide (**Fonctions»E/S sur fichiers»Constantes de fichiers**). La fonction “Chemin vide” initialise le registre à décalage afin que, la première fois que vous essayez d’écrire une valeur sur un fichier, le chemin soit vide. Une boîte de dialogue de fichier vous invite à entrer un nom de fichier.

Le VI “Thermomètre numérique” (Digital Thermometer.vi) retourne une mesure de température simulée provenant d’un capteur de température.

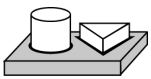
La fonction “Formater en chaîne de caractères” (**Fonctions»Chaîne de caractères**) convertit la mesure de température (un nombre) en une chaîne de caractères et la concatène avec la virgule.

Constante Chaîne de caractères (**Fonctions»Chaîne de caractères**). Cette chaîne de format spécifie que vous souhaitez convertir un nombre en une chaîne de format fractionnaire et faire suivre la chaîne de caractères par une virgule.

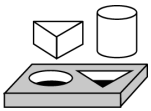
Le VI “Ecrire des caractères dans un fichier” (Write Characters To File.vi) (**Fonctions»E/S sur fichiers)** écrit une chaîne de caractères dans un fichier.

La constante booléenne (**Fonctions»Booléen**) positionne l'entrée `ajouter au fichier ?` du VI "Ecrire des caractères dans un fichier" (Write Characters To File.vi) sur Vrai (TRUE) afin que les nouvelles valeurs de température soient ajoutées au fichier sélectionné lors de l'itération de la boucle. Cliquez avec l'outil Doigt sur la constante pour mettre sa valeur sur Vrai (TRUE).

7. Revenez à la face-avant et exécutez le VI avec la commande `nb de points` réglée sur la valeur 20. Une boîte de dialogue de fichier vous invite à entrer un nom de fichier. Une fois que vous avez entré un nom de fichier, le VI commence à écrire les valeurs de température sur ce fichier à mesure que chaque point est généré.
8. Enregistrez le VI sous le nom `Ecrire la température dans un fichier.vi` dans le répertoire `LabVIEW\Activity`.
9. Utilisez un traitement de texte quelconque tel que Write pour Windows, Teach Text pour Macintosh, ou un éditeur de texte dans UNIX pour ouvrir ce fichier de données et pour afficher son contenu. Vous devez avoir un fichier contenant vingt valeurs de données (avec une précision au millième, soit trois chiffres après la virgule) séparées par des points.



Fin de l'exercice 6-5.

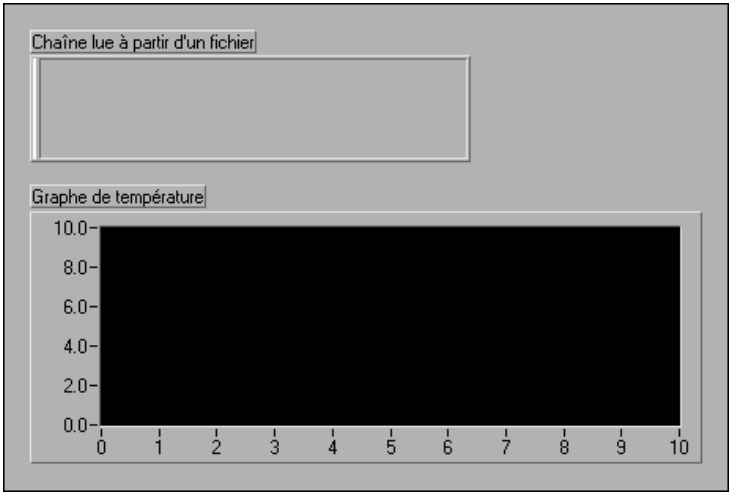


Exercice 6-6. Lire les données dans un fichier

Votre objectif est de créer un VI qui lit le fichier de données que vous avez écrit dans l'exemple précédent. Ce VI affiche ensuite les données dans un graphe. Vous devez lire les données dans le même format que celui utilisé pour enregistrer les données. Ainsi, comme vous avez enregistré les données dans un format ASCII en utilisant des types de données chaîne de caractères, vous devez lire les données comme des chaînes de caractères avec l'un des VIs d'E/S sur fichiers.

Face-avant

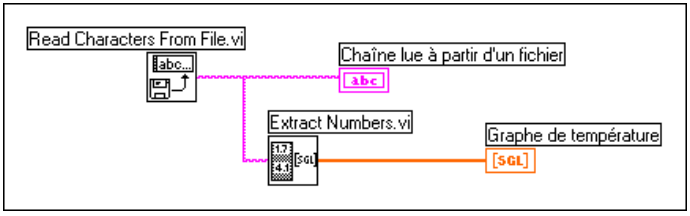
1. Ouvrez une nouvelle face-avant et construisez celle de l'illustration suivante.



Le face-avant contient un indicateur chaîne de caractères et un graphe. L'indicateur "Lire une chaîne de caractères dans un fichier" affiche les données de la température séparées par des virgules dans le fichier créé dans l'exercice précédent. Le graphe représente la courbe de la température.

Diagramme

2. Construisez le diagramme comme indiqué dans l'illustration suivante.



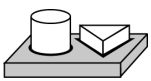
Le VI "Lire des caractères dans un fichier" (Read Characters From File.vi) (**Fonctions»E/S sur fichiers**) lit les données d'un fichier et génère les informations sous la forme d'une chaîne de caractères. Si aucun nom de chemin n'est précisé, une boîte de dialogue de fichier vous invite à entrer

un nom de fichier. Dans cet exemple, vous n'avez pas besoin de déterminer le nombre de caractères à lire car il y a moins de 512 caractères dans le fichier (valeur par défaut).

Vous devez connaître la façon dont les données ont été enregistrées dans un fichier afin de pouvoir les lire. Si vous ne connaissez pas la longueur d'un fichier, vous pouvez utiliser le VI "Lire des caractères dans un fichier" (Read Characters From File.vi) pour déterminer le nombre de caractères à lire.

Le VI "Extraction de nombres" (Extract Numbers.vi) (`Examples\General\Strings.llb`) prend une chaîne de caractères ASCII contenant des nombres séparés par des virgules, par des retours à la ligne, ou par d'autres caractères non numériques et les convertit en un tableau de valeurs numériques.

3. Revenez à la face-avant et exécutez le VI. Sélectionnez le fichier de données que vous avez créé précédemment lorsque la boîte de dialogue de fichier vous y invite. Vous devez obtenir les mêmes valeurs dans le graphe que celles affichées dans le VI d'exemple "Ecrire la température dans un fichier" (Write Temperature to File.vi).
4. Enregistrez le VI, nommez-le `Température d'un fichier.vi`, puis fermez le VI.



Fin de l'exercice 6-6.

Utilisation des fonctions "E/S sur fichiers"

Parfois, les fonctions "E/S sur fichiers" de niveau supérieur ne fournissent pas les fonctionnalités nécessaires pour enregistrer des données sur le disque. Dans ce cas, vous devez utiliser les fonctions fournies de la sous-palette **Fonctions»E/S sur fichiers»Avancé**.

Spécification d'un fichier

Il y a deux façons de spécifier un fichier : vous pouvez le programmer ou utiliser une boîte de dialogue. Dans la méthode programmation, vous fournissez le nom de fichier et le chemin du VI.

(Windows) Un chemin est constitué du nom du lecteur (par exemple, `C`), suivi par le caractère deux-points (:), suivi par des noms de répertoire séparés par des barres obliques inverses, suivi par le nom de fichier. Par

exemple, vous devez écrire `C:\DATADIR\TEST1` pour un fichier nommé `TEST1` dans le répertoire `DATADIR` sur le lecteur `C`.

(Macintosh) Un chemin est constitué du nom du lecteur, suivi par le caractère deux-points (:), suivi par des noms de dossier séparés par le caractère deux-points, suivi par le nom de fichier. Par exemple, la chaîne `HardDrive:DataFolder:Test1` fait référence à un fichier nommé `Test1` dans le dossier `DataFolder` sur le volume nommé `HardDrive`.

(UNIX) Un chemin est constitué des noms des répertoires séparés par des barres obliques, suivis par le nom du fichier. Par exemple, la chaîne `/usr/datadirectory/test1` fait référence à un fichier nommé `test1` dans le répertoire `/usr/datadirectory`.

(Toutes les plates-formes) En utilisant la méthode interactive, la fonction “Boîte de dialogue de fichier” affiche une boîte de dialogue permettant de rechercher de façon interactive un répertoire et d’entrer un nom de fichier.

Chemins et identificateurs

Un chemin est un type de données en G qui identifie les fichiers. Vous pouvez entrer ou afficher le chemin d’un fichier en utilisant une syntaxe normale pour une plate-forme donnée avec la commande de chemin et l’indicateur de chemin. Dans de nombreux aspects, la commande et l’indicateur de chemin fonctionnent comme une commande ou un indicateur de chaîne de caractères, mis à part le fait que le G formate le chemin de façon appropriée pour toutes les plates-formes supportées.

Un identificateur consiste en un type de données en G qui identifie les fichiers ouverts. Lorsque vous ouvrez un fichier, le G retourne un identificateur associé au fichier. Toutes les opérations effectuées sur des fichiers ouverts utilisent les identificateurs de fichier pour identifier chaque fichier. Un identificateur n’est valide que pour la période durant laquelle le fichier est ouvert. Si vous fermez le fichier, le G dissocie l’identificateur du fichier. Si vous ouvrez à nouveau le fichier, le nouvel identificateur peut être différent de l’identificateur utilisé précédemment.

En plus d’associer une opération à un fichier, le G se souvient des informations relatives à chaque identificateur, comme l’emplacement courant pour la lecture du fichier et le niveau d’accès au fichier autorisé pour les autres utilisateurs, de sorte que vous pouvez avoir des opérations concurrentes mais indépendantes sur un même fichier. Si vous ouvrez plusieurs fois un fichier, chaque ouverture retourne un identificateur différent.

Exemples d'E/S sur fichiers

Vous pouvez utiliser les exemples suivants pour voir comment utiliser les fonctions "E/S sur fichiers" avec les techniques de gestion d'erreur adéquates :

Le VI "Ecrire dans un fichier texte" (Write to Text File.vi) (dans `Examples\File\smplfile.llb`) écrit un fichier texte ASCII contenant des données horodatées.

Le VI "Lire dans un fichier texte" (Read from Text File.vi) (dans `Examples\File\smplfile.llb`) lit un fichier texte ASCII contenant des données horodatées.

Fichiers d'enregistrement de données

Les exemples de ce chapitre illustrent des méthodes simples pour gérer des fichiers contenant des données enregistrées comme une séquence de caractères ASCII. Cette approche est courante lors de création de fichiers lus par d'autres logiciels, un tableur par exemple. Le G possède un autre format de fichier, appelé *fichier d'enregistrement de données*. Un fichier d'enregistrement de données stocke les données comme une séquence d'enregistrements d'un type de données unique et arbitraire, que vous déterminez lors de la création du fichier. Le G indexe les données dans un fichier d'enregistrement de données, exprimées dans ces enregistrements. Tous les enregistrements d'un fichier d'enregistrement de données doivent être du même type, mais ce type peut cependant être complexe. Par exemple, vous pouvez définir un enregistrement de sorte qu'il contienne un cluster avec une chaîne de caractères, un nombre et un tableau.

Si vous allez récupérer les données avec un VI, vous ne souhaiterez peut-être pas écrire les données dans des fichiers ASCII, car la conversion des données en chaînes de caractères (et vice versa) peut prendre beaucoup de temps. Par exemple, la conversion d'un tableau à deux dimensions en une chaîne de caractères dans un format tableur avec en-têtes et horodatages s'avère une opération compliquée. Si vous n'avez pas besoin d'avoir les données enregistrées dans un format pouvant être lu par d'autres applications, vous avez la possibilité d'écrire les données dans un fichier d'enregistrement de données. Sous cette forme, l'écriture des données dans un fichier requiert peu de manipulation, permettant ainsi une écriture et une lecture plus rapides. Ceci simplifie également la récupération des données, car vous pouvez lire les blocs de données d'origine comme un enregistrement sans avoir à connaître le nombre d'octets de données

contenus dans les enregistrements. Le G enregistre la quantité de données pour chaque enregistrement d'un fichier d'enregistrement de données.

Le VI "Exemple d'écriture dans un fichier d'enregistrement de données" (Write Datalog File Example.vi) (dans `Examples\File\Datalog.llb`) crée un nouveau fichier d'enregistrement de données et écrit le nombre d'enregistrements précisé dans le fichier. Chaque enregistrement est un cluster contenant une chaîne de caractères et un tableau de nombres simple précision.

Pour lire un fichier d'enregistrement de données, vous devez utiliser le même type de données que celui utilisé lors de l'écriture. Le VI "Exemple de lecture dans un fichier d'enregistrement de données" (Read Datalog File Example.vi) (dans `Examples\File\Datalog.llb`) lit un fichier d'enregistrement de données créé par le VI "Exemple d'écriture dans un fichier d'enregistrement de données" (Write Datalog File Example.vi), un enregistrement à la fois. La lecture de l'enregistrement comprend un cluster contenant une chaîne de caractères et un tableau de nombres simple précision.

Partie II

Interfaces d’E/S

Cette section contient des informations de base sur les interfaces via lesquelles vous pouvez entrer et sortir des données : acquisition de données, GPIB, série et VXI. Reportez-vous au *Manuel de base d’acquisition de données LabVIEW* pour des informations sur l’acquisition de données en temps réel. VISA (Virtual Instrument Software Architecture : architecture logicielle d’instrument virtuel) est une bibliothèque logicielle unique qui sert d’interface avec les instruments GPIB, série et VXI. Les applications LabVIEW développées spécialement pour un instrument spécifique sont appelées drivers d’instruments. National Instruments fournit plusieurs drivers d’instruments utilisant la bibliothèque VISA, mais vous pouvez aussi construire vos propres drivers d’instruments.

La Partie II, *Interfaces d’E/S*, contient les chapitres suivants.

- Le chapitre 7, *Initiation aux drivers d’instruments LabVIEW*, explique comment créer et utiliser des drivers d’instruments National Instruments.
- Le chapitre 8, *Tutorial VISA de LabVIEW*, vous montre comment implémenter des applications VISA courantes en utilisant une communication basée message et basée registre, ainsi que des événements et le verrouillage.
- Le chapitre 9, *Introduction aux fonctions GPIB de LabVIEW*, explique comment fonctionne le GPIB et la différence entre les interfaces IEEE 488 et IEEE 488.2.
- Le chapitre 10, *VIs de port série*, décrit les VIs pour la communication de port série et explique les facteurs importants affectant la communication série.

7

Initiation aux drivers d'instruments LabVIEW

Ce chapitre commence par décrire la façon d'installer et d'utiliser les drivers d'instruments à partir de la bibliothèque des drivers d'instruments, puis vous explique comment créer votre propre driver d'instrument. Ce chapitre vous guide et vous assiste dans l'utilisation des techniques classiques de vérification de communication avec votre instrument, dans le développement d'une application utilisant des drivers d'instruments, et dans la création d'un driver d'instrument.

Qu'est-ce qu'un driver d'instrument LabVIEW ?

Un driver d'instrument est un ensemble de VIs LabVIEW qui communiquent avec un instrument grâce à des fonctions d'E/S VISA standard de LabVIEW. Chaque VI correspond à une opération de programmation, comme la configuration, la lecture, l'écriture, ou le déclenchement. Grâce aux drivers d'instruments LabVIEW, vous n'avez plus besoin de connaître les commandes de programmation complexes de bas niveau de chaque instrument.

La bibliothèque de drivers d'instruments LabVIEW possède des drivers d'instruments pour plusieurs instruments programmables utilisant l'interface série, GPIB ou VXI. Vous pouvez utiliser un driver d'instrument tel quel. Cependant, sachez que les drivers d'instruments sont distribués avec leur code source (diagramme) ; vous pouvez donc les personnaliser pour votre application spécifique si besoin est.

Où obtenir des drivers d'instruments ?

Vous pouvez installer les drivers d'instruments à partir d'un CD de driver d'instrument ou les télécharger depuis le site web de National Instruments. Pour obtenir les drivers d'instruments les plus récents, utilisez un téléphone à touches pour appeler le système de fax automatisé de National Instruments, Fax-sur-demande, au 512 418 1111 ou 800 329 7177. Vous

pouvez également charger les drivers avec le réseau de drivers d'instruments sur le web. Pour accéder à ce réseau, connectez-vous à `http://www.natinst.com/idnet`.

S'il n'existe pas de driver d'instrument pour votre instrument, vous pouvez :

1. Essayer d'utiliser un driver d'un instrument similaire. Les instruments similaires provenant d'un même fabricant possèdent souvent des ensembles de commandes semblables, si ce n'est identiques.
2. Créer un driver d'instrument conformément aux instructions de la section *Développement d'un driver d'instrument LabVIEW rapide et simple* de ce chapitre.
3. Développer un driver complet en utilisant toutes les fonctions de votre instrument. Pour développer un driver de qualité National Instruments, vous pouvez récupérer la Note d'application 006, intitulée *Developing a LabVIEW instrument driver (Développement d'un driver d'instrument LabVIEW)*, à partir de notre site web. Cette note d'application vous aide à développer un driver d'instrument complet.

Où dois-je installer mon driver d'instrument LabVIEW ?

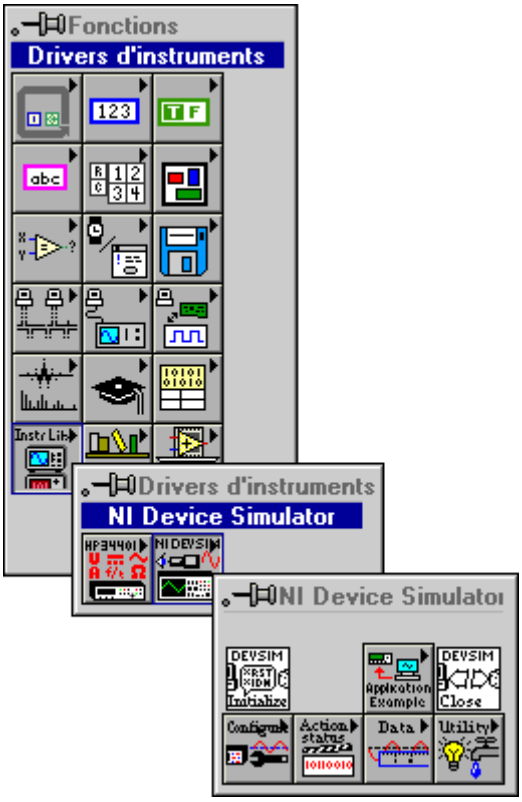
Les drivers d'instruments doivent être installés comme un sous-répertoire du répertoire `LabVIEW/instr.lib`. Par exemple, le driver d'instrument HP34401A, inclus avec LabVIEW, est installé dans le répertoire suivant :

```
Labview/instr.lib/hp34401a
```

Dans ce répertoire, vous trouverez les fichiers de menu et les bibliothèques de VIs qui composent un driver d'instrument. Les fichiers de menu vous permettent de voir vos VIs drivers d'instruments dans la palette **Fonctions**. Les bibliothèques de VIs contiennent les VIs représentant les drivers d'instruments.

Comment accéder aux VIs drivers d'instruments ?

Vous pouvez trouver les VIs drivers d'instruments en bas de la palette **Fonctions**, dans la sous-palette **Drivers d'instruments**.



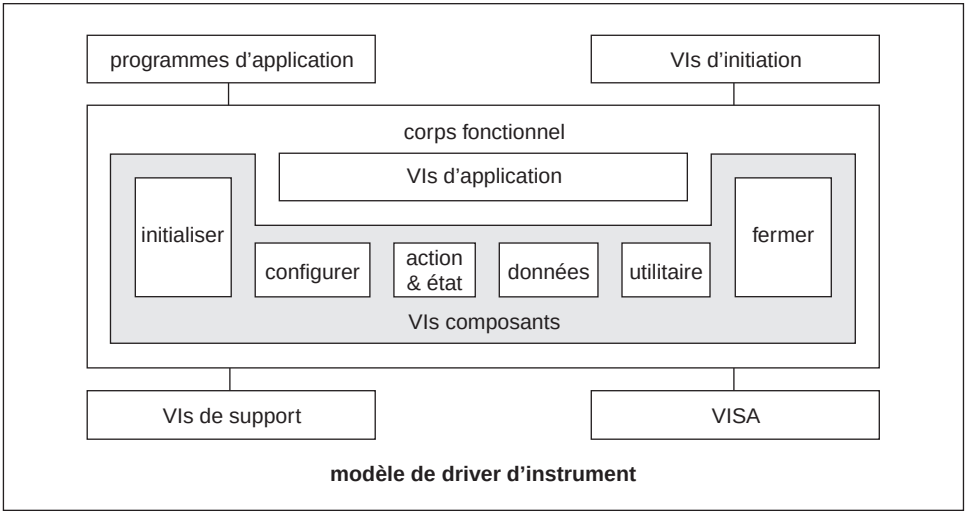
De nombreux drivers d'instruments possèdent des palettes de menus, dotées des composantes suivantes :

- VI "Initialiser" (Initialize.vi)
- VI "Fermer" (Close.vi)
- Sous-palette Exemple d'application
- Sous-palette Configuration
- Sous-palette Action/Etat
- Sous-palette Données
- Sous-palette Utility

Vous pouvez également accéder aux VIs drivers d'instruments avec l'option **Sélectionner un VI** dans la palette **Fonctions**. Pour afficher la hiérarchie entière du driver d'instrument, vous pouvez ouvrir le VI "Arbre de VIs" (VI Tree.vi). Il s'agit d'un VI non exécutable, conçu pour montrer la structure fonctionnelle d'un driver d'instrument.

Structure d'un driver d'instrument

La figure suivante montre l'organisation d'un driver d'instrument standard. Une fois que vous aurez compris ce modèle, vous verrez qu'il s'applique à de nombreux drivers d'instruments.



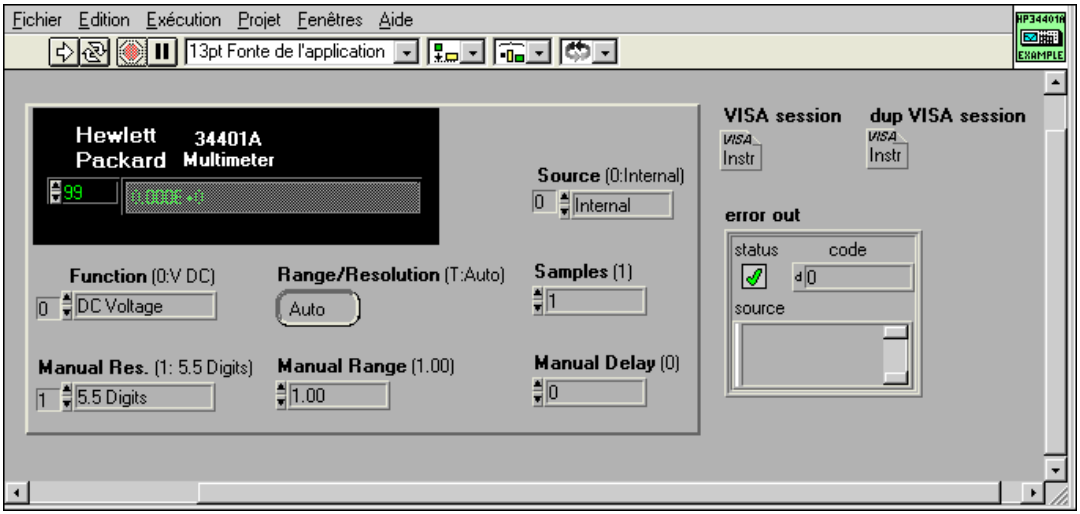
modèle de driver d'instrument

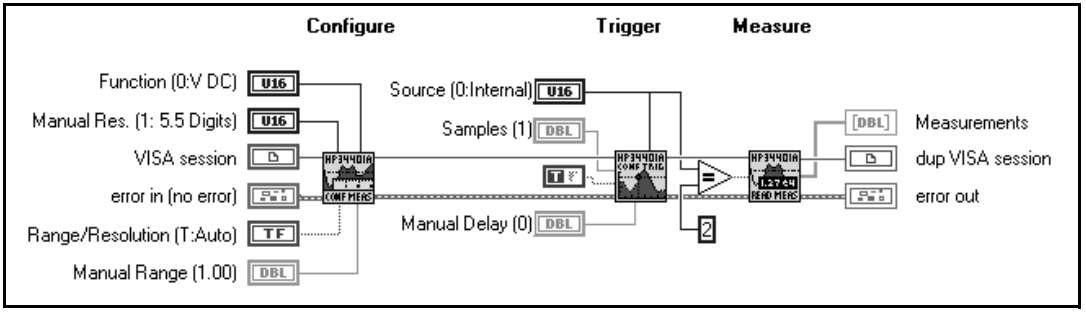
Les VIs Initiation (Getting Started.vi) sont des VIs d'application simple que vous pouvez utiliser sans modification. Exécutez ce VI pour vérifier la communication avec votre instrument. Typiquement, vous devez seulement changer l'adresse de l'instrument dans la face-avant avant d'exécuter le VI. Il y a cependant quelques VIs qui nécessitent également que vous spécifiiez le nom de la ressource VISA (par exemple, GPIB::2). Pour plus d'informations sur les noms de ressources VISA, consultez le chapitre 8, *Tutorial VISA de LabVIEW*. Le VI Initiation (Getting Started.vi) comprend généralement trois sous-VIs : le VI Initialiser (Initialize.vi), un VI d'application et le VI Fermer (Close.vi).

Les VIs d'application sont des exemples de regroupement de fonctions de bas niveau pour exécuter une opération avec un instrument contrôlée par programme. Par exemple, les VIs d'application peuvent inclure des VIs

pour contrôler la plupart des configurations d'instruments et de mesures communément utilisées. Ces VIs servent d'exemple de code pour exécuter des opérations classiques comme la configuration, le déclenchement et la prise d'une mesure par un instrument.

Comme les VIs d'application sont des VIs standard avec icônes et connecteurs, vous pouvez les appeler à partir de toute application de haut niveau lorsque vous souhaitez une interface unique et orientée mesures avec le driver. Pour de nombreux utilisateurs, les VIs d'application sont les seuls VIs drivers d'instruments nécessaires au contrôle d'instrument. Le VI d'exemple HP34401, représenté dans la figure suivante, montre la face-avant et le diagramme d'un VI d'application.





Le VI *d'initialisation*, premier VI driver d'instrument appelé, établit la communication avec l'instrument. Il peut également réaliser toutes les actions nécessaires pour placer l'instrument dans son état de démarrage par

défaut ou dans d'autres états spécifiques. Généralement, ce VI ne doit être appelé qu'une fois au début de votre programme d'application.

Les *VIs de configuration* sont un regroupement de routines logicielles qui configurent l'instrument pour effectuer l'opération désirée. Il peut y avoir de nombreux *VIs de configuration*, selon l'instrument. Une fois que ces VIs sont appelés, l'instrument est prêt à prendre des mesures ou à stimuler un système.

La catégorie *action/état* contient deux types de VIs. Les VIs *d'action* initialisent ou terminent des opérations d'essai et de mesure. Ces opérations peuvent inclure l'armement du système de déclenchement ou la génération d'un stimulus. Ces VIs sont différents des *VIs de configuration* parce qu'ils ne modifient pas les paramètres de l'instrument : ils ordonnent simplement à l'instrument de réaliser une action conformément à sa configuration actuelle. Les *VIs d'état* obtiennent l'état actuel de l'instrument ou l'état des opérations en cours.

Les *VIs de données* transfèrent les données en provenance ou en direction d'un instrument. Par exemple, il existe des VIs lisant une valeur ou un signal mesuré par un instrument de mesure, des VIs déchargeant des signaux ou des patterns numériques à partir d'un instrument source.

Les *VIs utilitaires* réalisent une variété d'opérations auxiliaires à la plupart des VIs drivers d'instruments souvent utilisés. Ces VIs incluent la majorité des VIs de modèle de driver d'instrument tels que les VIs concernant la remise à zéro, l'auto-test, la révision, la recherche d'erreur et les messages d'erreur. Il peut s'agir aussi d'autres VIs drivers d'instruments personnalisés effectuant des opérations comme l'étalonnage ou l'enregistrement et le rappel de configurations.

Le *VI de fermeture* termine la connexion logicielle sur l'instrument et libère les ressources système. Généralement, ce VI ne doit être appelé qu'une fois, à la fin de votre application ou lorsque vous terminez la communication avec votre instrument. Vous devez vous assurer que pour chaque appel du VI d'initialisation réussi, correspond un VI de fermeture, pour éviter de réserver inutilement des ressources mémoire.

☞ **Remarque** ***Les fonctions d'application n'appellent pas les VIs d'initialisation et de fermeture. Pour exécuter une fonction d'application, vous devez d'abord exécuter le VI d'initialisation. Le VI d'initiation effectue l'initialisation et la fermeture.***

Obtenir l'aide pour vos VIs drivers d'instruments

Les drivers d'instruments LabVIEW sont documentés dans la fenêtre **Aide** de LabVIEW. Il y a une description **Aide** pour chaque VI driver d'instrument ainsi que pour chaque commande de la face-avant. Pour afficher la fenêtre **Aide**, choisissez **Visualiser l'aide** dans le menu **Aide**. Pour afficher l'aide du VI, placez le curseur sur l'icône du VI. Pour afficher l'aide des commandes de la face-avant, placez le curseur sur la commande désirée. Si vous ne pouvez pas voir la description entière dans la fenêtre **Aide**, vous pouvez obtenir de l'aide relative à la commande ou à l'indicateur en sélectionnant **Opérations sur les données»Description...** dans le menu local de la commande ou de l'indicateur.

Exécution du VI d'initiation (Getting Started VI) de façon interactive

(Sélection de l'adresse GPIB, du port série et de l'adresse logique)

Pour vérifier la communication avec votre instrument et pour tester une opération d'instrument typique contrôlée par programmation, vous devez d'abord ouvrir le VI d'initiation. Parcourez chaque commande et définissez-les de façon appropriée. Généralement, à l'exception du champ d'adresse, les valeurs par défaut de la plupart des commandes sont suffisantes pour votre première exécution. Vous devrez définir l'adresse de façon appropriée. Si vous ne connaissez pas l'adresse de votre instrument, reportez-vous à l'Assistant d'instrument pour de l'aide. Après l'exécution du VI, vérifiez que des données cohérentes ont été retournées et qu'une erreur n'a pas été rapportée dans le cluster d'erreur. La plupart des causes d'échec courantes du VI d'initiation sont les suivantes :

1. NI-VISA n'est pas installé. Si vous n'avez pas choisi ceci comme option durant votre installation de LabVIEW, vous devez l'installer avant la nouvelle exécution du VI.
2. L'adresse d'instrument était incorrecte. Le VI d'initiation requiert que vous spécifiiez l'adresse correcte pour votre instrument. Si vous n'êtes pas certain de l'adresse de votre instrument, exécutez l'Assistant d'instrument ou la fonction "Rechercher une ressource". Si vous n'êtes pas familier avec la syntaxe de la chaîne de caractères d'adresse, reportez-vous au chapitre 8, *Tutorial VISA de LabVIEW*, pour de l'aide. Pour accéder à l'Assistant d'instrument, sélectionnez **Assistant**

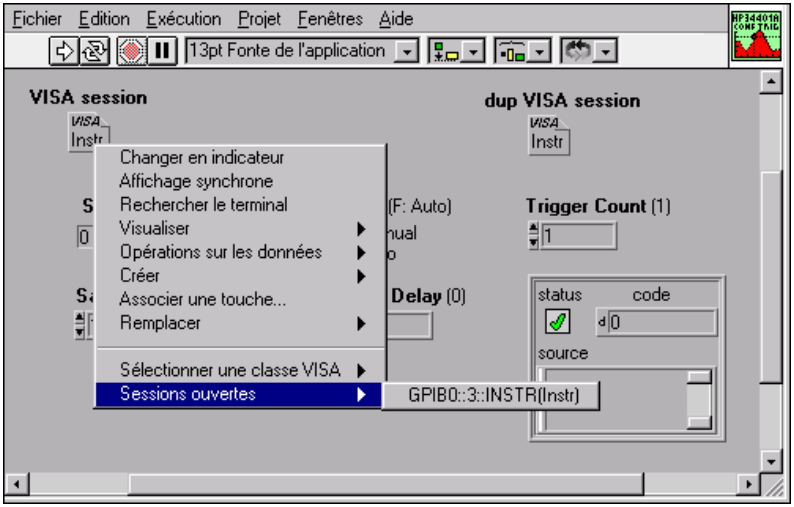
Solutions dans la boîte de dialogue LabVIEW. Lorsque le système vous y invite, sélectionnez **l'Assistant d'instrument**.

3. Le driver d'instrument ne supporte pas le modèle exact que vous utilisez. Il vous faut alors vérifier que le driver d'instrument supporte le modèle d'instrument que vous utilisez.

Après avoir vérifié la communication de base avec votre instrument, vous souhaiterez probablement personnaliser le contrôle d'instrument selon vos besoins. Si les exigences de votre application sont similaires au VI d'initiation, la façon la plus simple de créer un VI personnalisé est d'enregistrer une copie de ce VI en sélectionnant **Enregistrer sous...** dans le menu **Fichier**. Vous pouvez changer les valeurs par défaut de la face-avant en sélectionnant **Prendre les valeurs actuelles par défaut** dans le menu **Exécution**. Les modifications du diagramme peuvent inclure le changement des constantes câblées au VI d'application ou aux autres sous-VIs. Comme mentionné précédemment, le diagramme du VI d'initiation comprend généralement trois VIs : le VI d'initialisation, un VI de fonction d'application et le VI de fermeture.

Test interactif des VIs de composantes

De nombreux utilisateurs souhaitent tester les VIs de composantes de façon interactive avant de les inclure dans leur application. Ceci les aide à sélectionner les paramètres de configuration d'instrument appropriés. Pour exécuter les VIs de composantes dans leur face-avant, vous devez d'abord exécuter le VI d'initialisation. Pour les VIs suivants, vous devez d'abord ouvrir le menu local de la commande Session VISA et sélectionner votre nom de ressource dans le sous-menu **Sessions ouvertes...**, comme indiqué ci-dessous :



Une fois que vous avez sélectionné votre ressource, vous pouvez exécuter interactivement le VI de composante dans la face-avant, plusieurs fois, sans réinitialiser la sélection de la ressource.

En général, vous devez exécuter des VIs dans l'ordre dans lequel vous souhaitez les appeler dans votre application. Exécutez d'abord le VI d'initialisation, suivi par un ou plusieurs VIs de configuration. Si vous utilisez un déclenchement pour votre mesure, vous aurez peut-être besoin d'appeler un VI d'action pour armer le déclenchement. Les appels aux VIs de données recueillent ensuite la ou les valeurs mesurées. Lorsque vous terminez de tester les VIs de composantes de driver d'instrument, vous devez exécuter le VI de fermeture pour libérer les ressources.

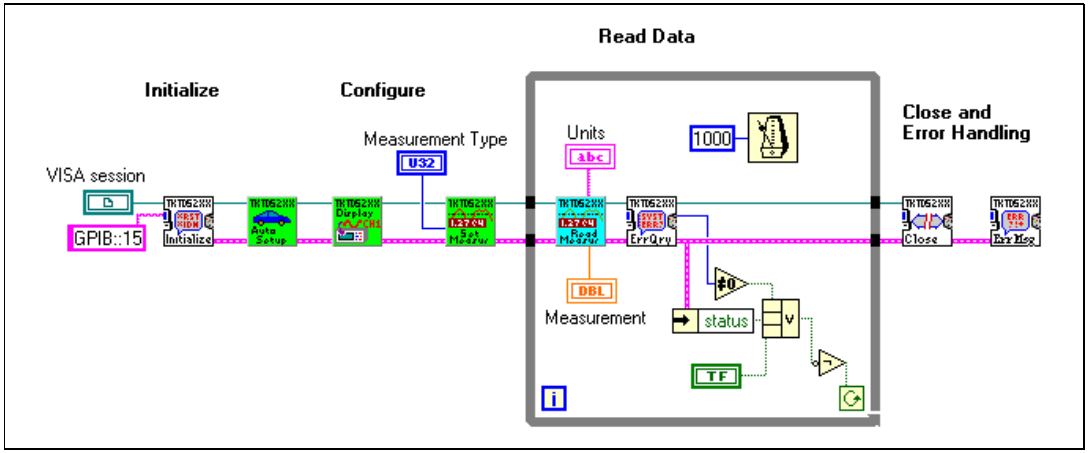
Construction de votre application

Après avoir sélectionné les VIs de composantes nécessaires et déterminé leur ordre d'exécution, vous pouvez construire votre application. Si l'ordre d'exécution est similaire au VI d'application, vous pouvez modifier le diagramme du VI d'application. Si votre application diffère de façon significative du VI d'application, vous devez construire votre propre VI.

Vous devez placer les VIs sur votre diagramme dans l'ordre, puis les câbler ensemble en utilisant la session VISA et les paramètres de cluster d'erreur. Il n'est pas nécessaire de câbler toutes les entrées pour tous les VIs de composantes. Si les valeurs par défaut sont suffisantes pour votre application, vous n'avez pas besoin de câbler les terminaux d'entrée. Pour

les entrées importantes, vous trouverez peut-être judicieux de câbler quand même les valeurs par défaut pour documenter votre VI. Dans le cas de mesures répétées, vous devez placer vos VIs de mesure de données dans une boucle. Notez que si vous placez un VI de composante dans une boucle, vous devez désactiver l'indexation sur la session VISA et sur les fils de liaison du cluster d'erreur qui sortent de la boucle ou qui y rentrent.

Pour contrôler les erreurs, vous devez appeler le VI "Recherche d'erreur" (Error Query.vi) de façon périodique. Comme montré dans la figure suivante, un utilisateur peut utiliser un oscilloscope pour prendre des mesures de fréquence, toutes les secondes, et afficher ces mesures pour l'opérateur. Remarquez que la boucle se termine de trois façons possibles : l'opérateur arrête le VI grâce à une commande de la face-avant, ou bien une erreur en provenance de l'instrument est détectée par le VI "Recherche d'erreur" (Error Query.vi), ou encore une erreur se produit avec l'interface d'E/S VISA. Si une erreur survient dans la boucle, le VI "Message d'erreur" (Error Message.vi) affiche alors un message automatique à l'opérateur. Le VI "Message d'erreur" est similaire au VI "Gestionnaire général d'erreurs" (General Error Handler.vi) de LabVIEW, excepté que les erreurs supplémentaires spécifiques aux instruments peuvent être rapportées. Vous devez utiliser le VI "Message d'erreur" après l'exécution de plusieurs VIs drivers d'instruments afin de reconnaître et d'afficher toutes les erreurs survenues.



Sujets connexes

VI "Contrôle de session VISA ouverte" (Open VISA Session Monitor.vi)

Le VI "Contrôle de session VISA ouverte" (Open VISA Session Monitor.vi) est pratique si vous participez à la mise au point (interactive ou contrôlée par programmation) de votre driver d'instrument. Lors de la mise au point, vous découvrirez peut-être que vous possédez de nombreuses sessions VISA ouvertes devant être fermées. Si vous ouvrez un nombre significatif de sessions VISA sans les fermer, vous diminuez les ressources mémoire disponibles. Pour fermer rapidement toutes les sessions ouvertes, vous pouvez exécuter le VI "Contrôle de session VISA ouverte" (Open VISA Session Monitor.vi) de la bibliothèque `labview/vi.lib/utility/visa.llb`. Vous pouvez également enregistrer votre travail, quitter LabVIEW, puis le relancer. Lorsque vous sortez de LabVIEW, vous fermez aussi toutes vos sessions VISA ouvertes.

Gestion d'erreurs

La gestion d'erreurs est importante dans les applications de contrôle d'instrument, car il y a en effet plusieurs sources d'erreurs potentielles.

- Les fonctions VISA peuvent retourner des erreurs si VISA, le logiciel ou le matériel sous-jacent n'est pas correctement installé. Par exemple, lors de la communication avec des instruments GPIB, NI-488.2 doit être installé correctement pour utiliser la carte contrôleur GPIB de National Instruments. De façon similaire, si la carte n'est pas installée ou si elle n'est pas correctement configurée, les VIs drivers d'instruments retournent une erreur. Ce type d'erreur peut être détecté avec le VI "Message d'erreur" (Error Message.vi) ou le VI "Gestionnaire général d'erreurs" (General Error Handler.vi) de LabVIEW.
- Les fonctions VISA peuvent retourner des erreurs si le périphérique auquel vous accédez ne répond pas aux commandes que vous avez envoyées. L'instrument n'est peut-être pas adressé correctement, ne fonctionne pas comme il faut, ou est incapable de comprendre les commandes envoyées. Ce type d'erreur peut être détecté avec le VI "Message d'erreur" (Error Message.vi) ou le VI "Gestionnaire général d'erreurs" (General Error Handler.vi) de LabVIEW.
- L'instrument retourne une erreur. Généralement, un instrument positionne un indicateur d'erreur pour des raisons diverses, que ce soit des commandes invalides, des paramètres dépassant la gamme ou des options matérielles manquantes. Ces erreurs d'instrument peuvent être

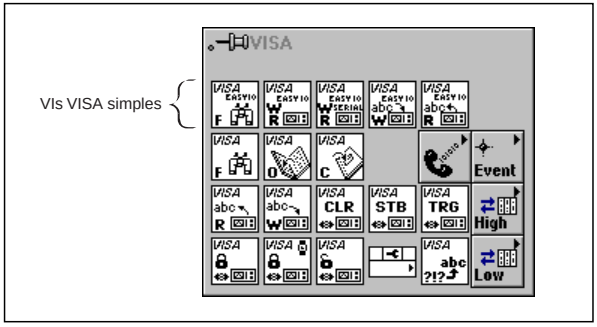
détectées en appelant le VI "Recherche d'erreur" (Error Query.vi) du driver d'instrument, suivi par le VI "Message d'erreur" (Error Message.vi).

Pour plus d'informations sur la gestion d'erreurs, consultez la section *Gestion d'erreurs avec VISA* au chapitre 8, *Tutorial VISA de LabVIEW*.

Test de communication avec votre instrument

Si vous avez des problèmes pour communiquer avec votre instrument à l'aide des VIs drivers d'instruments, utilisez les VIs d'E/S VISA simples (Easy VISA IO.vi) pour tester interactivement des lectures et des écritures simples, sans exécuter le VI "VISA Open" ni le VI "VISA Close". Par exemple, pour le VI "Easy VISA Write", vous devez seulement fournir le nom de la ressource et le message devant être envoyé à l'instrument. Les VIs d'E/S VISA simple (Easy VISA IO.vi) incluent les VIs suivants et sont représentés dans l'illustration suivante :

- VISA Find Resources
- VISA Write
- VISA Read
- VISA Write & Read
- VISA Serial Write & Read
- Register Write
- Register Read



Bien que ces VIs d'E/S VISA simple soient pratiques pour des tests, ils doivent être utilisés avec précaution pour le développement d'application. Pour chaque lecture ou écriture de l'instrument, ces VIs ouvrent et ferment une session VISA, ce qui peut ralentir votre application si celle-ci est appelée de façon répétée. Pour le développement d'application, il vaut donc mieux utiliser les fonctions VISA standard.

Développement d'un driver d'instrument LabVIEW rapide et simple

Bien que National Instruments continue de développer de nouveaux drivers d'instruments, nous ne possédons pas les ressources pour développer des drivers pour tous les instruments demandés. Vous vous trouverez peut-être dans une situation où vous aurez besoin d'une interface spécifique avec un instrument alors qu'aucun driver n'est disponible. Cette section décrit la façon de développer un driver d'instrument simple pour votre application.

Modification d'un driver existant

Avant de commencer, vérifiez s'il existe déjà un driver pour votre instrument, en vous reportant par exemple au site web du fabricant et au site web de National Instruments. Lorsque vous consultez les sites web, soyez à l'affût des drivers d'instruments qui supportent un instrument similaire au vôtre. Les instruments provenant de la même série de modèle possèdent souvent des jeux de commandes similaires. Dans la même lignée, les instruments SCPI de même fonctionnalité ont également des ensembles de commandes semblables. Obtenez ces drivers et évaluez le degré de similarité de l'ensemble de commandes avec celui de votre instrument. Pour des instruments provenant de la même série de modèle, vous devrez peut-être contacter le fabricant et lui demander des détails sur les différences entre les ensembles de commandes. Si vous comparez des instruments SCPI similaires, vous devrez comparer les commandes du driver d'instrument à celles figurant dans le manuel de programmation de votre instrument. Vous pourrez peut-être modifier un driver existant pour optimiser le code. Pour un driver qui sera utilisé par plusieurs utilisateurs, un VI de composante peut être utile pour modifier un paramètre qui n'est pas nécessaire à votre application. En général, vous ne devrez optimiser que les VIs appelés de façon répétée dans une boucle. Les VIs de configuration ne sont habituellement appelés qu'une seule fois, et ont peu d'effet sur la vitesse d'exécution de l'application.

La façon la plus simple de modifier un VI driver d'instrument est de le renommer en sélectionnant **Enregistrer sous...** dans le menu **Fichier**. Pour identifier le nouveau VI, vous devez changer le nom en modifiant le préfixe ou la description. Par exemple, si vous modifiez un driver d'instrument d'oscilloscope TDS Tektronix pour travailler avec un instrument différent, il est préférable d'adopter un nom approprié à votre instrument au lieu d'utiliser TKTDS7XX comme préfixe de VI. Après avoir modifié le nom, vous aurez à modifier les commandes du diagramme et de la face-avant. La plupart des changements apportés au diagramme sont liés aux fonctions de

chaînes de caractères. Si vous n'êtes pas familier avec ce type de fonctions, telles que "Piocher une ligne & Ajouter" ou "Sélectionner & Ajouter", reportez-vous à la *Référence en ligne* de LabVIEW pour plus d'informations.

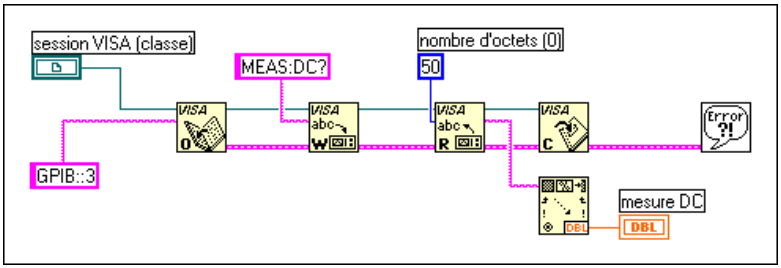
De façon facultative, chaque VI d'initialisation appelle une recherche d'identification spécifique à un modèle d'instrument ou à une série de modèle. Vous devez soit désactiver cette option, soit faire passer la réponse sur la commande de demande d'identification. Pour les instruments SCPI, cette commande est *IDN?.

Le nombre des modifications apportées à un driver d'instrument dépend du degré de similarité des instruments et de leur ensemble de commandes. Si les ensembles de commandes sont très différents, nous vous recommandons de commencer à partir de zéro.

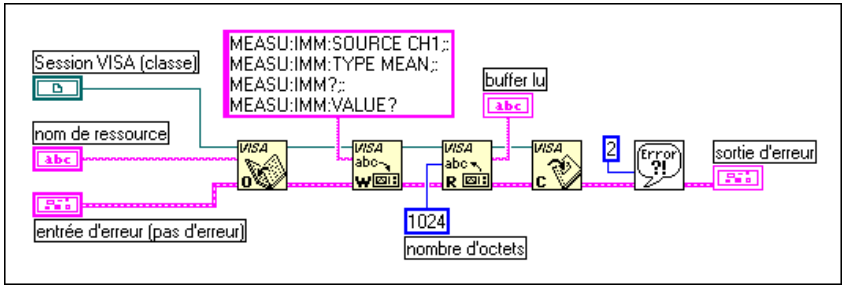
Développement d'un driver simple

La plupart des instruments basés messages sont contrôlés par programmation grâce à une série de lectures et d'écritures sur l'instrument. Pour la plupart des drivers d'instruments simples, il n'y a que quatre fonctions VISA nécessaires : VISA Open, VISA Write, VISA Read et VISA Close.

Le VI driver d'instrument simple présenté dans l'illustration suivante ne lit et n'écrit qu'une fois dans un instrument. Ce VI ouvre d'abord les ressources de l'instrument avec la fonction "VISA Open". Il envoie ensuite la commande MEAS:DC? (comme décrit dans le manuel utilisateur des instruments) pour retourner une mesure DC de l'instrument. La fonction "VISA Read" retourne la mesure sous la forme d'une chaîne de caractères. Pour utiliser la mesure dans d'autres fonctions numériques, la chaîne de caractères est convertie en un nombre numérique grâce à la fonction "*Balayer une chaîne de caractères*". Une fois que la dernière lecture ou écriture de l'instrument est achevée, la fonction "VISA Close" est appelée. Elle est suivie par le VI "Gestionnaire simple d'erreurs" (General Error Handler.vi) pour traiter toutes les erreurs pouvant s'être produites dans les fonctions E/S d'instrument.

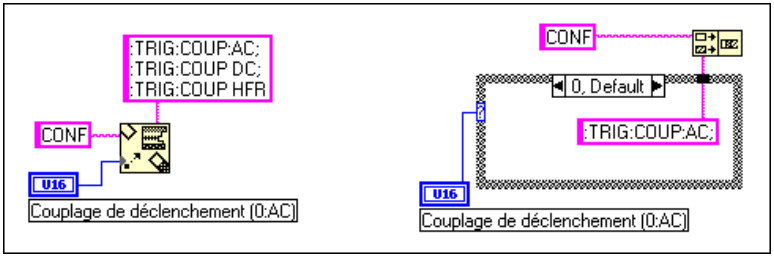


Pour un driver d'instrument plus modulaire, vous pouvez séparer la lecture et l'écriture dans l'instrument d'après le type d'opération que vous essayez d'accomplir. Vous pouvez également combiner la lecture et l'écriture nécessaires à l'établissement de la configuration dans un VI, tandis que la lecture des mesures se trouve dans un autre VI. Pour des mesures répétées, vous pouvez placer vos VIs de mesure dans une boucle. Si vous connaissez exactement le type de votre configuration, vous pourrez probablement inclure toutes les commandes de configuration dans une constante de chaîne de caractères, comme présenté ci-dessous.



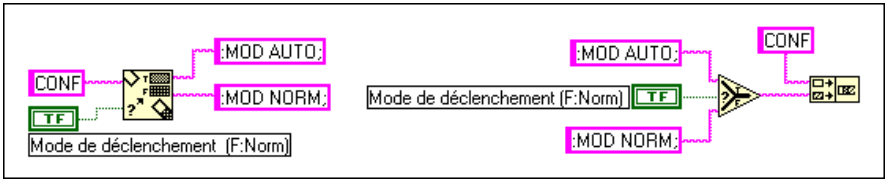
Si, d'un autre côté, vous souhaitez qu'un utilisateur soit capable de sélectionner des configurations différentes, vous devrez construire des chaînes de caractères de commandes par programme. Vous pouvez utiliser la fonction "Piocher une ligne & Ajouter" pour choisir parmi une sélection de chaînes de caractères, puis concaténer cette chaîne avec une autre chaîne de caractères, en une seule étape. Il est plus facile de suivre cette procédure que d'utiliser une structure Condition et la fonction "Concaténer des chaînes de caractères".

Dans l'illustration suivante, il est ainsi plus facile d'utiliser le diagramme de gauche que le diagramme de droite.

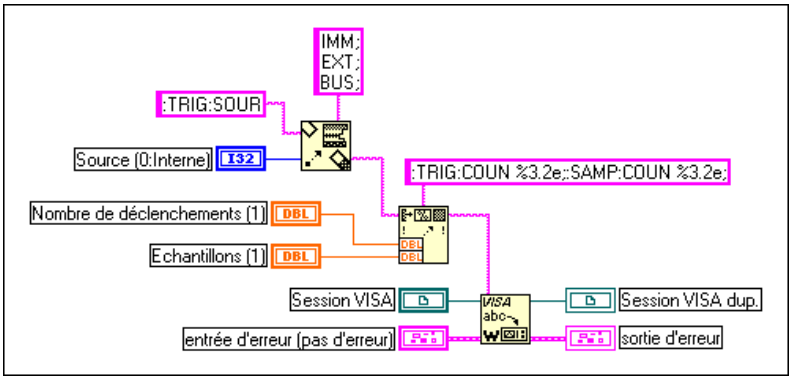


En utilisant la fonction "Sélectionner & Ajouter", vous pouvez sélectionner une constante chaîne de caractères et la concaténer avec une autre chaîne de caractères, en une seule étape. Il est plus facile de suivre cette procédure que d'utiliser une fonction Sélectionner suivie d'une fonction Concaténer.

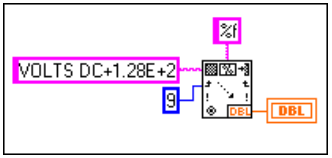
Dans l'illustration suivante, il est ainsi plus facile d'utiliser le diagramme de gauche que le diagramme de droite.



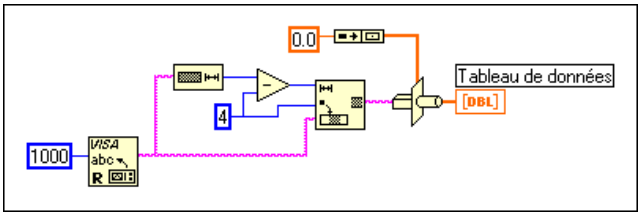
La fonction "Formater en chaîne de caractères" et la fonction "Formater & Ajouter" représentent d'autres fonctions de chaînes de caractères utiles pour la construction de chaînes de caractères de commandes. Ces fonctions convertissent une ou plusieurs valeurs en une chaîne de caractères, avec plusieurs options de formatage. La fonction "Piocher une ligne & Ajouter" et la fonction "Formater en chaîne de caractères" sont illustrées toutes deux dans le diagramme ci-dessous :



Après avoir lu la réponse de l'instrument, vous devez transformer la mesure en une valeur numérique. La fonction "Balayer une chaîne de caractères" est utile pour convertir des nombres ASCII en valeurs numériques. Le code suivant enlève la partie "VOLTS DC" de l'entrée de chaîne de caractères et convertit le "+1.28E+2" en un nombre numérique double précision. L'entrée de chaîne de caractères est typique d'une réponse d'un multimètre.



Si votre instrument retourne des données binaires, utilisez la fonction "Adapter le type". Cette fonction modifie le type des données d'un fil de liaison, mais pas la façon dont les données sont enregistrées en mémoire. La fonction "VISA Read" retourne les données de chaîne de caractères, qu'elles soient codées en ASCII ou en binaire. Ainsi, pour convertir la chaîne de caractères binaire en une valeur numérique ou en un tableau numérique, vous devez adapter le type de la chaîne de caractères en un type de données différent. L'exemple suivant enlève un en-tête de 4 octets d'une chaîne de caractères de réponse de 1000 octets, puis convertit les valeurs restantes en un tableau de valeurs en simple précision.



Développement d'un driver "toutes fonctions"

Si vous développez un driver qui sera utilisé par d'autres, vous souhaiterez peut-être développer un driver "toutes fonctions". Ces drivers sont plus modulaires et possèdent une architecture similaire à celles de la bibliothèque de drivers d'instruments de National Instruments, complétée par des fonctions utilitaires et de report d'erreurs. Pour plus de détails sur le développement d'un driver, reportez-vous à la Note d'application 006 intitulée *Developing a LabVIEW instrument driver (Développement d'un driver d'instrument LabVIEW)*, sur le site web de National Instruments.

Utilisation de LabVIEW avec des drivers d'instruments IVI

En plus des drivers de code source de LabVIEW (plus de 600 en tout), vous pouvez contrôler les instruments avec les drivers d'instruments IVI (Intelligent Virtual Instruments, ou *Instruments virtuels intelligents*). Les drivers d'instruments IVI sont des drivers basés sur des DLL développés sous LabWindows/CVI qui offrent aux utilisateurs des avantages supplémentaires, notamment :

- Une mise en mémoire cache de l'état de l'instrument pour améliorer ses performances
- La simulation
- La sécurité multi-thread
- Accès à l'attribut de l'instrument

Consultez la *Référence en ligne* de LabVIEW pour plus d'informations sur les drivers d'instruments IVI et sur la façon de les utiliser.

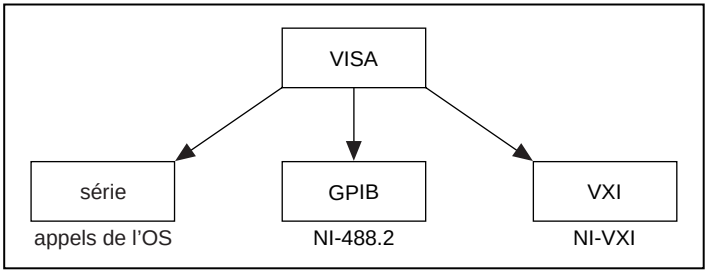
8

Tutorial VISA de LabVIEW

Ce chapitre présente une vue générale de l'implémentation de VISA dans LabVIEW. Il explique les concepts de base relatifs à la programmation d'instruments avec VISA et donne des exemples illustrant des concepts simples de VISA.

Qu'est-ce que VISA ?

VISA est une interface de programmation d'application d'E/S standard (Application Programming Interface – API) pour la programmation d'instrumentation. L'interface VISA ne permet pas en elle-même la programmation d'instrumentation. VISA est une interface API de haut niveau qui appelle des drivers de bas niveau. La hiérarchie VISA de National Instruments (NI–VISA) est présentée dans la figure ci-dessous.



VISA peut contrôler des instruments VXI, GPIB ou série, en appelant les drivers correspondant au type d'instrument utilisé. Lors des mises au point d'applications utilisant VISA, il est important de garder à l'esprit cette hiérarchie. Ce qui semble un problème VISA peut être en réalité un problème avec l'un des drivers appelés par VISA.

Plates-formes et environnements supportés

Comme VISA est le standard industriel pour le développement de drivers d'instruments, la plupart des drivers d'instruments actuellement écrits par National Instruments utilisent VISA et sont donc supportés sous Macintosh, Windows 3.*x*, Windows 95, Windows NT, Solaris 1, Solaris 2

et HP-UX, si les drivers de niveau système sont disponibles pour cette plate-forme.

Pourquoi utiliser VISA ?

VISA est le standard

VISA est l'interface API standard pour les drivers d'instruments dans l'industrie de l'instrumentation. De plus, vous pouvez utiliser une API pour contrôler une suite d'instruments de différent types, comme des instruments VXI, GPIB et série.

Indépendance d'interface

VISA utilise les mêmes fonctions pour communiquer avec les instruments, quel que soit le type d'interface. Par exemple, la commande VISA pour écrire une chaîne ASCII dans un instrument basé messages est la même, que l'instrument soit GPIB, VXI ou série. VISA fournit ainsi une indépendance d'interface. Cela permet de commuter plus facilement des interfaces de bus, ce qui signifie que les utilisateurs qui doivent programmer des instruments pour des interfaces différentes n'ont besoin d'apprendre qu'une seule interface API.

Indépendance de plate-forme

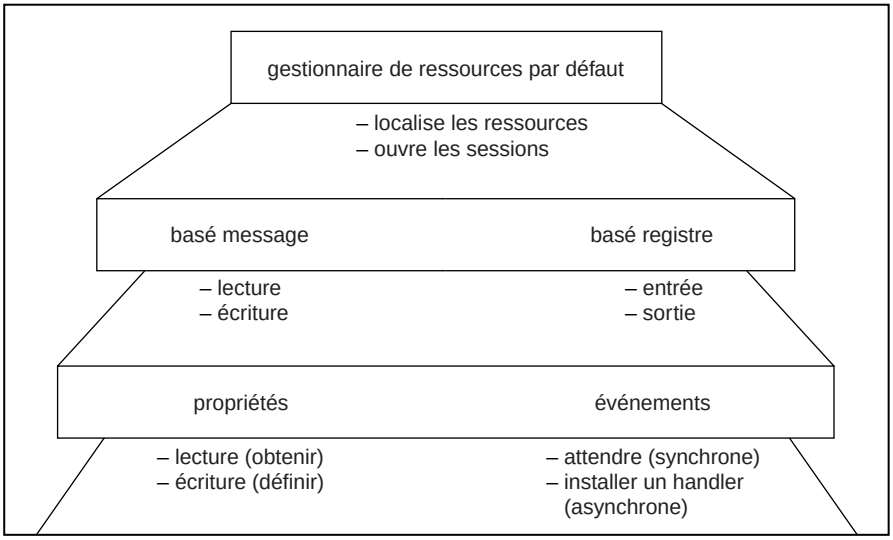
VISA est conçu pour que les programmes écrits en utilisant des appels de fonction VISA soient facilement transférables d'une plate-forme à une autre. Pour assurer l'indépendance de plate-forme, VISA définit strictement ses propres types de données. De cette manière, les problèmes tels que la taille en octets d'un entier, variable d'une plate-forme à une autre, n'affectent pas un programme VISA. Les appels de fonction VISA et leurs paramètres associés sont uniformes sur toutes les plates-formes. Un logiciel peut être transféré sur d'autres plates-formes, puis recompilé. Un programme LabVIEW peut être transféré sur toute plate-forme supportant LabVIEW.

Adaptation future facilitée

Un autre avantage de VISA est d'être une interface API orientée objet qui s'adaptera facilement aux nouvelles interfaces d'instrumentation développées dans le futur et facilitera le transfert d'applications avec les nouvelles interfaces.

Concepts de base de VISA

Une ébauche simplifiée de la structure interne de l'API VISA est présentée dans le diagramme ci-dessous.



Gestionnaire de ressources par défaut, session, et descripteurs d'instrument

Le gestionnaire de ressources par défaut se trouve au niveau le plus élevé des opérations VISA. LabVIEW établit automatiquement une communication avec le gestionnaire de ressources par défaut lors du premier appel de VI VISA. Ceci nous amène à deux termes qui ont besoin d'être définis : la ressource et la session.

Ressource : entité avec laquelle vous pouvez communiquer : par exemple, les ressources d'instrument (INSTR) et les ressources d'accès à la mémoire (MEMACC).

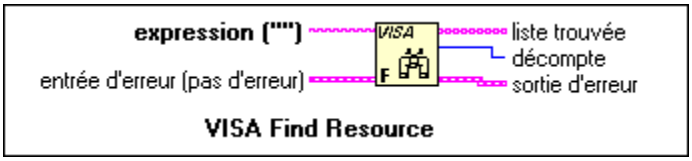
Session : connexion (lien) à n'importe quelle ressource existante, y compris le gestionnaire de ressources par défaut.

Vous devez utiliser le gestionnaire de ressources par défaut VISA pour ouvrir des sessions vers d'autres ressources. Vous devez ouvrir des sessions vers des instruments avant qu'un VI ne puisse communiquer avec eux.

Le gestionnaire de ressources par défaut VISA peut aussi chercher des ressources disponibles dans le système. Vous pouvez alors ouvrir des sessions vers n'importe laquelle de ces ressources.

Comment rechercher les ressources ?

La fonction "VISA Find Resource" présentée ci-dessous recherche les ressources disponibles dans le système. Cette fonction est un point de départ classique pour un programme VISA. Vous pouvez l'utiliser pour déterminer si toutes les ressources nécessaires à l'exécution de l'application sont disponibles.

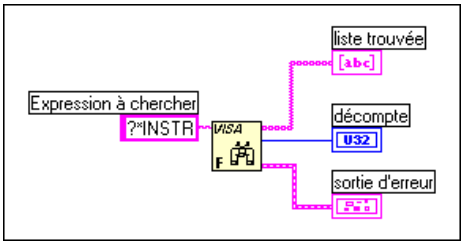


La seule entrée nécessaire à la fonction "VISA Find Resource" est une chaîne appelée l'**expression de recherche**. Elle détermine les types de ressources retournés par la fonction Find Resource. Des chaînes possibles pour l'**expression** de recherche sont indiquées dans le tableau ci-dessous et figurent dans la *Référence en ligne* LabVIEW.

Ressources d'instrument	Expression
GPIB	GPIB[0 – 9]*::? *INSTR
GPIB-VXI	GPIB-VXI?*INSTR
GPIB ou GPIB-VXI	GPIB?*INSTR
VXI	VXI?*INSTR
Toutes les ressources VXI	?*VXI[0 – 9]*::?*INSTR
Série	ASRL[0 – 9]*::?*INSTR
Toutes	?*INSTR

Ressources d'accès à la mémoire	Expression
VXI	VXI?*MEMACC
GPIB-VXI	GPIB-VXI?*MEMACC
Toutes les ressources VXI	?*VXI[0–9]*::?*MEMACC
Toutes	?*MEMACC

Les valeurs retournées d'une fonction sont le **nombre trouvé** (qui correspond au le nombre de ressources trouvées) et la **liste trouvée**. La **liste trouvée** est un tableau de chaînes de caractères. Chaque chaîne contient la description de l'une des ressources trouvées. Ces chaînes sont appelées des descripteurs d'instruments. Un exemple de VI recherchant toutes les ressources disponibles dans le système est présenté dans la figure ci-dessous.



Descripteur d'instrument : le nom et l'emplacement exacts d'une ressource VISA. Cette chaîne a le format suivant :

Type d'interface[indice de carte]::adresse::Classe VISA

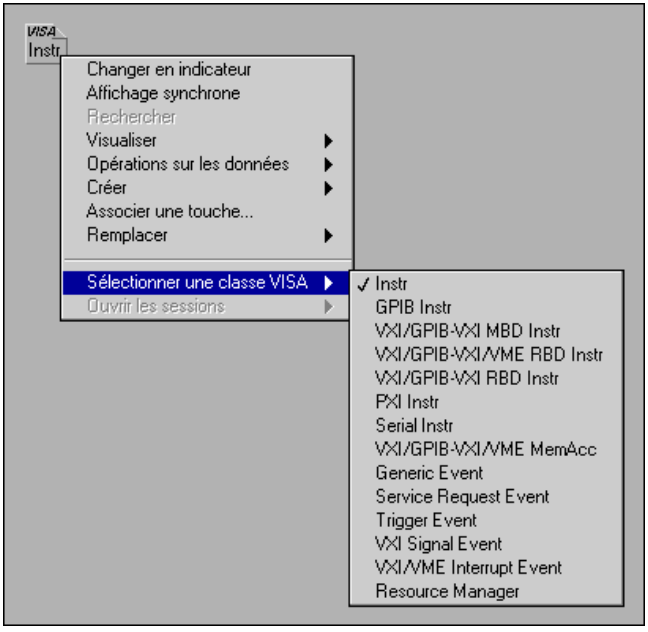
Les descripteurs d'instruments sont simplement des instruments spécifiques trouvés par une recherche. L'indice de carte n'a besoin d'être utilisé que si plus d'un type d'interface est présent dans le système. Par exemple, si le système contient deux cartes enfichables GPIB, une carte peut être identifiée comme GPIB0 et une autre comme GPIB1. Dans ce cas, l'indice de carte doit être utilisé dans les descripteurs d'instrument. Pour les instruments VXI, le paramètre d'adresse est l'adresse logique de l'instrument. Pour les instruments GPIB, c'est l'adresse GPIB principale. Les instruments série n'utilisent pas le paramètre d'adresse. Par exemple, ASRL1::INSTR est le descripteur pour le port série COM 1 sur un ordinateur personnel.

Qu’est-ce qu’une classe VISA ?

Une classe VISA est un groupe qui contient certaines ou l’ensemble des opérations VISA. INSTR est la classe la plus courante qui couvre toutes les opérations VISA pour un instrument. Dans le futur, d’autres classes pourront être ajoutées à la spécification VISA. Actuellement, la classe VISA n’a pas besoin d’être incluse comme partie du descripteur d’instrument, mais vous devez l’inclure pour assurer une compatibilité future. Actuellement, si la classe VISA est laissée vide, elle est définie par défaut par la classe INSTR.

Menu local d’une commande VISA

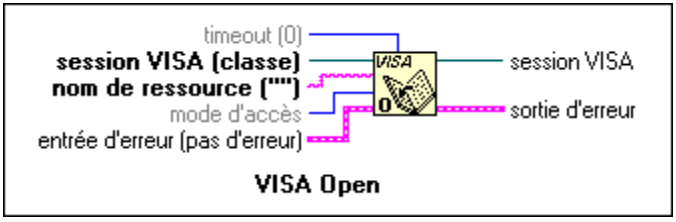
LabVIEW offre désormais une autre façon d’établir la classe d’une session VISA. Vous pouvez ouvrir le menu local d’une commande VISA Session de la face-avant et choisir l’option Sélectionner une classe VISA, comme indiqué dans la figure suivante.



Si une classe autre que la classe d’instrument par défaut est sélectionnée, seules les fonctions d’opérations associées à cette classe de périphérique peuvent être câblées correctement à cette session. Par exemple, si GPIB Instr est sélectionné pour la classe VISA, les fonctions pour l’accès aux registres VXI ne peuvent pas être câblées à la session.

Ouverture d'une session

Les descripteurs d'instruments sont utilisés pour ouvrir des sessions vers les ressources du système. La fonction "VISA Open" est présentée ci-dessous.



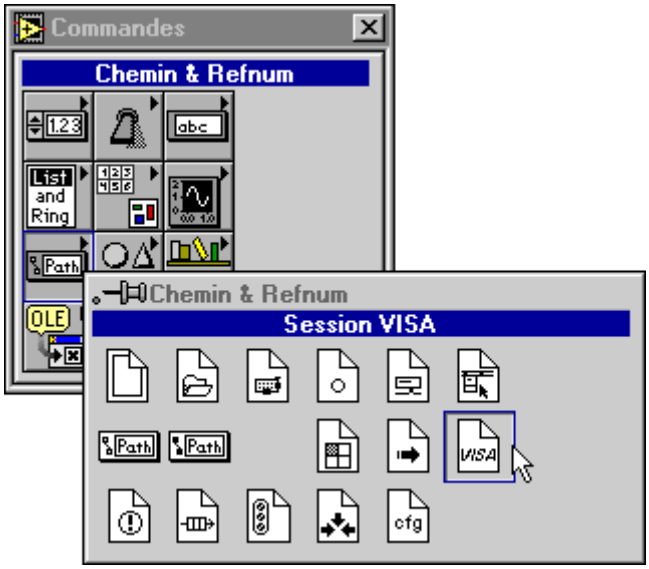
L'entrée **nom de ressource** est le descripteur d'instrument VISA de la ressource pour laquelle une session est ouverte.

☞ **Remarque** ***Vous n'avez pas besoin d'utiliser le VI "Find Resource" pour obtenir des descripteurs d'instruments pour les ressources. Le VI "VISA Open" peut être utilisé avec un descripteur d'instrument fourni par l'utilisateur ou le programmeur. Cependant, pour être sûr de la syntaxe du descripteur, il est préférable d'exécuter d'abord "Find Resource".***

☞ **Remarque** ***Dans la plupart des applications, les sessions n'ont besoin d'être ouvertes qu'une fois, pour chaque instrument avec lequel l'application communique. Cette session peut être transmise à travers l'application, puis fermée à la fin de l'application.***

Remarquez qu'il y a aussi une entrée de session VISA pour la fonction "VISA Open". Pour ouvrir des sessions aux ressources dans LabVIEW, la commande de la face-avant de la session VISA est nécessaire. Cette

commande se trouve dans la sous-palette **Chemin & Refnum** de la palette **Commandes**.



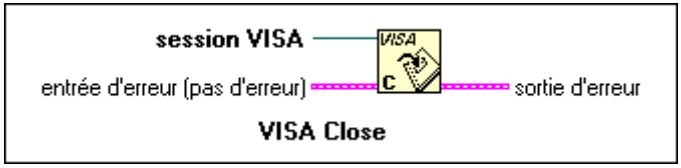
Quel est le lien entre le gestionnaire de ressources par défaut, les descripteurs d'instruments et les sessions ?

Il est important de comprendre clairement ce que sont le gestionnaire de ressources par défaut, les descripteurs d'instruments et les sessions. Une bonne analogie peut être faite entre le gestionnaire de ressources par défaut VISA et un opérateur de téléphone. Ouvrir une session dans le gestionnaire de ressources par défaut (rappelez-vous qu'elle s'ouvre automatiquement dans LabVIEW) équivaut à décrocher le téléphone et à appeler l'opérateur pour établir une ligne de communication entre un programme et le driver VISA.

A son tour, l'opérateur téléphonique peut composer des numéros de téléphone pour établir des lignes de communication avec des ressources système. Les numéros de téléphone utilisés par le gestionnaire de ressources sont les descripteurs d'instruments. Les lignes de communication sont les sessions ouvertes vers les ressources VISA. En outre, le gestionnaire de ressources peut rechercher tous les numéros de téléphone disponibles. Il s'agit de l'opération VISA Find Resource.

Fermeture d'une session

Une session ouverte vers une ressource VISA utilise également les ressources système de l'ordinateur. Pour fermer correctement un programme VISA, toutes les sessions VISA ouvertes doivent être fermées. Pour cela, vous devez utiliser la fonction "VISA Close", présentée ci-dessous.



L'entrée de session VISA de la fonction "VISA Close" est la session à fermer. Cette session provient en général d'un terminal de sortie "VISA Session" d'une fonction "VISA Open".

Si une session n'est pas fermée lorsqu'un VI fonctionne, elle reste ouverte. Il vaut mieux fermer les sessions dans une application pour que les sessions ouvertes ne s'accumulent pas, ce qui peut causer des problèmes avec les ressources système. Il y a cependant des cas où laisser des sessions ouvertes peut être utile.

☞ **Remarque** ***Si l'exécution d'un VI est abandonnée (par exemple, lors de sa mise au point), la session VISA n'est pas fermée automatiquement. Vous pouvez utiliser le VI de contrôle de l'ouverture des sessions VISA (Open VISA Session Monitor.vi), situé dans le répertoire*** `vi.lib/Utility`***, pour une assistance à la fermeture de telles sessions.***

Quand faut-il laisser une session ouverte ?

Si un VI fonctionne et laisse une session ouverte sans la fermer, cette session peut être utilisée, plus tard, par d'autres VIs. Vous pouvez accéder à une session ouverte en ouvrant le menu local d'une commande de Session VISA de la face-avant et en sélectionnant **Sessions ouvertes...** La sortie de la commande Session VISA devient alors la session ouverte sélectionnée. De cette manière, les sessions laissées ouvertes lors d'exécutions antérieures de VI peuvent être fermées. Cette méthode peut aussi être utilisée pour tester de manière interactive certaines parties d'une

application. Un exemple de sélection d'une session ouverte est présenté dans la figure suivante.

Vous pouvez utiliser la commande de session VISA pour contrôler quelles sont les sessions ouvertes. L'accès aux sessions ouvertes par le menu local des commandes de session VISA de la face-avant représente également une manière pratique d'exécuter certaines parties d'un driver d'instrument.

Gestion d'erreurs avec VISA

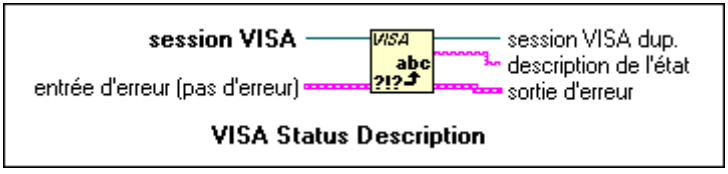
La gestion d'erreurs avec des VIs VISA est similaire à la gestion d'erreurs avec d'autres VIs d'E/S de LabVIEW. Chacun des VIs VISA contient des terminaux d'entrée et de sortie d'erreur utilisés pour passer des clusters d'un VI à un autre dans un diagramme. Le cluster d'erreur contient un flag booléen indiquant si une erreur s'est produite, un code d'erreur VISA numérique et une chaîne de caractères contenant l'emplacement du VI dans lequel l'erreur s'est produite. Si une erreur se produit, les prochains VIs n'essaient pas de s'exécuter et passent simplement au cluster d'erreur. Un indicateur de cluster d'erreur sur la face-avant montrant la sortie du terminal d'erreur d'un VI VISA est présenté dans la figure ci-dessous.

Remarquez que, dans ce cas, une erreur s'est produite. Notez également que le code d'erreur VISA est coupé dans l'indicateur de code. Les codes

d'erreur VISA sont des entiers à 32 bits, généralement dans un format hexadécimal. Le cluster de l'erreur LabVIEW affiche le code en décimal. Le sujet *Codes d'erreur VISA* dans la *Référence en ligne* LabVIEW, et la section *Codes d'erreur numérique* dans l'annexe A, intitulée *Codes d'erreur* dans le *Manuel de référence des VIs et des fonctions de LabVIEW*, listent aussi les codes d'erreur en décimal. Cependant, comme la figure ci-dessus le montre, ces codes d'erreur sont coupés dans le cluster d'erreur. L'indicateur de code doit être redimensionné pour afficher le code d'erreur en entier.

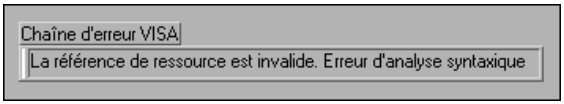
Les VIs gestionnaires d'erreur simple et général (General Error Handler.vi) LabVIEW se trouvent dans la sous-palette **Temps & Dialogue** de la palette de **Fonctions**. Si une erreur se produit, ces VIs affichent une boîte de dialogue locale qui contient les raisons possibles de l'erreur. Le VI Gestionnaire simple d'erreurs (Simple Error Handler.vi) retourne la même erreur que le cluster de l'erreur utilisé dans l'exemple précédent, mais fournit des informations plus détaillées sur l'erreur, comme présenté dans la figure suivante.

Remarquez que la description du code est listée comme une raison possible. Il n'est pas toujours pratique de gérer des erreurs avec des boîtes de dialogue locales dans les VIs gestionnaires d'erreur LabVIEW. VISA fournit aussi une opération qui prend un code d'erreur VISA et produit comme sortie la chaîne de message d'erreur correspondant au code. Ce VI est présenté dans la figure ci-dessous.



Les entrées à ce VI sont une session VISA et un cluster d'erreur VISA. Le VI vérifie le code VISA dans le cluster d'erreur d'entrée et génère la

description du code dans **Description de l'état**. La figure suivante montre un indicateur de type chaîne de caractères contenant la chaîne d'erreur retournée du VI Description de l'état VISA.



La méthode exacte utilisée pour implémenter la gestion d'erreurs dépend de la nature du programme. Cependant, certains mécanismes de gestion d'erreurs doivent être implémentés dans tout programme utilisant l'interface VISA.

VIs VISA simples

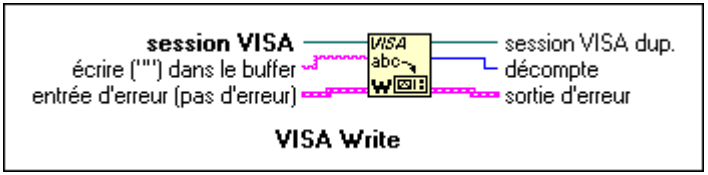
Vous pouvez utiliser les VIs VISA simples pour vérifier que vous avez établi une communication avec votre instrument. Lorsque vous développez vos applications, vous devez utiliser les autres VIs VISA dans la palette car ils vous permettent de mieux contrôler votre instrument. Pour plus d'informations sur les VIs VISA simples, reportez-vous à la section *Test de communication avec votre instrument*, au chapitre 7, *Initiation aux drivers d'instruments LabVIEW*. Les exemples des sections suivantes n'utilisent pas les VIs VISA simples.

Communication basée messages

Les périphériques série, GPIB et de nombreux périphériques VXI reconnaissent différentes commandes basées messages. En ce qui concerne VISA, le protocole réel utilisé pour envoyer une commande à un instrument est transparent. Un utilisateur a seulement besoin de savoir s'il s'agit de lire ou d'écrire un message dans un périphérique basé messages. Les VIs utilisés pour réaliser ces opérations sont VISA Write et VISA Read.

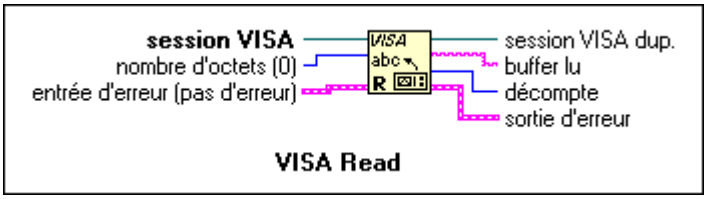
☞ **Remarque** ***Les mêmes VIs sont utilisés pour écrire des commandes basées messages aux instruments GPIB, série et VXI basés messages. VISA repère automatiquement les fonctions de driver à appeler, selon le type de ressource utilisé.***

Le VI VISA Write est présenté ci-dessous.



La seule entrée, en plus de la session, est la chaîne à envoyer à l'instrument.

Le VI VISA Read est aussi facile à utiliser. Il est présenté ci-dessous.

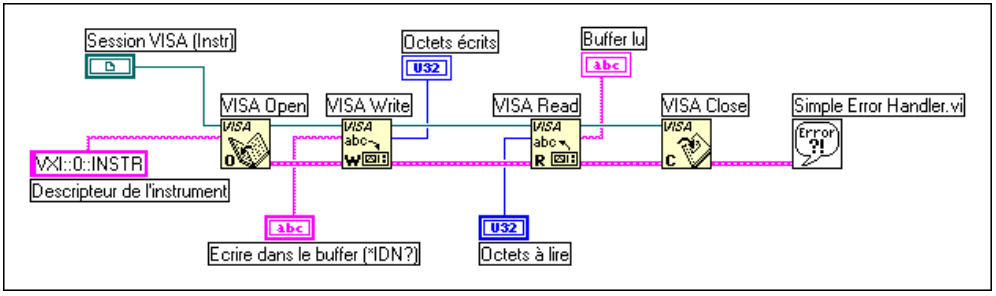


L'entrée nombre d'octets à donner au VI VISA Read doit égaler le nombre maximum d'octets qui doivent être lus dans l'instrument. Le VI arrête la lecture lorsque ce nombre d'octets spécifié a été lu ou lorsque la fin du transfert est indiquée.

Les commandes réelles basées messages reconnues par l'instrument varient selon les fabricants. IEEE 488.2 et SCPI ont standardisé les commandes et beaucoup d'instruments suivent ces standards. Cependant, la seule manière d'être certain des commandes à utiliser pour un instrument particulier est de se référer à la documentation fournie par le fabricant. Il existe toutefois des drivers d'instruments pour de nombreux périphériques basés messages. Ces drivers d'instruments contiennent des fonctions qui rassemblent les chaînes de commandes ASCII appropriées et les envoient à l'instrument. Reportez-vous au site web ou FTP de National Instruments pour obtenir les drivers les plus récents.

Comment écrire et comment lire dans un périphérique basé messages ?

Un exemple simple qui envoie la chaîne *IDN? (identification) à un instrument basé messages et qui lit la réponse est présenté dans la figure ci-dessous.



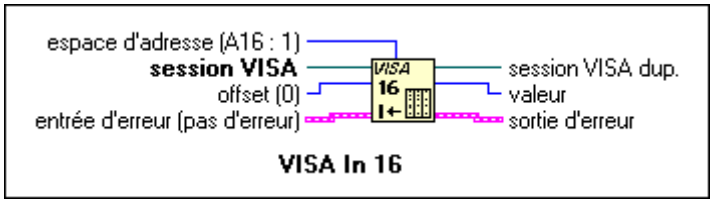
Ce programme peut être utilisé avec tout périphérique reconnaissant la commande *IDN?. Le périphérique peut être série, GPIB ou VXI basé messages. Le seul changement est le descripteur d'instrument.

Communication basée registres (VXI uniquement)

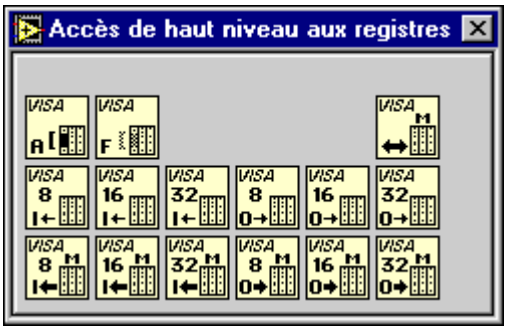
VISA contient un ensemble de VIs d'accès registre à utiliser avec des instruments VXI. Si vous utilisez seulement des périphériques GPIB ou série, vous n'avez pas besoin de lire cette section.

Certains instruments VXI ne supportent pas les commandes basées messages. La seule manière de communiquer avec ces instruments est d'accéder aux registres. Tous les instruments VXI ont des registres de configuration dans les 16 K-octets supérieurs de l'espace mémoire A16. Les fonctions d'accès aux registres peuvent donc également être utilisées pour lire et écrire dans des registres de configuration pour des périphériques basés messages. L'opération VISA de base utilisée pour lire une valeur dans un registre est VISA In. Il existe en fait trois versions différentes de cette opération pour lire une valeur de 8, 16 ou 32 bits. Vous devez utiliser le VI correspondant à la largeur d'accès supportée par votre instrument. Par exemple, un instrument particulier peut retourner une erreur bus pour des

accès 32 bits s'il est conçu pour un accès 16 bits. La fonction VISA In 16 est présentée dans la figure suivante.



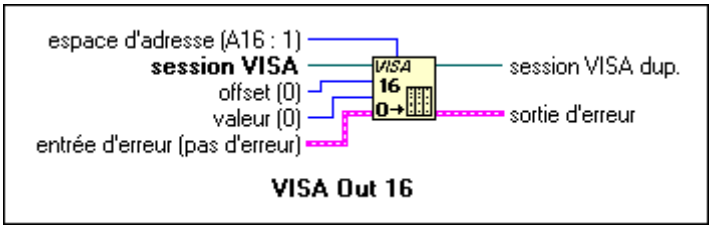
Cette fonction et les autres VIs d'accès registre de base se trouvent dans la sous-palette **Accès de haut niveau aux registres** de la palette des fonctions VISA.



L'entrée **espace d'adresse** indique l'espace d'adresse VXI à utiliser. L'entrée offset peut parfois créer une confusion. Rappelez-vous que VISA garde la trace de l'adresse mémoire de base demandée par un périphérique dans chaque espace d'adresse. L'entrée offset est liée à cette adresse de base.

Considérez l'exemple suivant. Supposez que vous avez un périphérique à l'adresse logique 1 et que vous voulez utiliser la fonction VISA In 16 pour lire son adresse logique et son identification à partir de son registre de configuration. Vous savez que ce registre est à l'adresse absolue 0xC040, dans l'espace A16, et que les registres de configuration pour le périphérique à l'adresse logique 1 vont de 0xC040 à 0xC07F. Cependant, VISA connaît aussi cette information, vous n'avez donc pas besoin de spécifier l'offset de la région à laquelle vous voulez accéder. Dans ce cas, cet offset est égal à zéro.

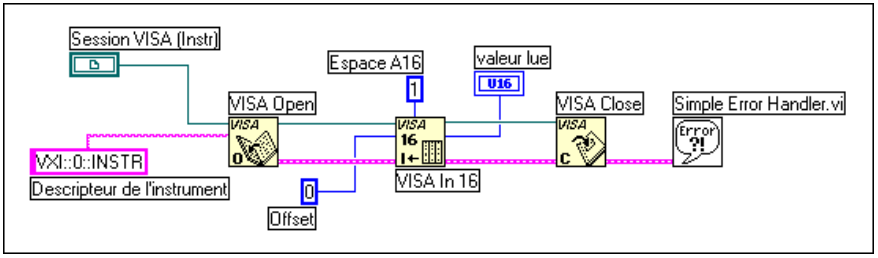
Il existe un autre ensemble d'opérations d'Accès de haut niveau aux registres, en parallèle aux opérations VISA In, mais il concerne l'écriture dans les registres. Ces opérations sont les opérations VISA Out. La fonction VISA Out 16 est présentée ci-dessous.



Cette fonction est similaire au VISA In 16, sauf que la valeur à écrire doit être fournie au terminal **valeur**. Gardez à l'esprit, en utilisant les fonctions VISA Out, que certains registres peuvent ne pas répondre à un cycle d'écriture ou peuvent causer une erreur de bus.

Accès au registre de base

Une utilisation des fonctions d'accès VISA de haut niveau dans un VI est présenté en exemple dans le programme simple ci-dessous.

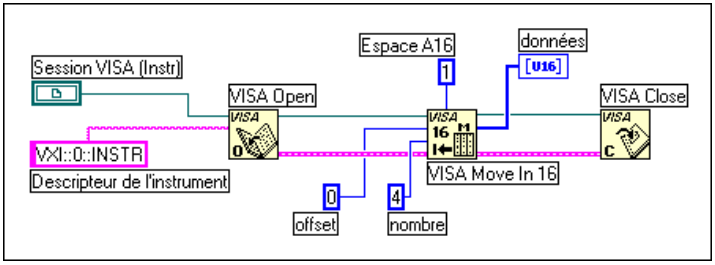


Ce diagramme montre comment utiliser la fonction VISA In 16 pour lire le premier registre de configuration pour un périphérique VXI à l'adresse logique 0. Le paramètre offset, zéro dans ce cas, est fonction de la plage mémoire dont le périphérique a besoin dans l'espace d'adresse VXI auquel il accède. Le paramètre d'espace d'adresse indique l'espace d'adresse VXI utilisé. Dans ce cas, le périphérique est à l'adresse logique 0. Ses registres de configuration sont situés de 0xC000 à 0xC03F dans l'espace d'adresse A16. L'opération VISA In 16 avec offset 0 lit réellement le registre situé à l'adresse 0xC000.

Le programme lit un registre à 16 bits dans l'espace d'adresse A16 à l'offset spécifié (0) de la ressource spécifiée (VXI:0::INSTR). Si une erreur survient dans la séquence des VIs VISA qui s'exécutent dans le diagramme du programme, le gestionnaire d'erreurs simple retourne une boîte de dialogue informant l'utilisateur de l'erreur et affichant le message associé au code d'erreur VISA.

Déplacement simple de registre

Le diagramme suivant montre comment utiliser la fonction VISA Move In 16 pour lire les quatre premiers registres de configuration pour un périphérique VXI à l'adresse logique 0.



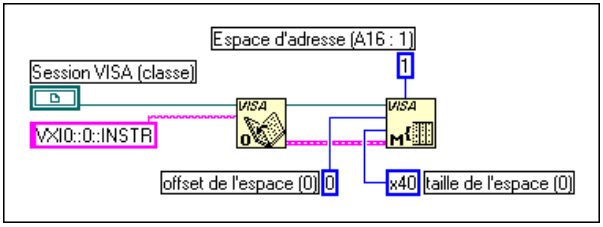
Les fonctions VISA Move In sont utilisées pour lire des blocs importants de données dans les périphériques VXI. Les données sont retournées sous la forme d'un tableau avec quatre valeurs à 16 bits. Il existe un ensemble correspondant de VIs VISA Move Out pour déplacer des blocs importants de données dans des périphériques VXI. Les VIs Move In et Move Out ont des versions à 8, 16 et 32 bits. Le VI approprié est déterminé par la taille des registres auxquels il va accéder.

Fonctions d'accès bas niveau

Les fonctions d'accès bas niveau (LLA – Low Level Access) représentent un moyen très efficace de réaliser une communication basée registres. Les fonctions LLA occasionnent beaucoup moins de délais supplémentaires que les fonctions de haut niveau (HLA – High Level Access) pour certains types d'accès. Les fonctions LLA réalisent les mêmes étapes que les fonctions HLA, à la différence que chaque tâche réalisée par une fonction HLA est une fonction individuelle sous LLA.

Utilisation de VISA pour réaliser des accès bas niveau

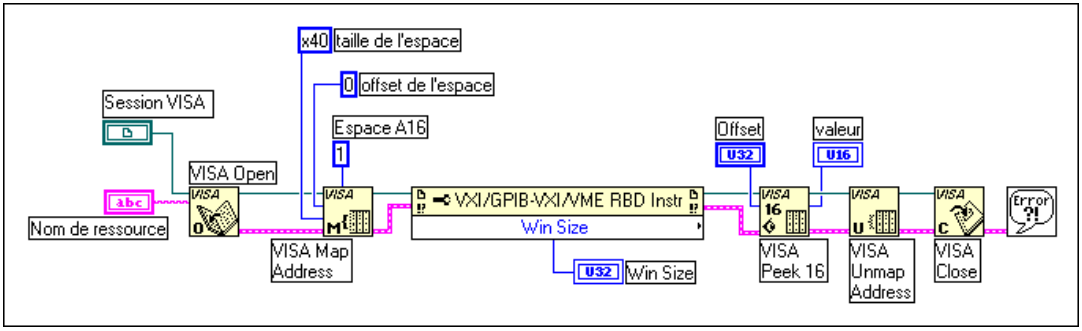
La première opération LLA que vous devez appeler pour accéder à un registre du périphérique est l'opération `VISA Map Address`, qui configure la fenêtre du matériel pour permettre l'accès à l'espace d'adresse VXI. L'opération `VISA Map Address` programme le matériel pour mapper les adresses de l'unité centrale (UC) locales avec les adresses VXI, comme décrit dans la section précédente. Le code suivant est un exemple de programmation du matériel pour accéder à l'espace d'adresse A16.



Ce code configure le matériel pour mapper l'espace A16, en commençant à l'offset 0 pour 0x40 octets. Rappelez-vous que l'offset est fonction de l'adresse de base du périphérique avec lequel nous communiquons grâce à la session `VISA`, et non de la base de l'espace A16 lui-même. Un offset égal à 0 ne signifie donc pas adresse 0 dans l'espace A16, mais représente plutôt le point de départ de la mémoire A16 du périphérique.

☞ **Remarque** ***Pour accéder aux registres du périphérique dans une session MEMACC, vous devez fournir les adresses VXIbus absolues (adresse de base du périphérique + offset du registre dans l'espace d'adresse du périphérique).***

Si vous avez besoin de plus d'un espace d'adresse pour un périphérique, vous devez ouvrir une deuxième session pour le périphérique car VISA ne supporte actuellement qu'un seul espace par session. Il y a un très faible délai supplémentaire avec deux sessions car les sessions elles-mêmes ne prennent pas beaucoup de place. Cependant, vous devez garder la trace des deux handles de session. Remarquez que ceci est différent du nombre maximum de fenêtres que vous pouvez avoir sur un système. Le matériel pour le contrôleur que vous utilisez peut avoir une limite concernant le nombres de fenêtres uniques qu'il peut supporter. Lorsque vous avez terminé avec la fenêtre, ou lorsque vous avez besoin de changer la représentation d'une autre adresse ou d'un nouvel espace d'adresse, vous devez d'abord démapper la fenêtre en utilisant l'opération `VISA Unmap Address`.



Cet exemple mappe 64 octets (hex 40) à l'espace d'adresse A16, en commençant à l'offset 0 à partir de l'adresse du périphérique à l'adresse logique 1. Lorsque vous utilisez les opérations LLA, spécifiez toujours une taille d'espace assez large pour la gamme d'adresses auxquelles vous voulez accéder. Vous pouvez faire cela en utilisant un nœud de propriétés pour déterminer la quantité d'espace d'adresse utilisée par le périphérique. Les nœuds de propriétés sont expliqués plus loin dans ce chapitre.

☞ **Remarque** ***La fenêtre maximale par défaut qui peut être mappée est typiquement de 64 Ko. Si vous utilisez un contrôleur MITE, vous pouvez demander plus de 64 Ko, mais vous devez augmenter la taille de votre fenêtre utilisateur. Cela se fait dans l'éditeur de ressource de votre contrôleur, T&M Explorer, VXIEdit ou VXItedit. Veuillez consulter la documentation fournie avec votre contrôleur.***

Erreurs bus

Les erreurs bus ne sont pas rapportées par les opérations LLA. En fait, `VISA Peek` et `VISA Poke` ne rapportent pas de conditions d'erreur. Cependant, les opérations HLA rapportent les erreurs bus. Lorsque vous utilisez les opérations LLA, vous devez vous assurer que les adresses auxquelles vous accédez sont valides.

Comparaison des accès haut niveau et bas niveau

Vitesse

En termes de vitesse de développement de votre application, les opérations HLA sont implémentées et mises au point plus rapidement grâce à l'interface plus simple et aux informations d'état reçues après chaque accès. Par exemple, les opérations HLA comprennent le mappage et le

démappage les fenêtres du matériel, ce qui veut dire que vous n'avez pas besoin d'appeler séparément `VISA Map Address` et `VISA Unmap Address`.

Concernant la vitesse d'exécution, les opérations LLA fonctionnent plus rapidement lorsqu'elles sont utilisées pour différents accès d'E/S de registre aléatoires dans une fenêtre mappée unique. Si vous savez que les prochains accès se situent dans une fenêtre unique, vous pouvez mapper la fenêtre une seule fois, ainsi chaque accès a un délai minimal.

Les opérations HLA sont plus lentes parce qu'elles doivent réaliser un mappage, un accès et un démappage lors de chaque appel. Même si la fenêtre est mappée correctement pour l'accès, l'appel HLA doit quand même réaliser une sorte de vérification pour déterminer s'il a besoin de mapper à nouveau la fenêtre. De même, étant donné que les opérations HLA comprennent beaucoup de fonctions de vérification d'état non comprises dans les opérations LLA, les opérations HLA ont un délai supplémentaire logiciel plus important. C'est pour cela que les opérations HLA sont plus lentes que les opérations LLA dans beaucoup de cas.

☞ **Remarque** ***Pour des transferts de bloc, les opérations VISA Move de haut niveau sont les plus rapides.***

Facilité d'utilisation

Les opérations HLA sont plus faciles à utiliser parce qu'elles comprennent de nombreuses fonctions de vérification d'état non incluses dans les opérations LLA, ce qui explique un délai supplémentaire logiciel plus important et une vitesse d'exécution plus lente. Les opérations HLA comprennent aussi les fonctions de mappage et de démappage des fenêtres du matériel, ce qui veut dire que vous n'avez pas besoin d'appeler séparément `VISA Map Address` et `VISA Unmap Address.`

Accès à plusieurs espaces d'adresses multiples

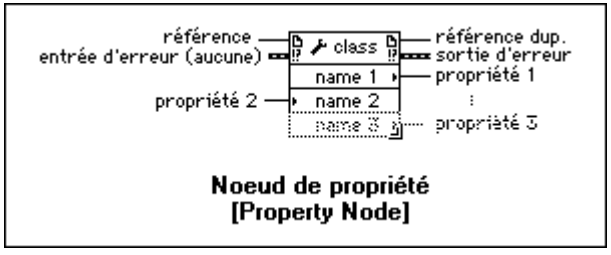
Vous pouvez utiliser les opérations LLA pour accéder uniquement à l'espace d'adresse actuellement mappé avec une session VISA unique. Dans le cas d'une session unique, pour accéder à un espace d'adresse différent, vous devez mapper à nouveau la fenêtre, ce qui implique d'appeler `VISA Unmap Address` et `VISA Map Address`. A cause de cela, la programmation LLA devient plus complexe lorsqu'il s'agit d'accéder à plusieurs espaces d'adresse simultanément.

En outre, si vous avez plusieurs sessions pour des périphériques identiques ou différents, toutes réalisant des E/S sur registres ; elles doivent rivaliser pour le nombre fini de fenêtres de matériel disponibles. Lorsque vous utilisez des opérations LLA, vous devez affecter les fenêtres et toujours vous assurer que le programme ne demande pas plus de fenêtres que celles qui sont disponibles. Les opérations HLA évitent ce problème en restaurant la fenêtre avec les paramètres précédents lorsque la fonction est terminée. Mêmes si toutes les fenêtres sont actuellement utilisées par des opérations LLA, vous pouvez toujours utiliser les fonctions HLA parce qu'elles enregistrent l'état de la fenêtre, remappent, accèdent à la fenêtre, puis la restaurent. Vous n'êtes pas limité par le nombre de fenêtres lorsque vous utilisez les opérations HLA.

Propriétés VISA

Les opérations de base associées aux ressources basées registres et messages dans VISA ont maintenant été présentées. Ces opérations permettent un accès aux registres et une communication basée messages. En plus des opérations de communication de base, les ressources VISA ont différentes propriétés (attributs) avec des valeurs qui peuvent être lues ou définies dans un programme.

Dans un programme LabVIEW, ces propriétés sont gérées par programme de la même façon que sont gérées les propriétés des indicateurs et des commandes de la face-avant. Les nœuds de propriété sont utilisés pour lire ou définir les valeurs des propriétés VISA. Le nœud de propriétés est présenté dans la figure suivante.



Noeud de propriété
[Property Node]

☞ **Remarque** ***Le nœud de propriétés est un nœud générique qui peut aussi être utilisé pour définir les propriétés de ActiveX/OLE et VI Serveur.***

Après avoir mis un nœud de propriétés dans le diagramme, vous pouvez définir les propriétés pour une classe VISA. Vous pouvez utiliser pour ce faire une des méthodes suivantes.

- Câbler une session VISA au terminal d'entrée de référence du nœud de propriétés.
- Ouvrir le menu local du nœud de propriétés et choisir **Instr** dans le menu **Sélectionner une classe VISA**.

Le nœud de propriétés contient un terminal de propriété unique lorsqu'il est mis initialement sur le diagramme. Cependant, sa taille peut être redéfinie pour contenir autant de terminaux que nécessaire. Le terminal initial sur le nœud de propriétés VISA est un terminal de lecture. Cela signifie que la valeur de la propriété sélectionnée dans ce terminal sera lue. Ceci est indiqué par la petite flèche pointant vers la droite sur le bord droit du terminal. Vous pouvez changer de nombreux terminaux individuellement, d'un terminal de lecture à un terminal d'écriture, en ouvrant le menu local de la propriété que vous voulez changer.

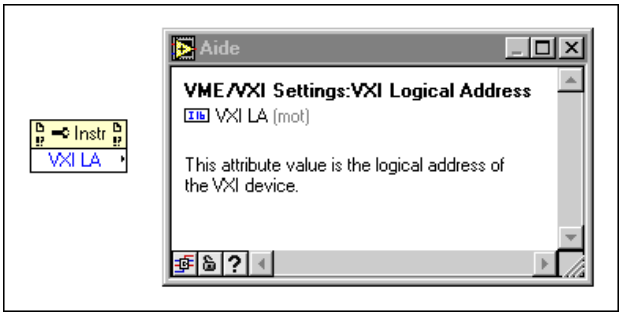
☞ **Remarque** ***Certaines propriétés sont en lecture seule. Leurs valeurs ne peuvent donc pas être définies.***

Pour sélectionner une propriété dans chaque terminal du nœud de propriétés, ouvrez le menu local et choisissez **Sélectionner un élément**. Vous obtenez ainsi une liste de toutes les propriétés possibles qui peuvent être définies dans le programme. Le nombre de propriétés différentes qui sont proposées sous le choix "Sélectionner un élément" peut être limité en changeant la classe VISA du nœud de propriétés.

Pour changer la classe VISA, ouvrez le menu local du nœud de propriétés et sélectionnez **Classe VISA**. Plusieurs classes différentes peuvent être sélectionnées avec cette option, en plus de la classe INSTR par défaut qui couvre toutes les propriétés VISA possibles. Ces classes limitent les propriétés affichées à celles liées à la classe sélectionnée. Une fois qu'une session est connectée au terminal d'entrée de la **Session** du nœud de propriétés, la classe VISA est positionnée en fonction de la classe associée à cette session.

Au départ, les propriétés VISA sont quelque peu obscures et leur nature exacte n'est pas évidente à la seule évocation de leur nom. La *Référence en ligne* LabVIEW contient des informations sur les propriétés. De brèves descriptions de propriétés individuelles sont aussi disponibles dans la fenêtre d'aide simple. Pour obtenir une courte description d'une propriété spécifique, sélectionnez la propriété dans l'un des terminaux d'un nœud de

propriétés, puis ouvrez la fenêtre d'aide. Celle-ci est présentée ci-dessous, dans le cas de la propriété VXI LA.



Remarquez que la fenêtre d'aide présente le type de variable spécifique de la propriété et donne une brève description de sa fonction. Dans les cas où le type de variable à utiliser (pour lire ou écrire dans une propriété), n'est pas clair, rappelez-vous qu'ouvrir le menu local du nœud de propriétés et sélectionner **Créer une constante**, **Créer une commande**, ou **Créer un indicateur** crée automatiquement le type de variable correspondant.

Il y a deux types fondamentaux de propriétés VISA : les propriétés globales et les propriétés locales. Les propriétés globales sont spécifiques à une ressource alors que les propriétés locales sont spécifiques à une session. La propriété VXI LA est un exemple de propriété globale. Elle s'applique à toutes les sessions qui sont ouvertes vers cette ressource. Une propriété locale est une propriété qui pourrait être différente pour des sessions individuelles d'une ressource spécifique. Un exemple de propriété locale est la valeur du timeout. Certaines des propriétés communes pour chaque type de ressource sont présentées dans les listes suivantes.

Série

Serial Baud Rate : débit en baud pour le port série.

Serial Data Bits : nombre de bits de données utilisés pour les transmissions série.

Serial Parity : bit de parité utilisé dans les transmissions série.

Serial Stop Bits : nombre de bits d'arrêt utilisés dans les transmissions série.

GPIB

GPIB Readdressing : spécifie si le périphérique doit être réadressé avant chaque opération d'écriture.

GPIB Unaddressing : spécifie si le périphérique doit être désadressé après des opérations de lecture et d'écriture.

VXI

Mainframe Logical Address : adresse logique la plus basse d'un périphérique situé dans le même châssis que la ressource.

Manufacturer Identification : numéro d'identification du fabricant contenu dans les registres de configuration du périphérique.

Model Code : code du modèle du périphérique contenu dans les registres de configuration du périphérique.

Slot : emplacement du châssis dans lequel se trouve le périphérique.

VXI Logical Address : adresse logique du périphérique.

VXI Memory Address Space : espace d'adresse VXI utilisé par la ressource.

VXI Memory Address Base : adresse de base de la région mémoire utilisée par la ressource.

VXI Memory Address Size : taille de la région de mémoire utilisée par la ressource.

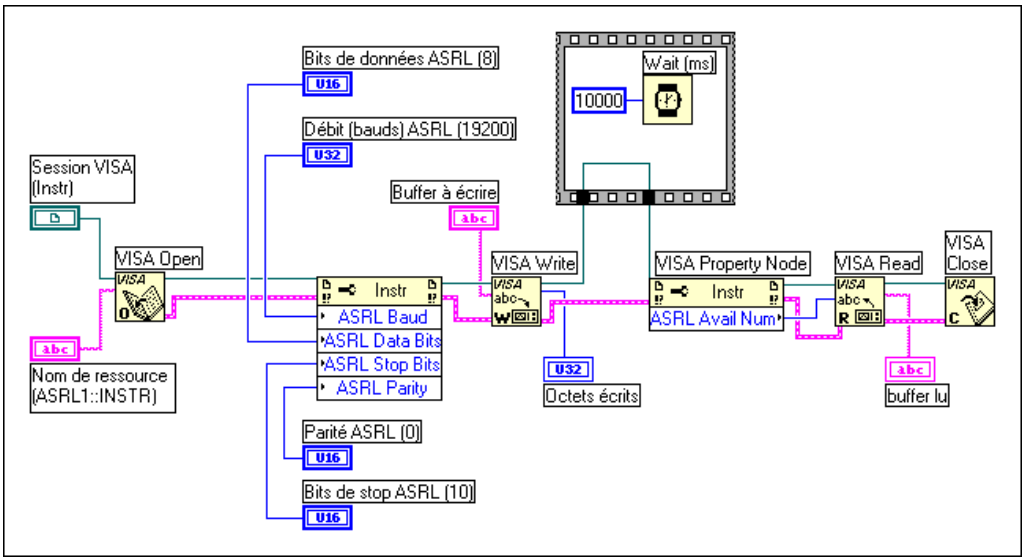
Il y a beaucoup d'autres propriétés en plus de celles listées ici. Il y a aussi des propriétés qui ne sont pas spécifiques à un certain type d'interface. La propriété timeout, qui est le timeout utilisé dans les opérations d'E/S basées messages, est un bon exemple d'une telle propriété. La source d'information la plus complète au sujet des propriétés est la *Référence en ligne* LabVIEW, à laquelle vous pouvez accéder en sélectionnant **Aide» Référence en ligne**.

L'aide en ligne montre à quel type d'interface la propriété s'applique, si la propriété est locale ou globale, son type de données, et la gamme de valeurs valide pour cette propriété. Elle indique aussi des sujets connexes et donne une description détaillée de la propriété.

Exemples de propriétés VISA

Ecriture et lecture série

Cette section présente trois exemples simples d'utilisation des propriétés dans les programmes VISA. Le premier, présenté dans la figure suivante, envoie une chaîne de caractères à un instrument série et lit sa réponse.

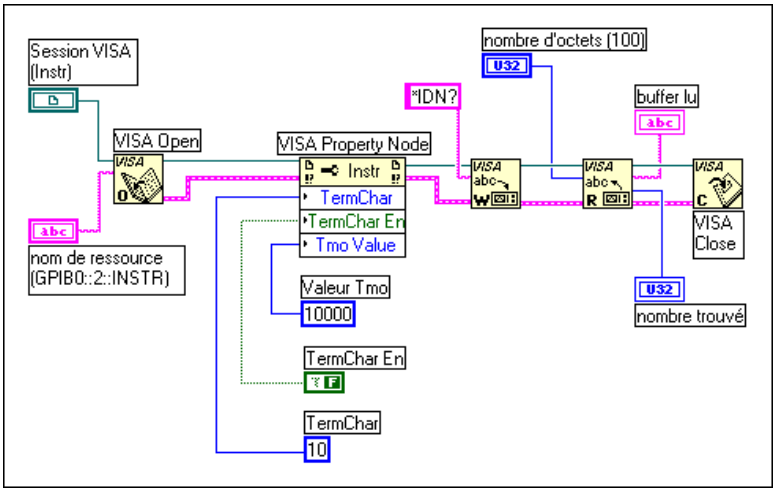


Le VI ouvre une session vers le port série COM 1 et l'initialise pour une communication à 19200 bauds, 8 bits de données, pas de bits de parité et 1 bit d'arrêt. La chaîne est ensuite envoyée vers le port. Dix secondes après l'envoi de la chaîne, le nombre d'octets retournés par le périphérique est obtenu en utilisant une autre propriété VISA. Ces octets sont ensuite lus à partir du port. Remarquez que vous utilisez la valeur 10 pour définir le nombre de bits d'arrêt sur un. (Ceci se trouve dans la spécification VISA. La valeur 10 correspond à 1 bit d'arrêt et 20 à 2 bits d'arrêt.)

Comment définir un caractère de terminaison pour une opération de lecture ?

Le prochain exemple montre comment utiliser les propriétés pour définir un caractère de terminaison pour des opérations de lecture VISA. Certains

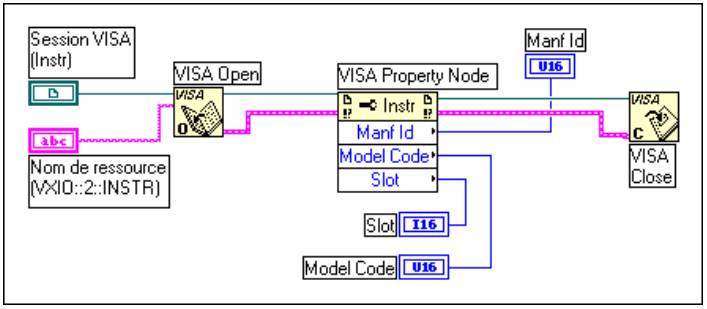
périphériques basés messages envoient un caractère de terminaison spécial lorsqu'ils n'ont plus de données à envoyer.



Ce VI ouvre une session vers l'instrument GPIB à l'adresse primaire 2. Le VI définit le caractère retour à la ligne (valeur décimale 10) comme caractère de terminaison, puis active l'utilisation d'un caractère de terminaison à l'aide d'une autre propriété. Le VI définit aussi le timeout sur 10000 millisecondes (10 secondes). Il envoie ensuite la chaîne *IDN? vers l'instrument et essaie de lire en retour une réponse comportant 100 caractères. La lecture se termine lorsque le caractère de terminaison est reçu. Le VI s'arrête lorsque le caractère de terminaison est reçu, après avoir lu 100 octets, ou après 10 secondes.

Propriétés VXI

Le dernier exemple explique comment lire certaines des propriétés classiques associées à un instrument VXI.



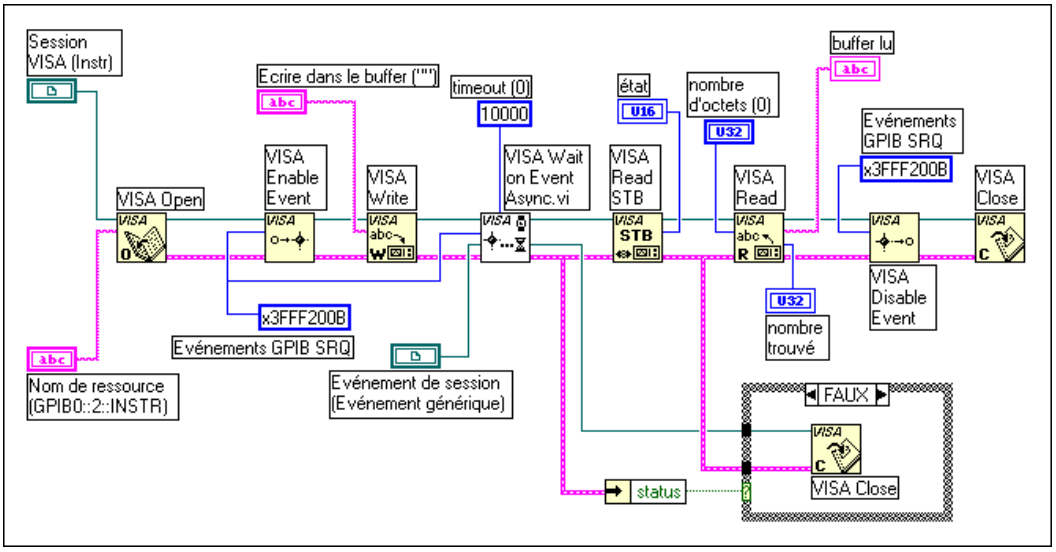
Ce VI ouvre une session vers un instrument VXI situé à l'adresse logique 2 et lit l'identification du fabricant, le code du modèle et l'emplacement du module VXI.

Evénements

Un événement est un moyen de communication VISA entre une ressource et ses applications. C'est une façon pour la ressource de notifier l'application qu'une condition s'est produite et qu'une action de la part de l'application est nécessaire. Des exemples de différents événements sont donnés dans les sections suivantes.

Evénements SRQ GPIB

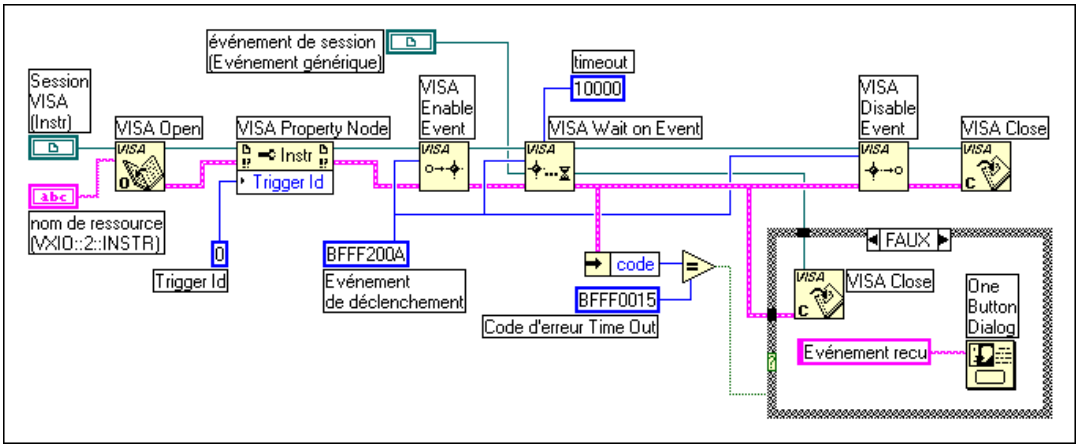
Le diagramme suivant montre comment gérer les événements de requête de service (SRQ) GPIB avec VISA.



Le VI active des événements de requête de service puis envoie une chaîne de commande vers l'instrument. L'instrument doit répondre avec une SRQ lorsqu'il a traité la chaîne. Le VI "Attendre un événement async" (Wait on Event Async VI) est utilisé pour attendre au maximum 10 secondes que l'événement SRQ se produise. Une fois que le SRQ se produit, l'octet d'état de l'instrument est lu avec le VI "Lire un octet d'état" (Read Status Byte VI). L'octet d'état doit être lu après l'apparition des événements SRQ GPIB, sinon des événements SRQ ultérieurs peuvent ne pas être reçus

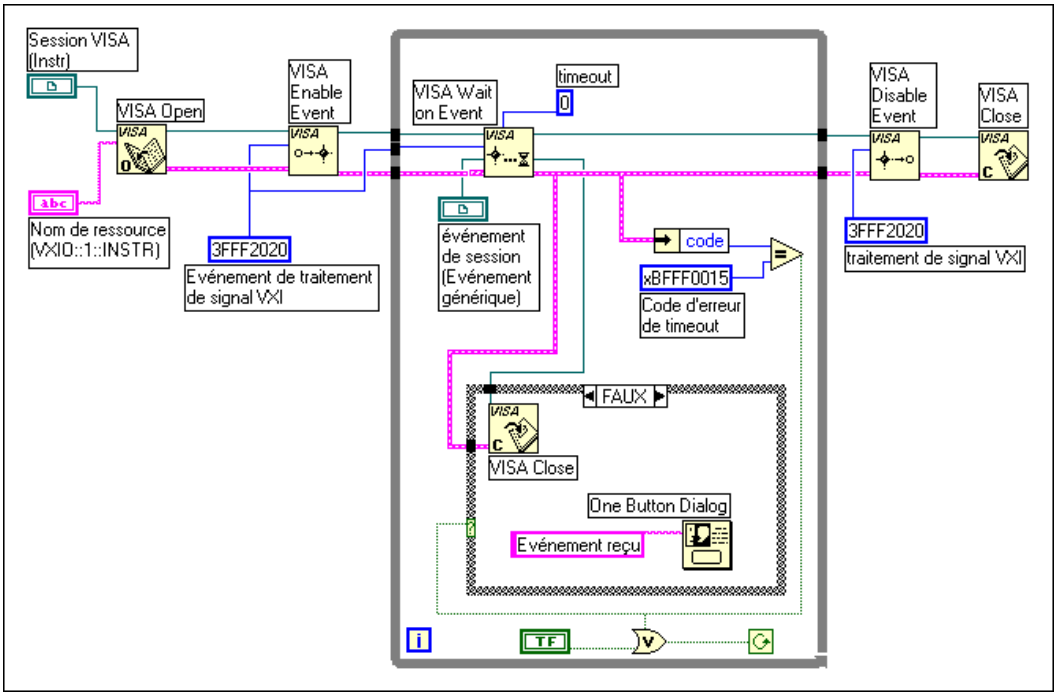
correctement. Finalement la réponse de l'instrument est lue et affichée. Le VI "Attendre un événement async" est différent du VI "Attendre un événement" (Wait on Event VI) normal dans le sens où il appelle continuellement "Attendre un événement" avec un timeout nul pour interroger l'événement. Cela libère du temps pour que d'autres segments parallèles du programme s'exécutent en attendant l'événement.

Evénements de déclenchement



Ce diagramme montre comment détecter un déclenchement sur la ligne de déclenchement TTL 0 pour un périphérique à l'adresse logique 1. Vous devez définir le type d'événement de déclenchement à détecter à l'aide d'une propriété VISA avant que les événements ne soient activés. Le VI attend au maximum 10 secondes que l'événement soit reçu. Si l'événement est reçu correctement, l'événements est fermé dans le VI.

Evénements d'interruption

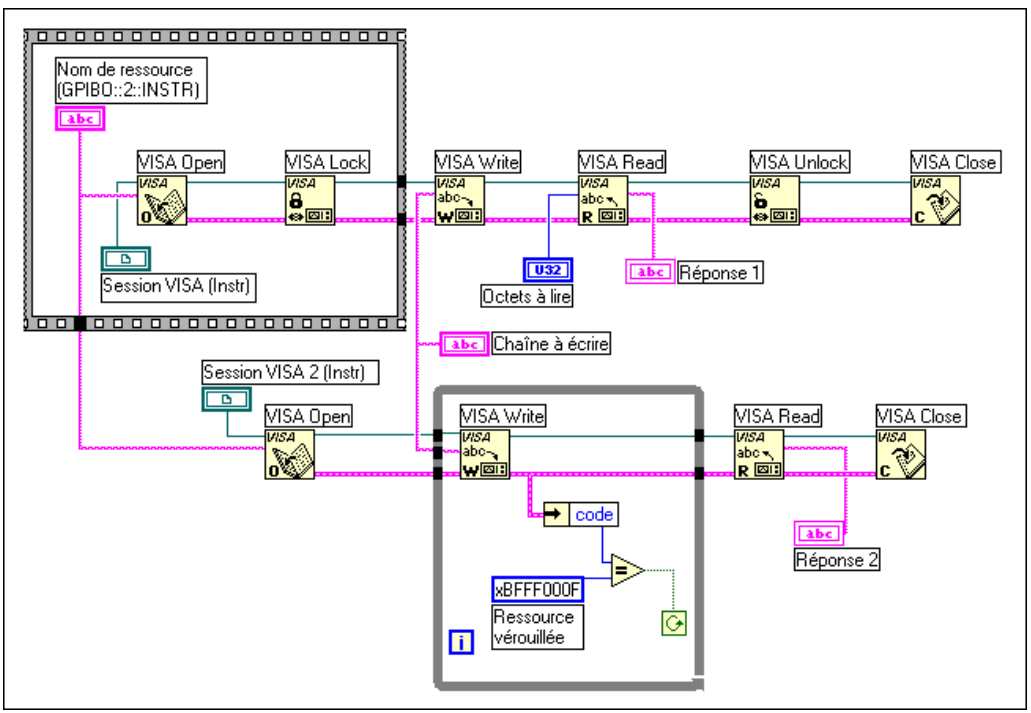


Ce diagramme montre l'utilisation de la capacité de gestion des événements VISA pour détecter une interruption VXI générée par un périphérique VXI à l'adresse logique 1. Le VI active les événements de traitement de signal VXI, puis entre dans une boucle qui appelle de façon répétée VISA "Attendre un événement". La boucle se termine si un événement est reçu ou si un interrupteur d'arrêt sur la face-avant est sélectionné. Le VI "Attendre un événement" a son terminal timeout défini avec la valeur zéro. Dans ce cas, le VI vérifie simplement si des événements ont été reçus, puis retourne immédiatement une erreur timeout s'il n'y a pas d'événement dans la file d'attente. Si un événement est reçu, la session de l'événement est fermée et une notification de l'événement est émise. Une fois la gestion d'événement terminée, les événements sont désactivés.

Verrouillage

VISA introduit des verrous pour le contrôle d'accès des ressources. Dans VISA, les applications GPIB et VXI peuvent ouvrir simultanément plusieurs sessions pour la même ressource et accéder en même temps à la ressource grâce à ces différentes sessions. Dans certains cas, des applications qui accèdent à une ressource doivent empêcher d'autres sessions d'accéder à cette même ressource. Par exemple, une application peut avoir besoin d'exécuter une opération de lecture et une opération d'écriture immédiatement après de sorte qu'aucune autre opération ne doit avoir lieu entre les opérations de lecture et d'écriture. L'application peut donc verrouiller la ressource avant d'appeler l'opération d'écriture et la déverrouiller après l'opération de lecture.

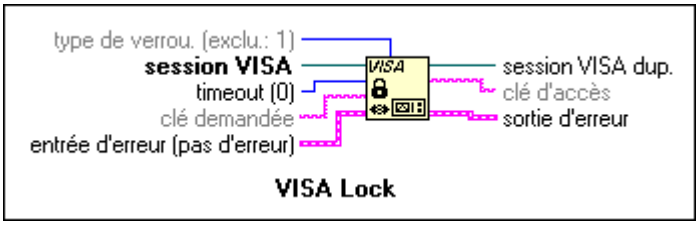
Le mécanisme de verrouillage VISA renforce l'arbitrage d'accès aux ressources sur une base individuelle. Si une session verrouille une ressource, les opérations appelées par d'autres sessions sont traitées ou retournées avec une erreur de verrouillage, selon l'opération et le type de verrou utilisé.



Ce VI ouvre deux sessions pour la même ressource, mais verrouille la première session. La première session envoie ensuite une commande à la ressource et lit sa réponse. Une fois que la séquence écriture/lecture est terminée, la première session déverrouille la ressource. A ce moment-là, la seconde session, qui essaie de réaliser la même séquence écriture/lecture, ne reçoit plus d'erreur de ressource verrouillée sur l'opération d'écriture et elle peut donc s'achever correctement. Le verrouillage peut être utilisé dans les cas où plusieurs applications accèdent à la même ressource ou lorsque plusieurs sessions sont ouvertes vers une ressource dans une application unique.

Verrouillage partagé

Il peut exister des cas où vous souhaitez verrouiller l'accès à une ressource, mais partager cet accès de façon sélective. La figure ci-dessous montre la fonction de verrouillage VISA Lock dans une vue d'aide complexe.



Type de verrouillage est mis par défaut sur exclusif, mais vous pouvez le définir sur partagé. Vous pouvez ensuite câbler une chaîne de caractères à la **clé demandée** pour définir le mot de passe nécessaire pour qu'une autre application puisse accéder à la ressource. Le VI en affecte un de toutes les façons dans la **clé d'accès** si vous n'en demandez pas. Vous pouvez alors utiliser cette clé pour accéder à une ressource verrouillée.

Eléments spécifiques à la plate-forme

Cette section apporte des informations sur la programmation qui vous seront utiles lors du développement d'applications qui utilisent le driver NI-VISA.

Après l'installation du logiciel de driver, vous pouvez commencer à développer votre logiciel d'application VISA. Rappelez-vous que le driver NI-VISA est basé sur NI-488.2 et NI-VXI pour les accès d'E/S au niveau du driver.

- **Windows 95/NT**—Sur les systèmes VXI et MXI, utilisez T&M Explorer pour exécuter le gestionnaire de ressources VXI, configurer votre matériel et affecter des adresses VME et GPIB-VXI. Pour des systèmes GPIB, utilisez le gestionnaire de périphérique du système pour configurer votre matériel. Pour contrôler des instruments à l'aide des ports série, vous pouvez utiliser T&M Explorer pour changer les paramètres par défaut, ou vous pouvez réaliser toute la configuration nécessaire à l'exécution en définissant les attributs VISA.
- **Toutes les autres plates-formes**—Sur les systèmes VXI et MXI, vous devez encore exécuter VXIinit et Resman, et utiliser VXIedit ou VXItedit pour des raisons de configuration. De la même manière, pour les systèmes GPIB et GPIB-VXI, vous devez encore utiliser l'applet du panneau de configuration GPIB ou IBCONF pour configurer votre système. Pour contrôler les instruments via des ports série, vous pouvez réaliser toute la configuration nécessaire lors de l'exécution en définissant les attributs VISA.

Considérations de programmation

Cette section contient des informations qui vous seront utiles lors du développement d'applications utilisant le logiciel d'interface d'E/S NI-VISA.

Plusieurs applications utilisant le driver NI-VISA

Un support multi-application est une fonction importante dans toutes les implémentations du driver NI-VISA. Vous pouvez avoir plusieurs applications qui utilisent NI-VISA s'exécutant en même temps. Vous pouvez même avoir plusieurs copies de la même application qui utilisent le driver NI-VISA en s'exécutant en même temps, si votre application est conçue pour cela. Les opérations NI-VISA fonctionnent de la même manière, que vous ayez une ou plusieurs applications (ou plusieurs cas d'une application) essayant toutes d'utiliser le driver NI-VISA.

Cependant, vous devez faire attention lorsque vous avez plusieurs applications ou sessions utilisant les fonctions d'accès VXIbus de bas niveau. Les fenêtres de mémoire utilisées pour accéder au VXIbus sont une ressource limitée. Vous devez appeler l'opération viMapAddress() avant d'essayer de réaliser des accès VXIbus de bas niveau avec viPeek*XX*() ou viPoke*XX*(). Une fois les accès terminés, vous devez toujours appeler l'opération viUnmapAddress() pour libérer la fenêtre de mémoire pour d'autres applications.

Questions liées à des supports d'interfaces multiples

Cette section contient des informations concernant la façon d'utiliser et/ou de configurer votre logiciel NI-VISA pour certains types d'interfaces.

Plates-formes VXI et GPIB

NI-VISA supporte tout le matériel série, GPIB et VXI National Instruments existant, pour les systèmes d'exploitation sur lesquels NI-VISA existe. Pour le VXI, ceci inclut les plates-formes MXI-1 et MXI-2, GPIB-VXI, et la ligne d'ordinateurs intégrés VXIpc. Pour le GPIB, ceci inclut, entre autre, les séries PCI-GPIB, NB-GPIB, GPIB-SPARC, la ligne complète de cartes AT-GPIB/TNT et la boîte GPIB-ENET que vous pouvez utiliser pour contrôler à distance les périphériques GPIB. Avec GPIB-ENET, vous pouvez même contrôler à distance les périphériques VXI en utilisant un contrôleur GPIB-VXI.

Support GPIB-VXI multiple

Les utilisateurs Windows 95/NT peuvent se référer à l'utilitaire T&M Explorer pour ajouter plusieurs contrôleurs GPIB-VXI de National Instruments, ou tout autre contrôleur GPIB-VXI, à votre système. Les utilisateurs de Windows 3.*x* et UNIX doivent utiliser l'utilitaire VISAconf pour ajouter les contrôleurs.

Support des ports série

NI-VISA ne supporte actuellement qu'une session à la fois sur un port série donné. NI-VISA supporte actuellement 32 ports série maximum sur une plate-forme quelconque. La numérotation par défaut des ports série dépend du système.

Plates-formes	Méthodes
Windows 3.*x*, Windows 95, Windows NT	`ASRL1 – ASRL4` accèdent à `COM1 – COM4` `ASRL10 – ASRL13` accèdent à `LPT1 – LPT4`
Macintosh 68K, Macintosh PPC	`ASRL1` accède au port modem `ASRL2` accède au port de l'imprimante
Solaris 1.*x*	`ASRL1 – ASRL6` accèdent à `/dev/ttya – /dev/ttyf`
Solaris 2.*x*	`ASRL1 – ASRL6` accèdent à `/dev/cua/a – /dev/cua/f`
HP-UX 9 HP-UX 10	`ASRL1` et `ASRL2` accèdent aux ports série 1 et 2 via `/dev/tty00` et `/dev/tty01` sur HP-UX9. HP-UX10 utilise `/dev/tty0p0` et `/dev/tty1p0`. Les ports supplémentaires sont numérotés dans l'ordre en commençant par ASRL3, qui utilise `/dev/tty02`.

Support VME

Pour accéder aux périphériques VME dans votre système, vous devez configurer NI-VXI pour voir ces périphériques. Les utilisateurs Windows 95/NT peuvent configurer NI-VXI en utilisant la fonction **Add Device Wizard** dans T&M Explorer. Les utilisateurs d'autres plates-formes doivent utiliser **Non-VXI Device Editor** dans VXIedit ou VXItedit. Pour chaque espace d'adresse dans lequel votre périphérique possède de la mémoire, vous devez créer une entrée de pseudo-périphérique séparée avec une adresse logique comprise entre 256 et 511. Par exemple, un périphérique VME avec de la mémoire à la fois dans les espaces A24 et A32 nécessite deux entrées. Vous pouvez aussi spécifier les niveaux d'interruption utilisés par le périphérique. Les périphériques VXI et VME ne peuvent pas partager des niveaux d'interruption. Vous pouvez alors accéder à ce périphérique depuis NI-VISA, tout comme vous le feriez avec un périphérique VXI, mais en spécifiant l'espace d'adresse et l'offset de base à laquelle vous l'avez configuré. Le support NI-VISA pour les périphériques VME inclut les opérations d'accès aux registres (de haut et de bas niveaux) et les opérations de déplacement de blocs, comme la capacité à recevoir des interruptions.

Mise au point d'un programme VISA

Cette section contient des informations sur la mise au point de programmes VISA. A cause de la nature de VISA, il y a davantage de possibilités à considérer lors de la mise au point d'un programme utilisant les fonctions VISA que lors d'un travail avec des drivers autonomes. VISA effectue des appels NI-VXI, NI-488, ou des API série du système d'exploitation. Par conséquent, les problèmes qui apparaissent dans VISA peuvent être liés au driver appelé par VISA, et non à VISA lui-même.

Si aucun VI VISA ne semble fonctionner dans LabVIEW (y compris les drivers d'instruments), la première étape est d'utiliser la fonction VISA Find Resource. Cette fonction s'exécute sans autre VI VISA dans le diagramme. Si cette fonction génère des erreurs étranges, telles que des erreurs VISA non standard, le problème est probablement qu'une mauvaise version de VISA est installée ou que VISA n'est pas installé correctement. Si VISA Find Resource s'exécute correctement, cela signifie que LabVIEW fonctionne correctement avec le driver VISA. La prochaine étape est d'identifier la séquence des VIs qui produit l'erreur dans le programme LabVIEW.

Si c'est une séquence d'événements simple qui produit l'erreur, il est judicieux, pour la mise au point, d'essayer la même séquence interactivement avec l'utilitaire VISAIC (voir la prochaine section). C'est en général une bonne idée de faire un développement de programme initial de façon interactive. Si l'utilitaire interactif fonctionne correctement, mais si ce n'est pas le cas pour la même séquence dans LabVIEW, il se peut que LabVIEW ait un problème d'interaction avec le driver VISA. Si la même séquence montre les mêmes problèmes dans VISAIC, il est possible qu'un problème existe avec l'un des drivers appelés par VISA. Vous pouvez utiliser les utilitaires interactifs pour ces drivers (VIC pour NI-VXI et IBIC pour NI-488.2) pour essayer de réaliser les opérations équivalentes. Si les problèmes persistent à ce niveau, cela signifie qu'il y a peut-être un problème avec le driver du plus bas niveau ou avec son installation.

Outil de mise au point sous Windows 95/NT

NI Spy répertorie les appels émis par votre application aux drivers de tests et de mesures (T&M) de National Instruments, y compris NI-VISA, NI-VXI et NI-488.2. Les utilisateurs NI-488.2 peuvent remarquer que NI Spy est similaire à GPIB Spy.

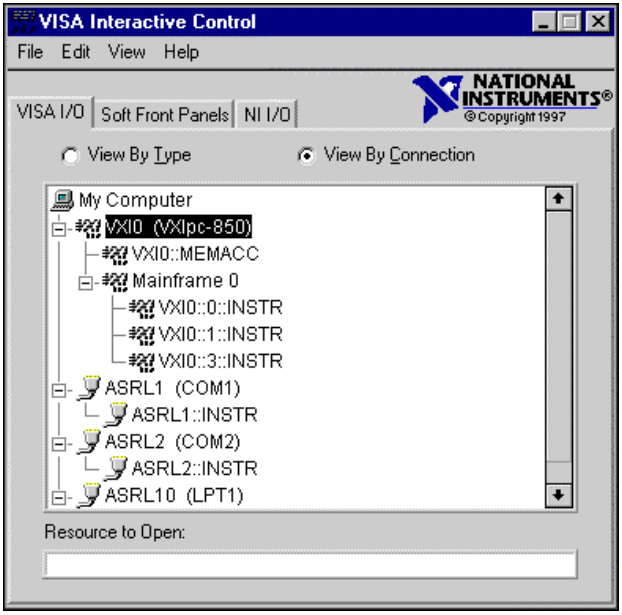
NI Spy met en avant les fonctions qui retournent les erreurs, vous pouvez donc rapidement déterminer les fonctions qui ont échoué pendant votre

développement. NI Spy peut aussi enregistrer les appels de votre programme à ces drivers, afin de pouvoir les vérifier et tenter d'y déceler des erreurs.

VISAIC

VISA est fourni avec un utilitaire appelé VISA Interactive Control (VISAIC) sur toutes les plates-formes supportant VISA et LabVIEW, sauf Macintosh. Cet utilitaire fournit un accès à toutes les fonctionnalités de VISA interactivement, dans un environnement graphique convivial. C'est un point de départ pratique pour le développement de programme et l'apprentissage de VISA.

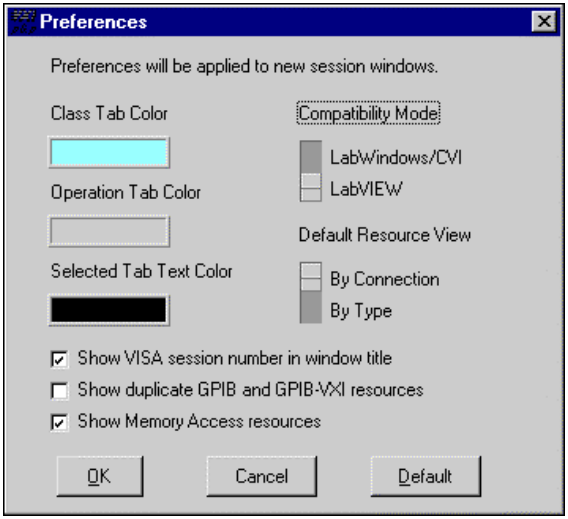
Lorsque VISAIC s'exécute, il trouve automatiquement toutes les ressources disponibles dans le système et liste les descripteurs d'instruments pour chacune de ces ressources, sous le type de ressources approprié. La fenêtre d'ouverture VISAIC est présentée dans la figure ci-dessous.



L'onglet Soft Front Panels du panneau VISAIC principal vous donne l'option de lancer les faces-avant de tous les drivers d'instruments VXI*plug&play* installés sur le système.

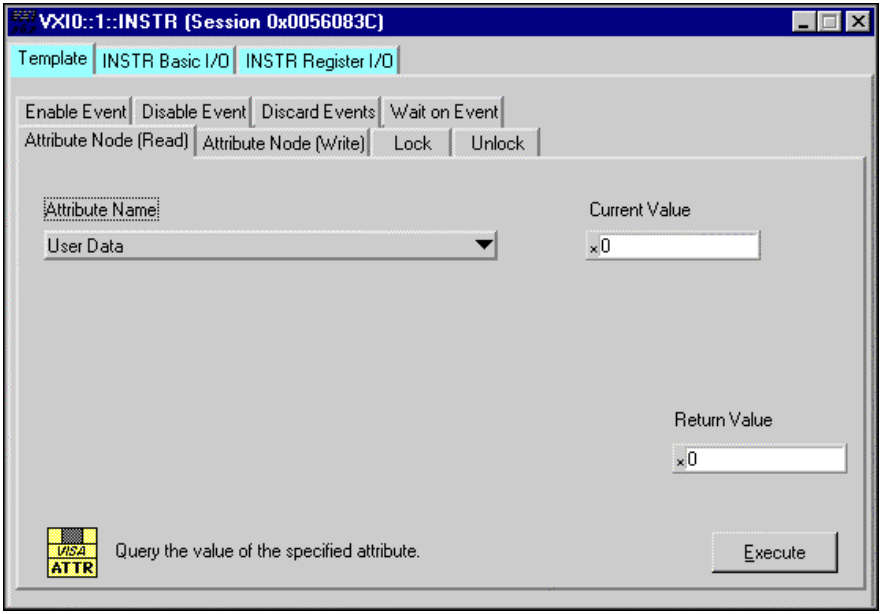
L'onglet NI I/O vous donne l'option de lancer l'utilitaire interactif NI-VXI ou l'utilitaire interactif NI-488. Ceci vous donne des liens pratiques avec les utilitaires interactifs pour les drivers appelés par VISA.

Double-cliquer sur l'un des descripteurs d'instruments affichés dans la fenêtre VISAIC ouvre une session à cet instrument. L'ouverture d'une session vers l'instrument génère une fenêtre avec une série d'onglets, pour l'exécution interactive des commandes VISA. L'apparence exacte de ces onglets dépend du mode de compatibilité dans lequel se trouve VISAIC. Pour accéder au mode de compatibilité et à d'autres préférences VISAIC, sélectionnez **Edit»Preferences…** pour ouvrir la fenêtre ci-dessous.



Les implémentations VISA sont légèrement différentes dans LabVIEW et LabWindows/CVI. Ces différences sont reflétées dans les onglets d'opération qui s'affichent quand vous ouvrez une session vers une ressource. Le mode de compatibilité est établi par défaut sur LabWindows/CVI, mais vous devez faire passer ce mode sur LabVIEW. Une fois que les préférences sont changées, les nouvelles préférences prennent effet pour toute session ouverte ultérieurement.

Lorsque la session d'une ressource est ouverte de façon interactive, une fenêtre similaire à celle présentée ci-dessous apparaît.



Il y a trois onglets principaux qui apparaissent dans la fenêtre. L'onglet initial est l'onglet Template qui contient toutes les opérations traitant des événements, des propriétés et des verrous. Remarquez qu'il y a un onglet différent pour chacune de ces opérations à l'intérieur de l'onglet principal. Les autres onglets principaux sont INSTR Basic I/O et INSTR Register I/O. L'onglet Basic I/O contient les opérations de base pour les instruments basés messages alors que l'onglet Register I/O contient les opérations de base pour les instruments basés registres. L'onglet Register I/O apparaît uniquement pour les instruments VXI.

9

Introduction aux fonctions GPIB de LabVIEW

Ce chapitre explique le fonctionnement du bus d'interface universel (GPIB), et la différence entre les interfaces IEEE 488 et IEEE 488.2.

Il existe deux standards GPIB : IEEE 488 et IEEE 488.2. Hewlett-Packard a conçu le bus GPIB (appelé au départ HP-IB) pour interconnecter et contrôler sa ligne d'instruments programmables. Le GPIB a rapidement été utilisé pour d'autres applications, telles que la communication entre ordinateurs et le contrôle de périphérique, grâce à ses taux de transfert de données de 1 Mo/s maximum. Il a été plus tard accepté comme le standard IEEE 488-1975 et a depuis évolué vers le standard ANSI/ IEEE 488.2-1987. La versatilité du système lui a donné le nom de bus d'interface universel.

National Instruments a amené le GPIB aux utilisateurs d'ordinateurs et de périphériques non Hewlett-Packard se spécialisant dans les interfaces de matériel à grande vitesse et haute performance, et dans les logiciels utilisant la totalité des fonctions. Les fonctions GPIB dans LabVIEW suivent la spécification IEEE 488.2.

Types de messages

Le bus GPIB transporte des messages dépendant de l'appareil et des messages d'interface.

- Les messages dépendant de l'appareil, souvent appelés *données* ou *messages de données*, contiennent des informations spécifiques à l'appareil telles que les instructions de programmation, les résultats des mesures, l'état de la machine et les fichiers de données.
- Les messages d'interface gèrent le bus lui-même. Ils sont en général appelés *commandes* ou *messages de commande*. Les messages d'interface réalisent des tâches telles que l'initialisation du bus, l'adressage et le désadressage d'appareils, et la définition des modes de l'appareil pour la programmation locale ou à distance.

Ne confondez pas le terme *commande* utilisé ici avec des instructions de certains appareils, qui peuvent aussi être appelées des commandes. Ces instructions spécifiques à l'appareil sont en réalité des messages de données.

Le standard 488.2-1987 ANSI/IEEE étendu sur le standard précédent IEEE 488.1 pour décrire exactement comment le contrôleur doit gérer le bus GPIB, y compris les messages standard que les appareils compatibles doivent savoir interpréter, les mécanismes pour le retour d'erreurs d'appareil et d'autres informations d'état, et les différents protocoles qui découvrent et configurent les appareils compatibles connectés au bus.

Le standard IEEE 488.2 permet de contrôler les lignes de bus à tout moment. Cette capacité est cruciale pour la détection d'appareils actifs (talker et listener) sur le bus GPIB. Les appareils GPIB peuvent être talkers, listeners et/ou contrôleurs. Un voltmètre numérique, par exemple, est un talker et peut aussi devenir un listener. Un talker envoie des messages de données à un ou plusieurs listeners. Le contrôleur gère le flux d'informations sur le bus GPIB en envoyant des commandes à tous les appareils.

Le bus GPIB est comme un bus d'ordinateur ordinaire, sauf que l'ordinateur a ses cartes de circuit interconnectées via un bus fond de panier, alors que le GPIB a des appareils autonomes interconnectés via un bus câble.

Le rôle du contrôleur GPIB est semblable à celui de l'unité centrale d'un ordinateur, mais une meilleure analogie peut être faite avec le centre de commutation d'un système de téléphone urbain. Le centre de commutation (contrôleur) contrôle le réseau de communication (GPIB). Lorsque le centre (contrôleur) remarque qu'un abonné (appareil) veut appeler (envoyer un message de données), il connecte l'appelant (talker) à l'appelé (listener).

Le contrôleur adresse un talker et un listener avant que le talker ne puisse envoyer son message à le listener. Après que le talker a transmis le message, le contrôleur peut désadresser les deux appareils.

Certaines configurations de bus n'ont pas besoin d'un contrôleur. Par exemple, un appareil peut toujours être un talker (appareil talk-only) et il peut y avoir un ou plusieurs appareils listen-only.

Un contrôleur est nécessaire lorsque vous devez changer le talker ou le listener adressé actif. Un ordinateur assure habituellement la fonction du contrôleur.

Avec la carte GPIB et son logiciel, votre ordinateur personnel joue ces trois rôles :

- Contrôleur : pour gérer le bus GPIB
- Talker : pour envoyer des données
- Listener : pour recevoir des données

Le contrôleur en charge et le contrôleur système

Bien qu'il puisse y avoir plusieurs contrôleurs sur le bus GPIB, un seul contrôleur à la fois est actif. Ce contrôleur est appelé contrôleur en charge (CIC). Vous pouvez changer le contrôle actif du CIC actuel à un contrôleur inactif. Seul un appareil sur le bus – le contrôleur système – peut se changer lui-même en CIC. La carte GPIB est en général le contrôleur système.

Matériel GPIB compatible

Les produits GPIB National Instruments suivants sont compatibles avec LabVIEW :

LabVIEW pour Windows 95 et Windows 95 japonais

- AT-GPIB/TNT, AT-GPIB/TNT (PnP), AT-GPIB/TNT+, PCI-GPIB
- PCMCIA-GPIB, PCMCIA-GPIB+
- GPIB-ENET
- EISA-GPIB
- VXIpc modèle 850
- NEC-GPIB/TNT, NEC-GPIB/TNT (PnP)
- GPIB-PCII/IIA
- PC/104-GPIB
- CPCI-GPIB
- GPIB-ENET
- PMC-GPIB

LabVIEW pour Windows NT

- AT-GPIB, AT-GPIB/TNT
- PCMCIA-GPIB
- PCI-GPIB
- VXIpc modèle 850
- GPIB-ENET

LabVIEW pour Windows 3.1

- AT-GPIB, AT-GPIB/TNT, AT-GPIB/TNT (PnP), AT-GPIB/TNT+, PCI-GPIB
- PCMCIA-GPIB, PCMCIA-GPIB+
- GPIB-ENET
- EISA-GPIB
- VXIpc modèle 850
- NEC-GPIB/TNT (japonais), NEC-GPIB/TNT (PnP) (japonais), GPIB-PCII/IIA
- GPIB-232CT-A
- GPIB-485CT-A
- GPIB-1284CT
- PCII/IIA
- STD-GPIB
- EXM-GPIB
- MC-GPIB

LabVIEW pour Mac OS

- PCI-GPIB
- NB-GPIB/TNT, NB-GPIB-P/TNT
- PCMCIA-GPIB
- LC-GPIB
- GPIB-ENET
- GPIB-232CT-A
- GPIB-SCSI-A

- PC/104-GPIB
- NB-DMA2800 (seulement les fonctions GPIB classiques)

LabVIEW pour HP-UX

- GPIB-ENET
- EISA-GPIB
- AT-GPIB/TNT

LabVIEW pour Sun

- GPIB-ENET
- GPIB-SCSI-A
- SB-GPIB/TNT

LabVIEW pour Concurrent PowerMAX

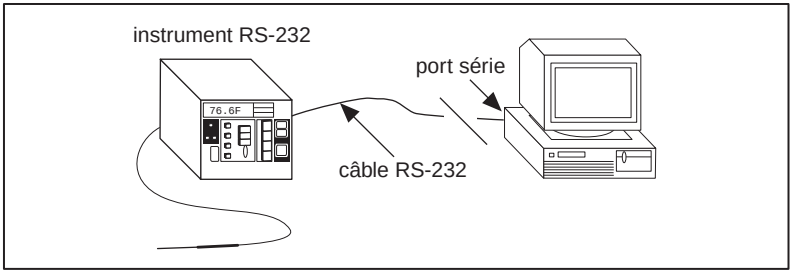
- GPIB-1014
- GPIB-1014D
- GPIB-1014P
- GPIB-1014DP

10

VIs de port série

Ce chapitre décrit les VIs pour des opérations utilisant le port série et explique les facteurs importants qui affectent la communication série.

La communication série est un moyen populaire de transmission de données entre un ordinateur et un périphérique comme un instrument programmable ou même un autre ordinateur. La communication série utilise un transmetteur pour envoyer des données, un bit à la fois, sur une ligne de communication unique, à un récepteur. Vous pouvez utiliser cette méthode lorsque les taux de transfert de données sont faibles ou lorsque vous devez transférer des données sur de longues distances.



La communication série est très utilisée parce que la plupart des ordinateurs ont un ou deux ports série. Beaucoup d'instruments GPIB sont aussi disponibles avec des ports série. La communication série est limitée, cependant, dans le sens où un port série peut communiquer avec un seul périphérique.

Certains périphériques requièrent des caractères pour terminer les chaînes de données qui leur sont envoyées. Le retour chariot, le retour à la ligne ou le point-virgule sont des caractères de terminaison classiques. Consultez le manuel du périphérique pour déterminer si un caractère de terminaison est nécessaire.

Pour des exemples d'utilisation des VIs de ports série, reportez-vous à la bibliothèque `examples\instr\smplserl.llb`.

Modes de handshaking

Un problème courant avec les communications série est de s'assurer que l'expéditeur et le destinataire sont à la même cadence que la transmission de données. Le driver de port série peut bufférisier des informations entrantes/sortantes, mais ce buffer est de taille finie. Quand le buffer est plein, l'ordinateur ignore les nouvelles données jusqu'à ce que vous ayez lu assez de données dans le buffer pour faire de la place à de nouvelles informations.

Le handshaking aide à prévenir le débordement du buffer. Avec le handshaking, l'expéditeur et le destinataire se signalent l'un à l'autre quand leur buffer est rempli. L'expéditeur peut alors cesser d'envoyer de nouvelles informations jusqu'à ce que l'autre extrémité de la communication série soit prête pour en recevoir.

Vous pouvez réaliser deux sortes de handshaking dans LabVIEW : un handshaking logiciel et un handshaking de matériel. Vous pouvez ouvrir ou fermer ces formes de handshaking en utilisant le VI "Initialiser le port série" (Serial Port Init.VI). Par défaut, les VIs n'utilisent pas de handshaking.

Handshaking logiciel – XON/XOFF

XON/XOFF est un protocole de handshaking logiciel que vous pouvez utiliser pour éviter le débordement des buffers du port série. Lorsque le buffer récepteur est presque plein, le récepteur envoie XOFF (<Ctrl-S> [décimal 19]) pour signaler à l'autre périphérique d'arrêter d'envoyer des données. Lorsque le buffer récepteur est suffisamment vide, le récepteur envoie XON (<Ctrl-Q> [décimal 17]) pour indiquer que la transmission peut recommencer. Quand vous activez le mode XON/XOFF, les périphériques interprètent toujours <Ctrl-Q> et <Ctrl-S> comme des caractères XON et XOFF, jamais comme des données. Lorsque vous désactivez le mode XON/XOFF, vous pouvez envoyer les valeurs <Ctrl-Q> et <Ctrl-S> comme données. N'utilisez pas XON/XOFF avec des transferts de données binaires car les caractères <Ctrl-Q> ou <Ctrl-S> peuvent être contenus dans les données, et les périphériques les interpréteront comme XON et XOFF plutôt que comme des données.

Codes d'erreur

Vous pouvez connecter le paramètre **code d'erreur** à l'un des VIs de gestion d'erreurs. Ces VIs peuvent décrire l'erreur et vous donner des indications sur la façon de procéder lorsqu'une erreur survient.

Certains codes d'erreur retournés par les VIs de port série sont spécifiques à des plates-formes. Veuillez vous référer à la documentation de votre système pour une liste des codes d'erreur.

Numéro de port

Windows 95/NT et 3.x

Lorsque vous utilisez les VIs de port série sous Windows 95/NT et Windows 3.*x*, le paramètre **numéro de port** peut prendre les valeurs suivantes :

0 : COM1	5 : COM6	10 : LPT1
1 : COM2	6 : COM7	11 : LPT2
2 : COM3	7 : COM8	12 : LPT3
3 : COM4	8 : COM9	13 : LPT4
4 : COM5		

Lorsque vous utilisez les VIs de port série sous Windows 95 ou Windows NT, le paramètre **numéro de port** est 0 pour COM1, 1 pour COM2, etc.

Macintosh

Sur Macintosh, le port 0 est le modem utilisant les drivers `.ain` et `.aout`. Le port 1 est l'imprimante utilisant les drivers `.bin` et `.bout`. Pour avoir plus de ports sur un Macintosh, vous devez installer d'autres cartes et les drivers qui les accompagnent.

UNIX

Sur Sun SPARCstation, sous Solaris 1 et sur Concurrent PowerMAX, le paramètre **numéro de port** pour les VIs de port série est 0 pour `/dev/ttya`, 1 pour `/dev/ttyb`, etc. Sous Solaris 2, le port 0 fait référence à `/dev/cua/a`, 1 à `/dev/cua/b`, etc. Sous HP-UX, le numéro de port 0 fait référence à `/dev/tty00`, 1 à `/dev/tty01`, etc.

Sur Concurrent PowerMAX, le port 0 fait référence à `/dev/console`, le port 1 à `/dev/tty1`, le port 2 à `/dev/tty2`, etc.

Comme des cartes de port série provenant d'autres revendeurs peuvent avoir des noms de périphériques arbitraires, LabVIEW a développé une interface facile pour que le numérotage des ports reste simple. Dans LabVIEW pour Sun, HP-UX et Concurrent PowerMAX, une option de configuration existe pour indiquer à LabVIEW comment adresser les ports série. LabVIEW supporte toute carte qui utilise les périphériques UNIX standard. Certains fabricants conseillent d'utiliser les nœuds de périphériques `cua` plutôt que les nœuds `tty` avec leurs cartes. LabVIEW peut adresser ces deux types de nœuds.

Le fichier `.labviewrc` contient les options de configuration LabVIEW. Pour définir les périphériques que les VIs de ports série utilisent, définissez l'option de configuration `labview.serialDevices` pour la liste des périphériques que vous avez l'intention d'utiliser.

Par exemple, le paramètre par défaut est :

```
labview.serialDevices:/dev/ttya:/dev/ttyb:/dev/
ttyc:...:/dev/ttyz.
```

☞ **Remarque** ***Il faut pour cela que toute installation de carte série d'un autre fabricant comprenne une méthode de création de fichier (nœud)*** `/dev` ***standard, et que l'utilisateur connaisse le nom de ce fichier.***

Partie III

Analyse

Cette section contient des informations de base sur l'analyse des données post-acquisition, le traitement et la génération de signal, l'algèbre linéaire, l'ajustement de courbe, la probabilité et les statistiques.

La Partie III, *Analyse*, contient les chapitres suivants.

- Le chapitre 11, *Introduction à l'analyse dans LabVIEW*, introduit des concepts qui s'appliquent à toutes les applications d'analyse, notamment la fonctionnalité supportée, les conventions de notation et d'appellation, et les méthodes d'échantillonnage de signal.
- Le chapitre 12, *Génération de signaux*, explique comment produire des signaux utilisant la fréquence normalisée et comment construire un générateur de fonction simulée.
- Le chapitre 13, *Traitement des signaux numériques*, décrit les bases de la transformée rapide de Fourier (FFT) et de la transformée discrète de Fourier (DFT), et leur utilisation dans l'analyse de spectre.
- Le chapitre 14, *Fenêtres de lissage*, explique comment l'utilisation de fenêtres prévient la fuite spectrale et améliore l'analyse de signaux acquis.
- Le chapitre 15, *Analyse et mesure de spectre*, indique comment déterminer l'amplitude et le spectre de phase, développer un analyseur de spectre et calculer la distorsion harmonique totale (THD) de vos signaux.
- Le chapitre 16, *Filtrage*, explique comment filtrer des fréquences non désirées depuis des signaux utilisant des filtres à réponse impulsionnelle infinie (RII), des filtres de réponse à réponse impulsionnelle finie (RIF) et des filtres non linéaires.
- Le chapitre 17, *Ajustement de courbe*, présente comment extraire des informations dans un ensemble de données pour obtenir une description fonctionnelle.

- Le chapitre 18, *Algèbre linéaire*, explique comment réaliser l'analyse et le calcul de matrice.
- Le chapitre 19, *Probabilités et statistiques*, explique des concepts fondamentaux de probabilité et de statistiques, et montre comment utiliser ces concepts pour la résolution de problèmes du monde réel.

11

Introduction à l'analyse dans LabVIEW

Les signaux numériques sont partout dans le monde autour de nous. Les compagnies de téléphone utilisent les signaux numériques pour représenter la voix humaine. La radio, la télévision et les chaînes haute-fidélité se convertissent toutes petit à petite au numérique à cause de sa fidélité supérieure, de la réduction des signaux parasites et de la souplesse du traitement de signal. Les données sont transmises depuis des satellites vers des stations terrestres sous forme numérique. Les photos de la NASA de planètes éloignées et de l'espace extra-atmosphérique sont souvent traitées numériquement pour éliminer les signaux parasites et extraire les informations utiles. Les données économiques, les résultats du recensement et les cours de la Bourse sont tous disponibles sous forme numérique. Grâce aux nombreux avantages du traitement de signal numérique, les signaux analogiques sont aussi convertis au format numérique avant d'être traités par un ordinateur.

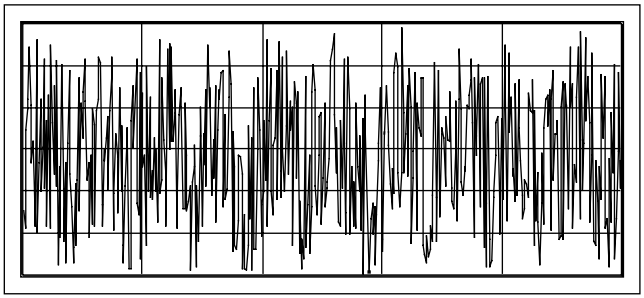
Ce chapitre fournit des éléments fondamentaux du traitement de signaux numériques de base ainsi qu'une introduction à la bibliothèque d'analyse LabVIEW, qui contient des centaines de VIs destinés à l'analyse et au traitement de signal.

L'importance de l'analyse des données

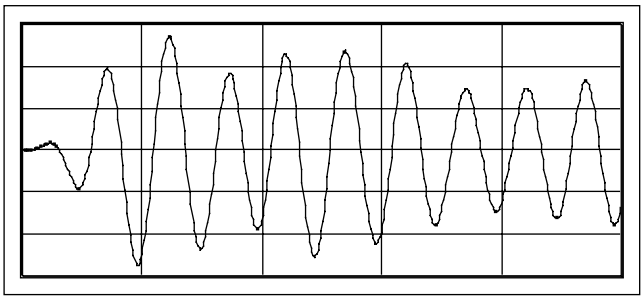
Les processeurs modernes de signaux numériques à virgule flottante, à haute vitesse, sont devenus de plus en plus importants pour les systèmes d'analyse en temps réel. Parmi les nombreuses applications possibles, nous trouvons le traitement de données biomédicales, la reconnaissance et la synthèse vocales, et le traitement de l'image et du son numériques.

L'intégration des bibliothèques d'analyse dans les stations d'ingénierie est importante dans le sens où les données *brutes*, comme présentées dans la figure suivante, ne comportent pas toujours des informations utiles immédiatement. Vous devez souvent transformer le signal, retirer le bruit,

corriger les données altérées par un équipement défectueux ou compenser les effets de l'environnement, comme la température et l'humidité.



En analysant et en traitant les données numériques, vous pouvez extraire les informations utiles des signaux parasités et présenter ces informations sous une forme plus claire que celle des données brutes. La figure suivante représente les données traitées.



L'approche de programmation en diagramme et l'ensemble étendu de VIs d'analyse de LabVIEW simplifient le développement des applications d'analyse.

Les VIs d'analyse LabVIEW vous offrent les techniques d'analyse de données les plus récentes, avec des VIs que vous pouvez câbler ensemble. Au lieu de vous soucier d'implémenter des détails pour des routines d'analyse, comme vous le faites dans des langages de programmation conventionnels, vous pouvez vous concentrer sur la résolution de vos problèmes d'analyse de données.

Système de développement complet

La bibliothèque de VIs d'analyse de base est un sous-ensemble de la bibliothèque de VIs d'analyse avancée. La bibliothèque d'analyse de base comprend des VIs pour l'analyse statistique, l'algèbre linéaire et l'analyse numérique. La bibliothèque d'analyse avancée comprend davantage de VIs dans ces domaines, de même que des VIs pour la génération de signal, des algorithmes des domaines fréquentiels et temporels, des routines de fenêtrage, des filtres numériques, des évaluations et des régressions.

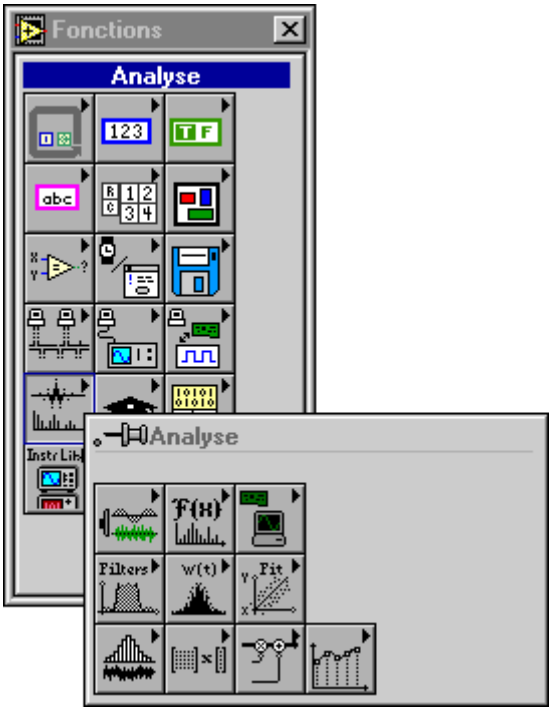
Si les VIs de la bibliothèque d'analyse de base ne répondent pas à vos besoins, vous pouvez ajouter les bibliothèques d'analyse avancée LabVIEW au package de base LabVIEW. Après la mise à niveau, vous aurez tous les outils d'analyse disponibles dans le système de développement complet.

Aperçu des VIs d'analyse

Une fois que le signal analogique a été converti au format numérique par le convertisseur analogique-numérique (C A/N) et qu'il est disponible dans votre ordinateur comme signal numérique (un ensemble d'échantillons), vous devez en général traiter ces échantillons d'une certaine façon. Le traitement peut être destiné à déterminer les caractéristiques du système depuis lequel vous avez obtenu les échantillons, à mesurer certaines fonctions du signal, ou à les convertir dans une forme compréhensible pour un opérateur.

La bibliothèque d'analyse contient des VIs pour réaliser des analyses numériques étendues, la génération et le traitement de signaux, l'ajustement de courbe, des mesures et d'autres fonctions d'analyse. La bibliothèque d'analyse, comprise dans le système de développement complet de LabVIEW, est une composante clé de la construction d'un système d'instrumentation virtuelle. En plus de contenir les fonctions d'analyse qu'on peut trouver dans de nombreux progiciels mathématiques, elle comprend également de nombreuses fonctions de mesure et de traitement de signaux uniques, conçues exclusivement pour l'industrie de l'instrumentation.

Les VIs d'analyse LabVIEW sont disponibles dans la sous-palette **Analyse** de la palette **Fonctions** dans LabVIEW ou BridgeVIEW.



Il existe dix bibliothèques de VIs d'analyse. Les principales sont :

Génération de signal : VIs générant des formes d'onde et des motifs numériques.

Traitement des signaux numériques : VIs réalisant des transformations dans le domaine fréquentiel, l'analyse dans le domaine fréquentiel et temporel, et d'autres transformées telles que les transformées de Hartley et Hilbert.

Fonctions de mesure : VIs réalisant des fonctions orientées mesures telles que des spectres asymétriques, des fenêtrages mis à l'échelle, et l'estimation de pics de puissance et de fréquence.

Filtres numériques : VIs réalisant des fonctions de filtrage numérique non linéaire, RII et RIF.

Fonctions de fenêtrage : VIs réalisant le fenêtrage des données.

Fonctions statistiques et de probabilité : VIs réalisant des fonctions de statistique descriptive, telles que l'identification de la moyenne ou de l'écart type d'un ensemble de données, de même que des fonctions statistiques inférentielles pour la probabilité et l'analyse de variance (ANOVA).

Fonctions d'ajustement de courbe : VIs réalisant des fonctions d'ajustement de courbe et des interpolations.

Fonctions d'algèbre linéaire : VIs réalisant des fonctions algébriques sur des vecteurs et des matrices complexes et réels.

Opérations sur les tableaux : VIs réalisant des opérations classiques sur tableaux numériques à une ou deux dimensions, telles que l'évaluation linéaire et la mise à l'échelle.

Méthodes numériques supplémentaires : VIs utilisant des méthodes numériques pour réaliser la détection de pics et l'intégration numérique avec recherche de racines.

Dans ces chapitres, vous apprendrez à utiliser les VIs de la bibliothèque d'analyse pour construire un générateur de fonctions et un analyseur de spectre simple mais pratique. Vous apprendrez aussi à utiliser des filtres numériques, l'objectif du fenêtrage et les avantages des différents types de fenêtres, la façon de réaliser des tâches d'ajustement de courbe simples, et d'autres choses encore. Les exercices de ces chapitres ont besoin du système de développement complet LabVIEW/BridgeVIEW. Pour les plus aventureux, davantage d'exemples d'utilisation des VIs d'analyse se trouvent dans le dossier `labview\examples\analysis`.

Outre la bibliothèque d'analyse, National Instruments offre aussi de nombreuses options d'analyse qui font de LabVIEW un des logiciels d'analyse les plus puissants disponibles sur le marché. Ces options incluent le *Joint Time-Frequency Analysis Toolkit*, avec l'algorithme primé du spectrogramme de Gabor de National Instruments qui analyse les caractéristiques fréquence-temps (fonctions difficiles à obtenir par une analyse de Fourier conventionnelle) ; le *G Math Toolkit* qui offre des fonctions mathématiques étendues comme, entre autres, l'analyse grammaticale de formule, les routines pour l'optimisation et la résolution d'équations différentielles, de nombreux types de tracés 2D et 3D ; le *Digital Filter Design Toolkit* ; et beaucoup d'autres choses encore.

Conventions d'appellation et de notation

Pour vous aider à identifier le type de paramètres et des opérations, cette section du manuel utilise les conventions d'appellation et de notation suivantes, sauf spécification contraire dans une description de VI. Bien qu'il y ait peu de fonctions et d'opérations scalaires, la plupart des VIs d'analyse traitent de larges blocs de données sous la forme de tableaux uni-dimensionnels (ou vecteurs) et de tableaux bi-dimensionnels (ou matrices).

Les lettres en minuscules normales représentent des scalaires ou des constantes. Par exemple :

$$a,$$
$$\pi,$$
$$b = 1{,}234.$$

Les lettres capitales représentent des tableaux. Par exemple :

$$X,$$
$$A,$$
$$Y = a\,X + b.$$

En général, *X* et *Y* représentent des tableaux 1D, et *A*, *B* et *C* représentent des matrices.

Les indices de tableaux dans LabVIEW sont à base zéro. C'est-à-dire que l'indice du premier élément d'un tableau, quelle que soit sa dimension, est zéro. La séquence de nombres suivante représente un tableau 1D *X* contenant *n* éléments.

$$X = \{x_0, x_1, x_2, \ldots, x_{n-1}\}$$

La quantité scalaire suivante représent le $i^{\text{ème}}$ élément de la séquence *X*.

$$x_i, \quad 0 \le i < n$$

Le premier élément de la séquence est x_0 et le dernier élément de la séquence est x_{n-1}, pour un total de *n* éléments.

La séquence de nombres suivante représente un tableau 2D contenant *n* lignes et *m* colonnes.

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0m-1} \\ a_{10} & a_{11} & a_{12} & \cdots & a_{1m-1} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-10} & a_{n-11} & a_{n-2} & \cdots & a_{n-1m-1} \end{bmatrix}$$

Le nombre total d'éléments dans le tableau 2D est le produit de *n* par *m*. Le premier indice correspond au numéro de ligne, et le second indice correspond au numéro de colonne. La quantité scalaire suivante représente l'élément situé sur la $i^{ème}$ ligne et la $j^{ème}$ colonne.

$$a_{ij},\ 0 \le i < n \text{ et } 0 \le j < m$$

Le premier élément dans *A* est $a_{0\,0}$ et le dernier élément est $a_{n-1\,m-1}$.

Sauf spécification contraire, ce manuel utilise les notations d'opération de tableau simplifiées suivantes.

La définition des éléments d'un tableau sur une constante scalaire est représentée par

$$X = a,$$

qui correspond à la séquence

$$X = \{a, a, a, \ldots, a\}$$

et est utilisée à la place de

$$x_i = a,$$

pour

$$i = 0, 1, 2, \ldots, n-1.$$

La multiplication des éléments d'un tableau par une constante scalaire est représentée par

$$Y = a\,X,$$

qui correspond à la séquence

$$Y = \{a\,x_0, a\,x_1, a\,x_2, \ldots, a\,x_{n-1}\}$$

et est utilisée à la place de

$$y_i = a\,x_i,$$

pour

$$i = 0, 1, 2, \ldots, n-1.$$

De la même façon, la multiplication d'un tableau 2D par une constante scalaire est représentée par

$$B = k\,A,$$

qui correspond à la séquence

$$B = \begin{bmatrix} ka_{00} & ka_{01} & ka_{02} & \ldots & ka_{0m-1} \\ ka_{10} & ka_{11} & ka_{12} & \ldots & ka_{1m-1} \\ ka_{20} & ka_{21} & ka_{22} & \ldots & ka_{2m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ka_{n-10} & ka_{n-11} & ka_{n-12} & \ldots & ka_{n-1m-1} \end{bmatrix}$$

et est utilisée à la place de

$$b_{ij} = k\,a_{ij},$$

pour

$$i = 0, 1, 2, \ldots, n-1$$

et

$$j = 0, 1, 2, \ldots, m-1.$$

Un tableau sans élément est un tableau vide et est représenté par

$$\text{Vide} = \text{NULL} = \varnothing = \{\ \}\ .$$

En général, les opérations sur des tableaux vides donnent des tableaux de sortie vides ou des résultats non définis.

Echantillonnage des données

Signaux d'échantillonnage

Pour utiliser les techniques de traitement des signaux numériques, vous devez d'abord convertir un signal analogique dans sa représentation numérique. En pratique, ceci est réalisé en utilisant un convertisseur analogique-numérique (C A/N). Considérez un signal analogique $x(t)$ qui est échantillonné toutes les Δt secondes. L'intervalle de temps Δt est appelé *intervall*e *d'échantillonnage* ou *période d'échantillonnage*. Son inverse, $1/\Delta t$, est appelé *fréquence d'échantillonnage*, avec pour unités le nombre d'échantillons/seconde. Chaque valeur discrète de x(t) à t = 0, Δt, $2\Delta t$, $3\Delta t$, etc., est appelée *échantillon*. Ainsi, x(0), x(Δt), x($2\Delta t$), ..., sont tous des échantillons. Le signal $x(t)$ peut donc être représenté par l'ensemble discret des échantillons

$$\{x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \ldots, x(k\Delta t), \ldots\}.$$

La figure 11-1 ci-dessous montre un signal analogique et sa version échantillonnée correspondante. L'intervalle d'échantillonnage est Δt. Observez que les échantillons sont définis à des points discrets dans le temps.

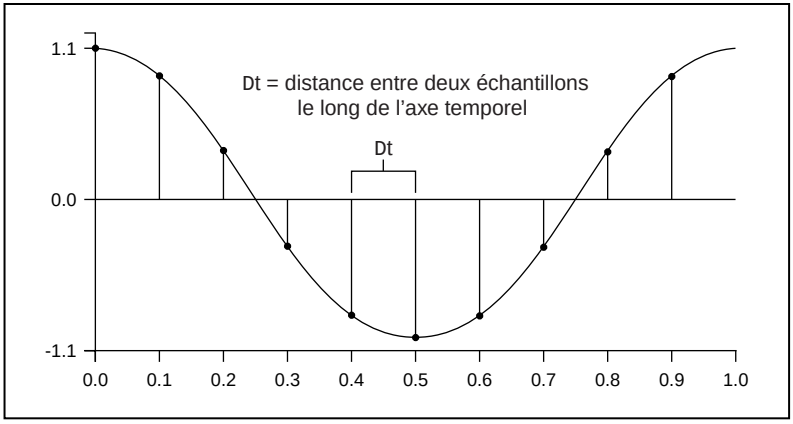


Figure 11-1. Signal analogique et signal échantillonné correspondant

La notation suivante représente les échantillons individuels :

$$x[i] = x(i\Delta t),$$

pour

$$i = 0, 1, 2, \ldots$$

Si n échantillons sont obtenus à partir du signal $x(t)$, alors $x(t)$ peut être représenté par la séquence

$$X = \{x[0], x[1], x[2], x[3], \ldots, x[N-1]\}$$

Ceci est appelé la *représentation numérique* de $x(t)$. Remarquez que la séquence $X = \{x[i]\}$ est indexée sur la variable entière i, et ne contient pas d'informations sur la fréquence d'échantillonnage. Aussi, en connaissant simplement les valeurs des échantillons contenus dans X, vous ne pouvez pas connaître la fréquence d'échantillonnage.

Considérations sur l'échantillonnage

Les convertisseurs analogique-numérique (C A/N) font partie intégrante des cartes DAQ de National Instruments. Un des paramètres les plus importants d'un système d'entrée analogique est la fréquence à laquelle la carte DAQ échantillonne un signal entrant. La fréquence d'échantillonnage détermine la fréquence de conversion analogique-numérique (A/N). Une fréquence d'échantillonnage rapide acquiert davantage de points dans un temps donné et peut ainsi souvent donner une meilleure représentation du signal original qu'une fréquence d'échantillonnage lente. Echantillonner trop lentement peut aboutir à une représentation pauvre de votre signal analogique. La figure 11-2 montre un signal échantillonné correctement, ainsi que les effets du sous-échantillonnage. Le sous-échantillonnage fait apparaître le signal avec une fréquence différente de celle qu'il a réellement. Cette représentation faussée d'un signal est appelée *repliement*.

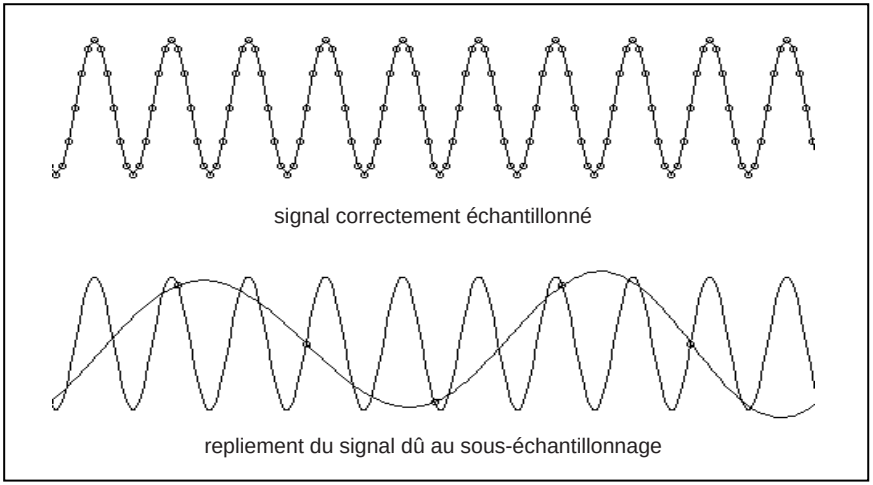


Figure 11-2. Effets du repliement avec une fréquence d'échantillonnage incorrecte

Selon le *théorème de* **Nyquist**, pour éviter le repliement, vous devez échantillonner à une fréquence au moins deux fois plus grande que la composante de la fréquence maximale du signal que vous voulez acquérir. Pour une fréquence d'échantillonnage donnée, la fréquence maximale pouvant être représentée précisément, sans repliement, est appelée fréquence de **Nyquist**. La fréquence de Nyquist est égale à la moitié de la fréquence d'échantillonnage. Les signaux contenant des composantes de fréquence supérieures à la fréquence de Nyquist sont reportés entre la fréquence des signaux continus (fréquence nulle) et la fréquence de Nyquist. La fréquence de repliement est la valeur absolue de la différence entre la fréquence du signal d'entrée et le multiple entier le plus proche de la fréquence d'échantillonnage. Les figures 11-3 et 11-4 illustrent ce phénomène. Par exemple, supposons que *f*s, la fréquence

d'échantillonnage, soit égale à 100 Hz. Supposons également que le signal d'entrée contienne les fréquences suivantes : 25 Hz, 70 Hz, 160 Hz et 510 Hz. Ces fréquences sont présentées dansla figure 11-3.

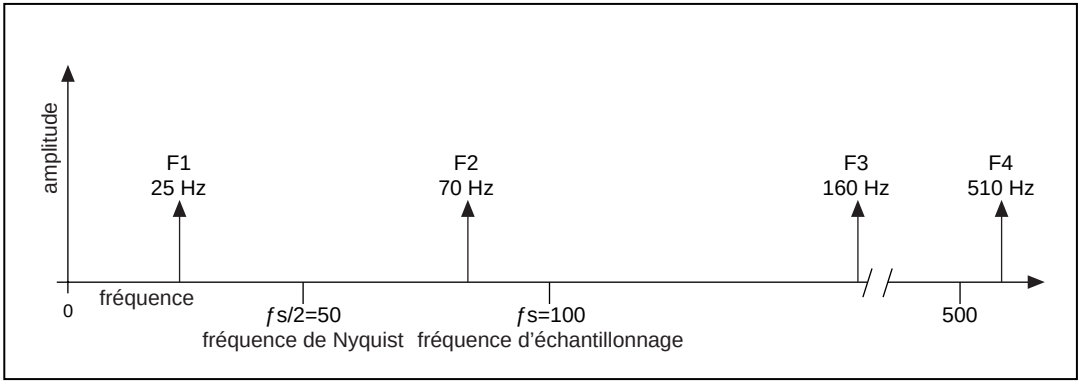


Figure 11-3. Composantes en fréquence d'un signal réel

Dans la figure 11-4, nous voyons que les fréquences inférieures à la fréquence de Nyquist (*f*s/2=50 Hz) sont échantillonnées correctement. Les fréquences supérieures à la fréquence de Nyquist apparaissent comme des fréquences repliées. Par exemple, F1 (25 Hz) apparaît à la fréquence correcte, mais F2 (70 Hz), F3 (160 Hz) et F4 (510 Hz) sont repliées et apparaissent respectivement à 30 Hz, 40 Hz et 10 Hz. Pour calculer la fréquence de repliement, utilisez l'équation suivante :

Fréquence de repliement = ABS (multiple entier le plus proche de la fréquence d'échantillonnage – fréquence d'entrée)

avec ABS signifiant "la valeur absolue". Par exemple :

Repli F2 = |100 – 70| = 30 Hz
Repli F3 = |(2)100 – 160| = 40 Hz
Repli F4 = |(5)100 – 510| = 10 Hz

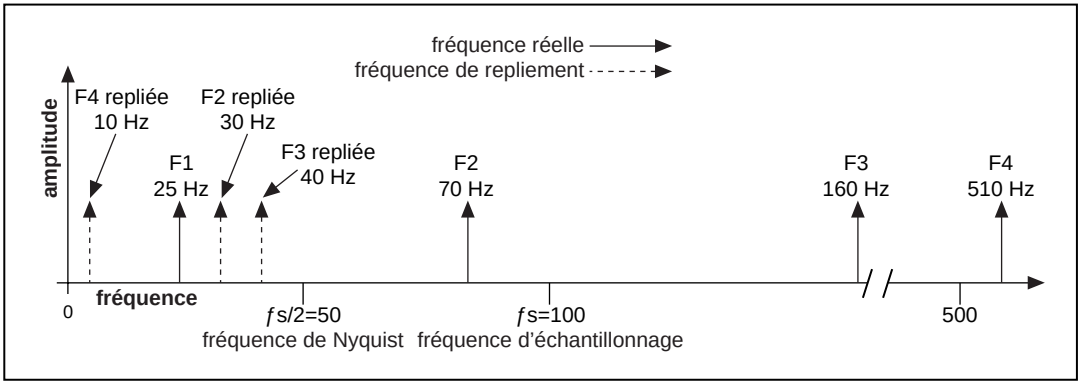


Figure 11-4. Composantes en fréquence d'un signal et repliements

Une question souvent posée est "à quelle vitesse dois-je échantillonner ?". Vous pouvez d'abord penser qu'il faut échantillonner à la fréquence maximum disponible sur votre carte DAQ. Cependant, si vous échantillonnez à des vitesses très rapides sur de longues périodes, vous n'aurez peut-être pas assez de mémoire sur votre disque dur pour contenir toutes les données. La figure 11-5 montre les effets de différentes fréquences d'échantillonnage. Dans le cas a, le signal sinusoïdal de fréquence f est échantillonné à la même fréquence fs (échantillons/s) = f (périodes/s), c'est-à-dire 1 échantillon par période. Le signal reconstruit apparaît comme un signal continu. En augmentant l'échantillonnage à 7 échantillons/4 périodes, comme dans le cas b, le signal augmente en fréquence, mais la fréquence est repliée sur une fréquence inférieure au signal original (3 périodes au lieu de 4). La fréquence d'échantillonnage dans le cas b est fs = 7/4 f. Si vous augmentez la fréquence d'échantillonnage à fs = 2f, le signal numérisé a la fréquence correcte (même nombre de périodes), et peut être reconstruit comme le signal sinusoïdal original, comme présenté dans le cas c. Pour le traitement de domaine temporel, il peut être important d'augmenter votre fréquence d'échantillonnage de manière à ce que les échantillons représentent plus fidèlement le signal original. En augmentant la fréquence d'échantillonnage bien au-dessus de f, disons à fs=10f, ou 10 échantillons

par période, vous pouvez reproduire précisément le signal, comme montré dans le cas d.

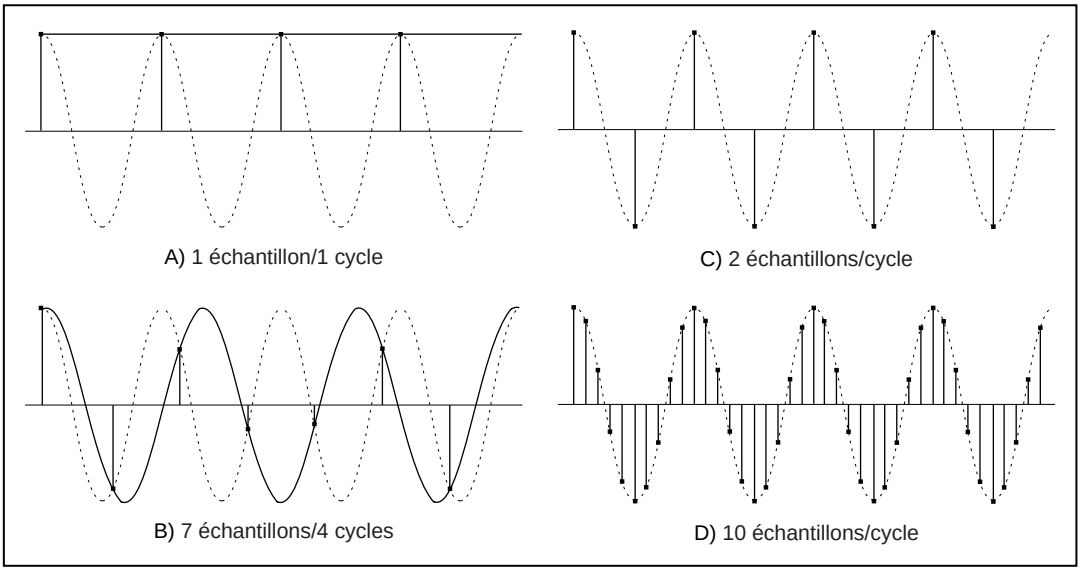


Figure 11-5. Effets de l'échantillonnage à des fréquences différentes

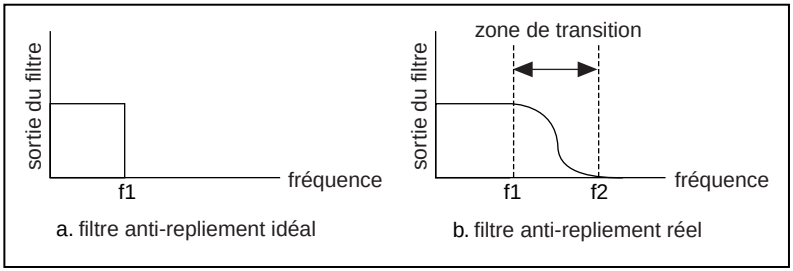
Pourquoi avez-vous besoin de filtres anti-repliement ?

Nous avons vu que la fréquence d'échantillonnage doit être égale à au moins deux fois la fréquence maximum du signal à échantillonner. En d'autres termes, la fréquence maximum du signal d'entrée doit être inférieure ou égale à la moitié de la fréquence d'échantillonnage. Mais comment s'assurer pratiquement que c'est vraiment le cas ? Même si vous êtes sûr que le signal mesuré admet une fréquence maximale, vous pouvez capter des signaux égarés (comme la fréquence de lignes d'alimentation ou de stations de radio locales) contenant des fréquence supérieures à la fréquence Nyquist. Ces fréquences peuvent alors se replier dans la gamme de fréquence désirée et fausser ainsi les résultats.

Pour être complètement sûr que le contenu de la fréquence du signal d'entrée est limité, un filtre passe bas (un filtre qui laisse passer les basses fréquences et atténue les hautes fréquences) est ajouté avant l'échantillonneur et le C A/N. Ce filtre est appelé un filtre *anti-repliement* parce qu'en atténuant les fréquences les plus élevées (supérieures à la fréquence de Nyquist), il empêche le repliement des fréquences élevées, et évite donc leur prise en compte au moment de l'échantillonnage. Comme à

ce stade (avant l'échantillonneur et le C A/N) nous sommes toujours dans l'sunivers analogique, le filtre anti-repliement est un filtre analogique.

Un filtre anti-repliement idéal est représenté dans la figure (a) ci-dessous.



Il laisse passer toutes les fréquences d'entrée désirées (au-dessous de *f1*) et rejette toutes les fréquences non désirées (au-dessus de *f1*). Cependant, un tel filtre n'est pas réalisable physiquement. En pratique, les filtres sont comme dans la figure (b) ci-dessus. Ils laissent passer toutes les fréquences < *f1*, et rejettent toutes les fréquences > *f2*. La région entre *f1* et *f2* est appelée la *bande de transition*, et contient une atténuation graduelle des fréquences d'entrée. Bien que vous ne souhaitiez laisser passer que des signaux de fréquences < *f1*, les signaux dont les fréquences sont dans la bande de transition peuvent toujours provoquer un repliement. En pratique, la fréquence d'échantillonnage doit donc être plus de deux fois supérieure à la fréquence la plus haute dans la bande de transition. Cela s'avère être plus de deux fois la fréquence d'entrée maximale (*f1*). C'est une des raisons pour laquelle vous constaterez que la fréquence d'échantillonnage est supérieure à deux fois la fréquence d'entrée maximale. Vous verrez dans une prochaine section comment la bande de transition du filtre dépend du type de filtre.

Pourquoi utiliser des décibels ?

Sur certains instruments, vous pouvez afficher l'amplitude dans une échelle linéaire ou en décibel (dB). L'échelle linéaire montre les amplitudes réelles, alors que l'échelle décibel est une transformation de l'échelle linéaire dans une échelle logarithmique. Nous allons maintenant voir pourquoi cette transformation est nécessaire.

Supposez que vous souhaitiez représenter un signal avec des amplitudes très larges et aussi très petites. Supposons que vous ayez un écran d'une hauteur de 10 cm et que vous utilisiez la hauteur entière de l'écran pour l'amplitude la plus grande. Ainsi, si l'amplitude la plus grande dans le signal est 100 V, 1 cm de la hauteur de l'écran correspond à 10 V. Si la plus

petite amplitude du signal est 0,1 V, ceci correspond à une hauteur de seulement 0,1 mm. Cela sera à peine visible à l'écran.

Pour voir toutes les amplitudes, de la plus grande à la plus petite, vous devez changer l'échelle d'amplitude. Alexander Graham Bell a inventé une unité logarithmique, le Bell, qui compresse les grandes amplitudes et grossit les petites amplitudes. Cependant, le Bell étant une unité trop grande, c'est le décibel (1/10ème de Bell) qui est couramment utilisé. Le décibel (dB) est défini comme

$$\text{un dB} = 10 \log_{10} (\text{rapport de puissance}) = 20 \log_{10} (\text{rapport de tension})$$

Le tableau suivant montre la relation entre le décibel et les rapports de puissance et de tension.

dB	Rapport de puissance	Rapport de tension
+40	10000	100
+20	100	10
+6	4	2
+3	2	1.4
0	1	1
–3	1/2	1/1.4
–6	1/4	1/2
–20	1/100	1/10
–40	1/10000	1/100

Vous voyez ainsi que l'échelle dB est utile pour la compression d'une large gamme d'amplitudes dans un petit ensemble de nombres. L'échelle décibel est souvent utilisée pour les mesures de sons et de vibrations, et pour l'affichage d'informations du domaine fréquentiel.

12

Génération de signaux

Ce chapitre explique comment produire des signaux utilisant une fréquence normalisée et comment construire un générateur de fonction simulé. Pour des exemples concernant l'utilisation de VIs de génération de signaux, consultez les exemples qui se trouvent dans la bibliothèque `examples\analysis\sigxmpl.llb`.

Vous apprendrez à utiliser les VIs de la bibliothèque d'analyse pour générer différents types de signaux. Les applications pour la génération de signaux sont par exemple :

- La simulation de signaux pour tester votre algorithme lorsque des signaux réels ne sont pas disponibles (par exemple, lorsque vous n'avez pas de carte DAQ pour obtenir des signaux réels, ou lorsque l'accès aux signaux réels n'est pas possible).
- La génération de signaux à appliquer à un convertisseur N/A.

Fréquence normalisée

Dans l'univers analogique, la fréquence d'un signal est mesurée en Hz ou en périodes (cycles) par seconde. Toutefois, le système numérique utilise souvent une fréquence numérique, qui est le rapport entre la fréquence analogique et la fréquence d'échantillonnage :

> fréquence numérique = fréquence analogique / fréquence d'échantillonnage

La fréquence numérique est appelée la fréquence *normalisée*. Ses unités sont les périodes (cycles)/échantillon.

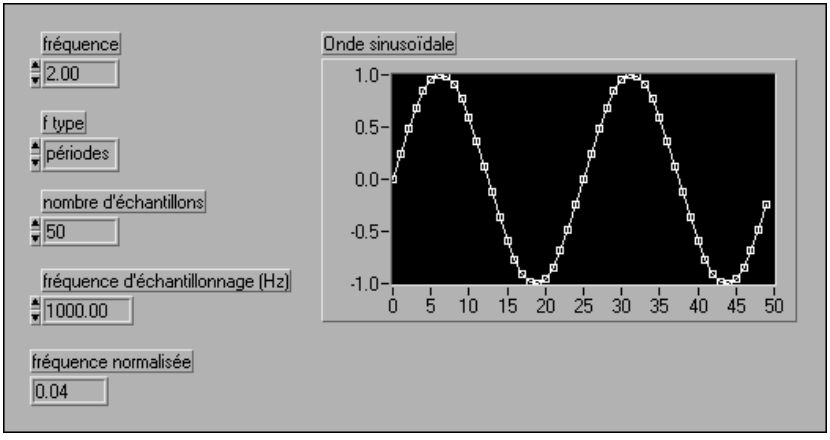
Certains des VIs de génération de signaux utilisent une commande de fréquence d'entrée, *f*, qui est supposée utiliser les unités de *fréquence normalisée* en *période (cycle) par échantillon*. Cette fréquence s'inscrit entre 0,0 et 1,0, ce qui correspond à une gamme de fréquence réelle de 0 à la fréquence d'échantillonnage *fs*. Cette fréquence est aussi autour de 1,0, pour qu'une fréquence normalisée de 1,1 soit égale à 0,1. Par exemple, un signal échantillonné à la fréquence de Nyquist (*fs/2*) est échantillonné deux fois par période (c'est-à-dire deux échantillons/période). Cela correspond à

une fréquence normalisée de 1/2 périodes/échantillon = 0,5 période/échantillon. L'inverse de la fréquence normalisée, *1/f*, vous donne le nombre de fois que le signal est échantillonné au cours d'une période.

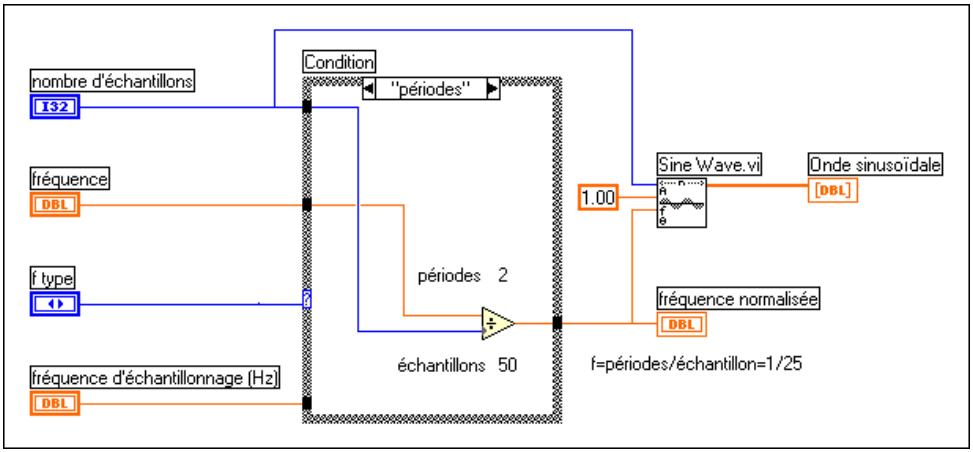
Lorsque vous utilisez un VI qui requiert la fréquence normalisée comme entrée, vous devez convertir vos unités de fréquence en unités normalisées (périodes/échantillons). Vous devez utiliser ces unités normalisées avec les VIs suivants.

- Signal sinusoïdal (Sine Wave)
- Signal carré (Square Wave)
- Signal en dents de scie (Sawtooth Wave)
- Signal triangulaire (Triangle Wave)
- Signal arbitraire (Arbitrary Wave)
- Motif de modulation de fréquence (Chirp Pattern)

Si vous avez l'habitude de travailler avec des unités de fréquence de cycles, vous pouvez convertir les cycles en cycles/échantillon en divisant la période par le nombre d'échantillons générés. L'illustration suivante présente le VI **Signal sinusoïdal** (Sine Wave), qui est utilisé pour générer deux cycles d'un signal sinusoïdal.



L'illustration suivante présente le diagramme pour la conversion de cycles en périodes/échantillon.

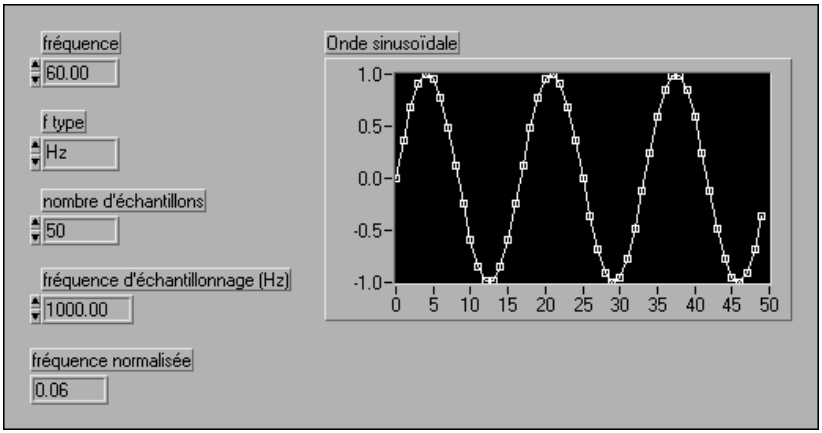


Vous avez seulement besoin de diviser la fréquence (en cycles) par le nombre d'échantillons. Dans l'exemple ci-dessus, la fréquence de 2 cycles est divisée par 50 échantillons, donnant une fréquence normalisée de f = 1/25 cycles/échantillon. Ceci signifie qu'il faut 25 (l'inverse de f) échantillons pour générer un cycle du signal sinusoïdal.

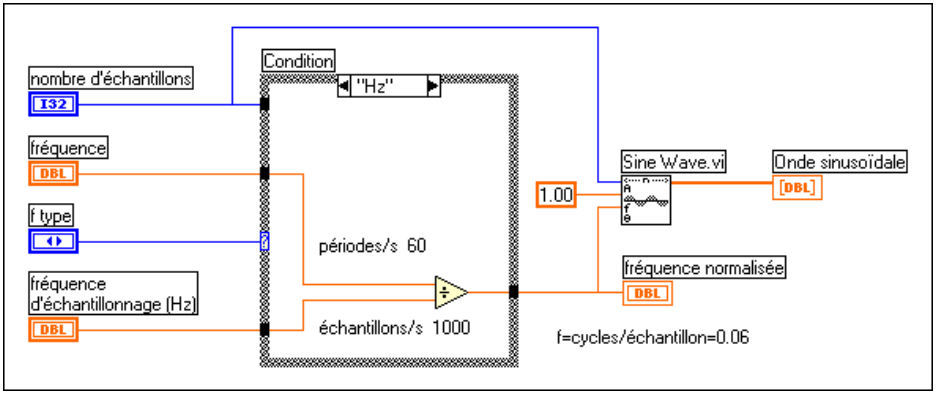
Cependant, vous pouvez avoir besoin d'utiliser des unités de fréquence en Hz (cycles/s). Si vous avez besoin de convertir des Hz (ou cycles/s) en cycles/échantillon, divisez votre fréquence en cycles/s par la fréquence d'échantillonnage donnée en échantillons/s.

$$\frac{\text{cycles/s}}{\text{échantillons/s}} = \frac{\text{cycles}}{\text{échantillon}}$$

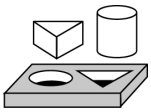
L'illustration suivante présente le VI **Signal sinusoïdal** (Sine Wave) utilisé pour générer un signal sinusoïdal de 60 Hz.



L'illustration suivante présente un diagramme permettant de générer un signal sinusoïdal en Hertz. Vous divisez la fréquence de 60 Hz par la fréquence d'échantillonnage 1000 Hz pour obtenir la fréquence *normalisée* de f = 0,06 cycle/échantillon. Il faut donc presque 17 (1/0,06) échantillons pour générer un cycle du signal sinusoïdal.



Les VIs de génération de signaux créent de nombreux signaux classiques nécessaires à l'analyse et à la simulation de réseau. Vous pouvez aussi utiliser les VIs de génération de signaux en conjonction avec le matériel de National Instruments pour générer des signaux de sorties analogiques.

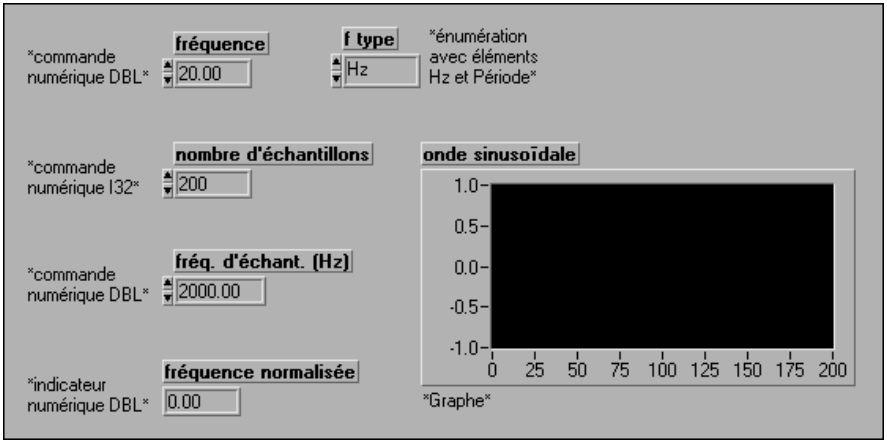


Exercice 12-1. Pour en savoir plus sur la fréquence normalisée

Votre objectif est d'en savoir plus sur la fréquence normalisée en réglant la fréquence, la fréquence d'échantillonnage et le nombre d'échantillons, puis en observant les effets de ces changements sur un signal sinusoïdal.

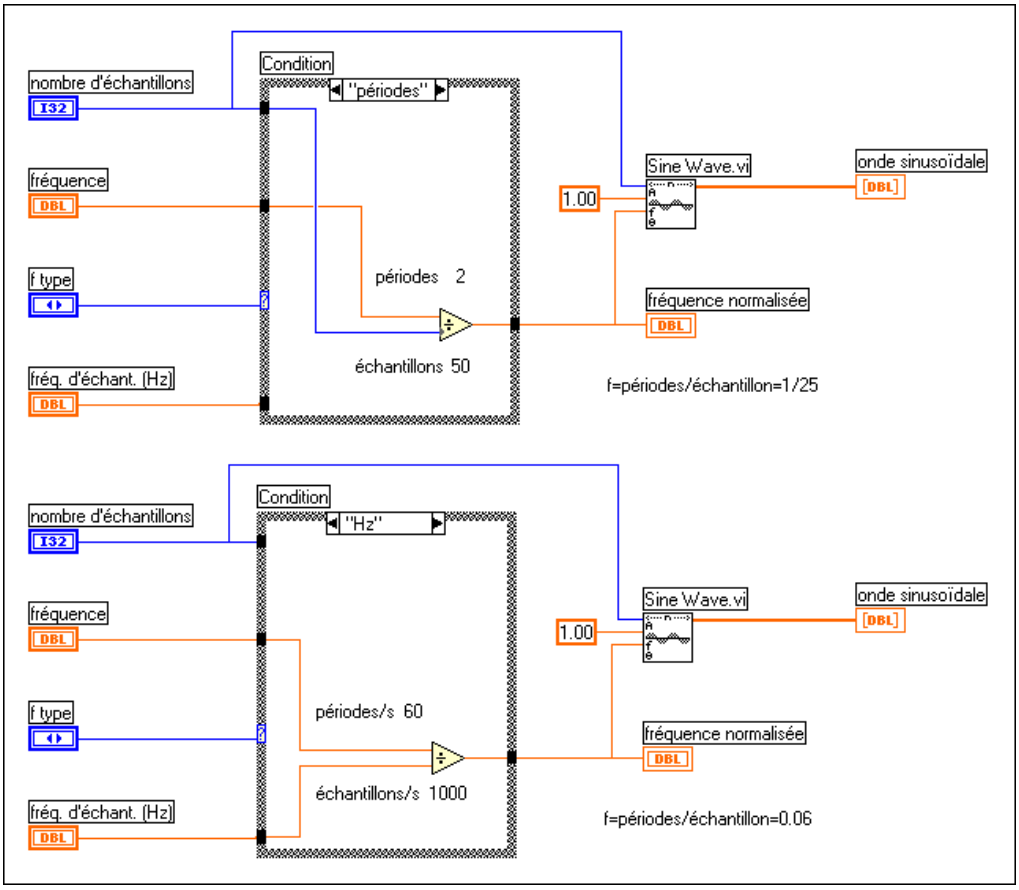
Face-avant

1. Ouvrez une nouvelle face-avant et créez les objets comme indiqué dans l'illustration suivante.



Diagramme

2. Construisez le diagramme présenté dans l'illustration suivante.

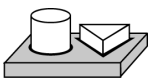


VI Signal sinusoïdal (palette **Analyse»Génération de signal**).

3. Enregistrez le VI sous `Fréquence normalisée.vi` dans le répertoire `LabVIEW\Activity`.
4. Sélectionnez une fréquence de 2 cycles (**fréquence** = 2 et **ftype** = cycles) et un **nombre d'échantillons** = 100. Exécutez le VI. Remarquez que le tracé représente 2 cycles.
5. Augmentez le **nombre d'échantillons** jusqu'à 150, 200 et 250. Combien de cycles voyez-vous ?

6. Restez maintenant avec le **nombre d'échantillons** = 100. Augmentez le nombre de cycles à 3, 4 et 5. Combien de cycles voyez-vous ?

   Vous pouvez donc constater que lorsque vous choisissez la fréquence en termes de cycles, vous voyez ce nombre de cycles du signal d'entrée sur le tracé. Remarquez que la fréquence d'échantillonnage n'a pas de signification dans ce cas.
7. Changez **ftype** en Hz et mettez la **fréquence d'échantillonnage (Hz)** sur 1000.
8. Conservez le **nombre d'échantillons** sur 100 et faites passer la **fréquence** sur 10, 20, 30 et 40. Combien de cycles du signal voyez-vous sur le tracé pour chaque cas ? Expliquez vos observations.
9. Répétez l'étape ci-dessus en conservant la **fréquence** sur 10 et mettez le **nombre d'échantillons** sur 100, 200, 300 et 400. Combien de cycles du signal voyez-vous sur le tracé pour chaque cas ? Expliquez vos observations.
10. Conservez la **fréquence** sur 20 et le **nombre d'échantillons** sur 200. Changez la **fréquence d'échantillonnage (Hz)** sur 500, 1000 et 2000. Assurez-vous de comprendre les résultats.



Fin de l'exercice 12-1.

VIs de motif et de signaux

Vous remarquerez que les noms de la plupart des VIs de génération de signaux comportent le mot *signal* ou *motif*. Il existe une différence de base dans le fonctionnement de ces deux types de VIs : un de ces VIs garde la trace de la phase du signal qu'il génère chaque fois qu'il est appelé, l'autre pas.

Contrôle de la phase

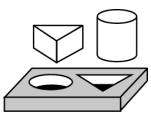
Les VIs *de signaux* ont une commande d'*entrée de phase* dans laquelle vous pouvez spécifier la phase initiale (en degrés) du premier échantillon du signal généré. Ils ont aussi un indicateur de *sortie de phase* qui spécifie la phase du prochain échantillon du signal généré. En outre, une commande *mise à zéro de la phase* décide si la phase du premier échantillon généré, lorsque le VI *signal* est appelé, est la phase spécifiée dans la commande d'*entrée de phase*, ou si c'est la phase disponible dans la commande *sortie de phase* quand le VI s'exécute pour la dernière fois. Une valeur Vrai (TRUE) de l'entrée *mise à zéro de la phase* définit la phase initiale sur

entrée de phase, alors qu'une valeur Faux (FALSE) la définit sur la valeur de *sortie de phase* quand le VI s'exécute pour la dernière fois.

Les VIs *de signaux* sont tous réentrants (ils peuvent garder la trace d'une phase de manière interne) et acceptent la fréquence dans des unités normalisées (cycles/échantillon). Le seul VI de *motif* qui utilise actuellement des unités normalisées est le VI **Motif de modulation de fréquence**. Définir le booléen *mise à zéro de la phase* sur Faux (FALSE) permet une simulation d'échantillonnage en continu.

☞ **Remarque** ***Les VIs de signaux sont réentrants et acceptent l'entrée de fréquence en termes d'unités normalisées.***

Dans le prochain exercice, vous allez générer un signal sinusoïdal à l'aide du VI **Signal sinusoïdal** (**Sine Wave**) et du VI **Motif sinus (Sine Pattern)**. Vous verrez de quelle façon vous avez davantage de contrôle sur la phase initiale avec le VI **Signal sinusoïdal** qu'avec le VI **Motif sinus**.

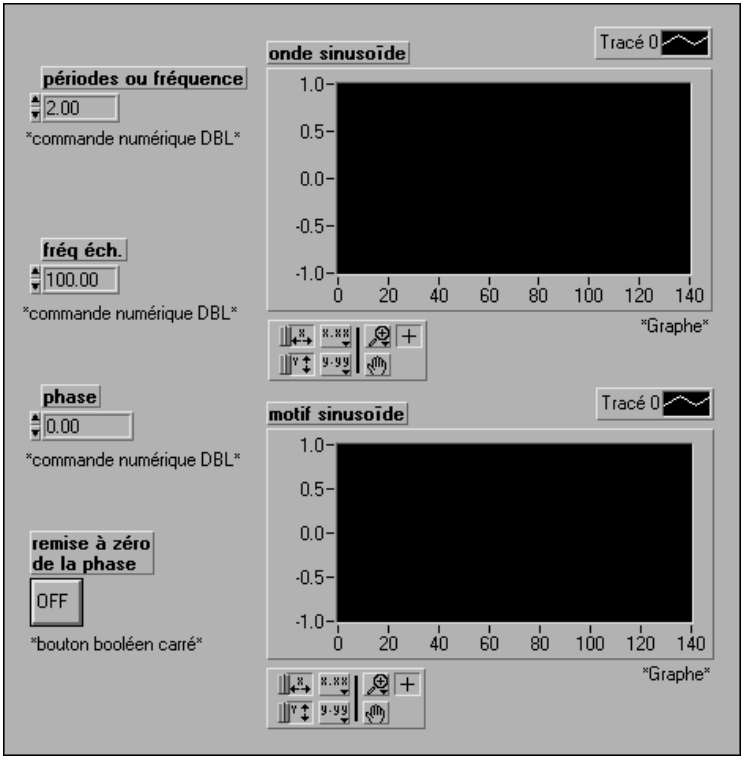


Exercice 12-2. Utiliser les VIs Signal sinusoïdal et Motif sinus

Votre objectif est de générer un signal sinusoïdal à l'aide du VI* Signal sinusoïdal *(Sine Wave) et du VI Motif sinus (Sine Pattern), et de comprendre les différences.

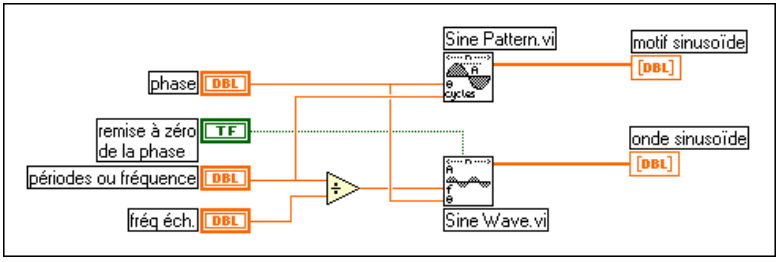
Face-avant

1. Ouvrez une nouvelle face-avant et créez les objets comme indiqué dans l'illustration suivante.



Diagramme

2. Construisez le diagramme présenté dans l’illustration suivante.



VI Motif sinus (**Sine Pattern**) (palette **Analyse»Génération de signal**).

VI **Signal sinusoïdal** (**Sine Wave**) (palette **Analyse»Génération de signal**).

3. Enregistrez le VI sous `Signal et motif.vi` dans le répertoire `LabVIEW\Activity`.
4. Affectez aux commandes les valeurs suivantes :

   cycles/fréq. : `2,00`

   fréq. d'échantillonnage : `100`

   entrée de phase : `0,00`

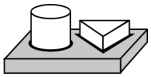
   mise à zéro de la phase : `ARRET`

   Exécutez le VI plusieurs fois.

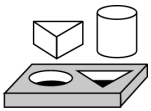
   Observez que le tracé du **signal sinusoïdal** change chaque fois que vous exécutez le VI. Comme la **mise à zéro de la phase** est définie sur ARRET, la phase du **signal sinusoïdal** change lors de chaque appel du VI et est égale à la valeur de la **sortie de phase de l'appel précédent**. Cependant, le tracé du motif **sinusoïdal** reste toujours le même, indiquant 2 cycles du signal sinusoïdal. La phase initiale du tracé de motif **sinusoïdal** est égale à la valeur définie dans la commande **entrée de phase**.

Remarque ***"Entrée de* phase*" et "sortie de* phase*" sont spécifiés en degrés.***

5. Changez l'**entrée de phase** sur 90 et exécutez le VI plusieurs fois. Comme précédemment, le tracé du **signal sinusoïdal** change chaque fois que vous exécutez le VI. Cependant, le tracé du motif **sinusoïdal** ne change pas, mais la phase initiale du motif sinusoïdal est de 90 degrés – la même que celle spécifiée dans la commande **entrée de phase**.
6. Avec l'**entrée de phase** toujours sur 90, mettez **mise à zéro de la phase** sur MARCHE et exécutez le VI plusieurs fois. Les **signaux sinusoïdaux** représentés dans les tracés du **signal sinusoïdal** et du motif **sinusoïdal** commencent à 90 degrés, mais ne changent pas lors des appels successifs du VI.
7. En gardant la **mise à zéro de la phase** sur MARCHE, exécutez le VI plusieurs fois pour chacune des valeurs suivantes d'**entrée de phase** : 45, 180, 270 et 360. Remarquez la phase initiale du signal généré chaque fois que le VI s'exécute.



Fin de l'exercice 12-2.



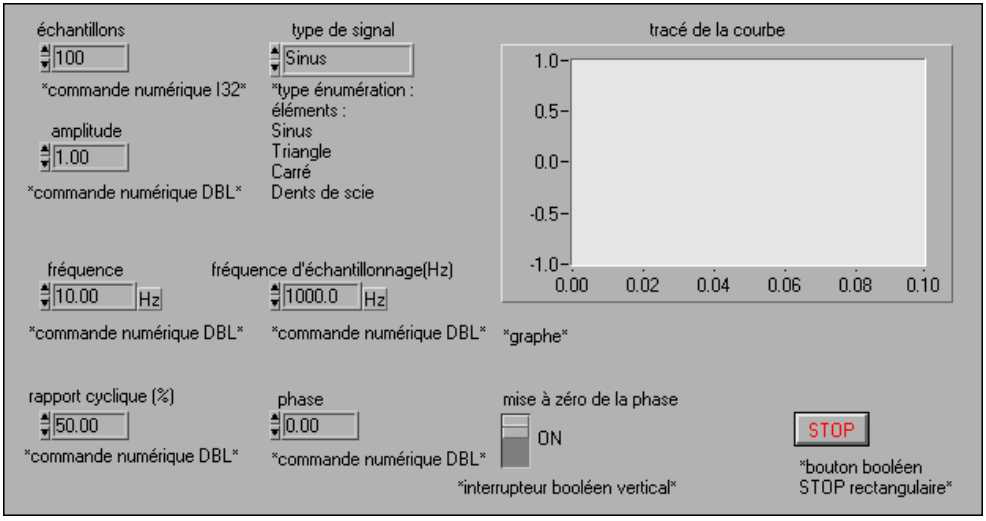
Exercice 12-3. Construire un générateur de fonction

Votre objectif est de construire un générateur de fonction simple qui peut générer les signaux suivants :

- ***Signal sinusoïdal***
- ***Signal carré***
- ***Signal triangulaire***
- ***Signal en dents de scie***

Face-avant

1. Ouvrez une nouvelle face-avant et créez les objets comme indiqué dans l'illustration suivante.



La commande **source du signal** sélectionne le type de signal que vous voulez générer.

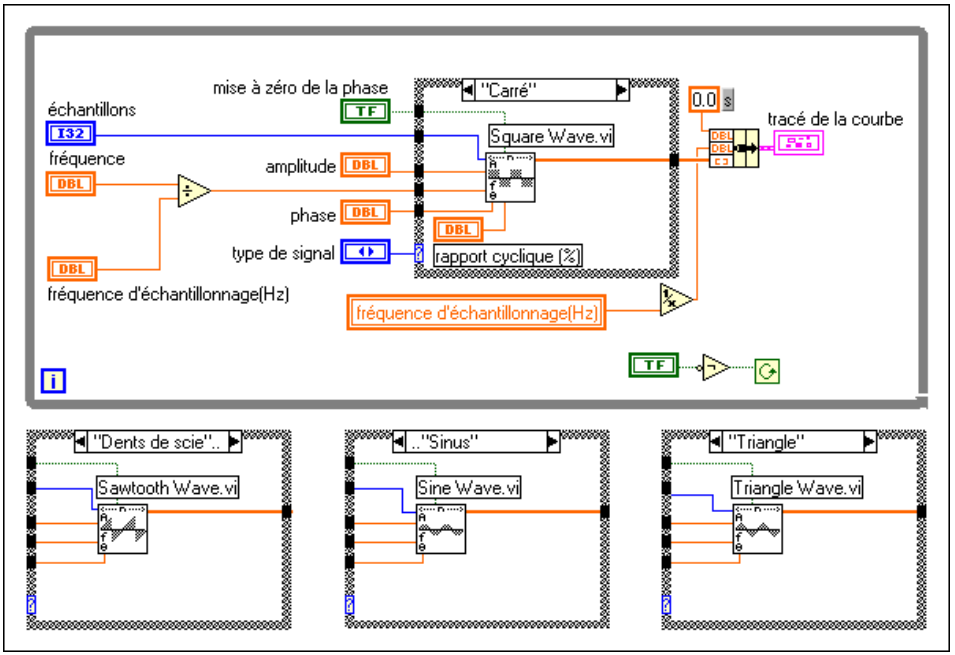
La commande **facteur d'utilisation carré** n'est utilisée que pour définir le facteur d'utilisation du signal carré.

La commande **échantillons** détermine le nombre d'échantillons dans le tracé.

Remarquez qu'il s'agit uniquement de VIs de *signaux*, ils ont donc besoin que l'entrée de fréquence soit la fréquence normalisée. Vous devez ainsi diviser la **fréquence** par la **fréquence d'échantillonnage** et le résultat donne la fréquence normalisée câblée sur l'entrée *f* des VIs.

Diagramme

2. Construisez le diagramme présenté dans l'illustration suivante.



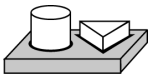
Le **VI Signal sinusoïdal (Sine Wave)** (palette **Analyse»Génération de signal**) génère un **signal sinusoïdal** de fréquence normalisée *f*.

Le **VI Signal triangulaire (Triangle Wave)** (palette **Analyse» Génération de signal**) génère un signal triangulaire de fréquence normalisée *f*.

Le **VI Signal carré (Square Wave)** (palette **Analyse»Génération de signal**) génère un signal carré de fréquence normalisée *f* avec le facteur d'utilisation spécifié.

Le **VI Signal en dents de scie (Sawtooth Wave)** (palette **Analyse» Génération de signal**) génère un signal en dents de scie de fréquence normalisée *f*.

3. Enregistrez le VI sous `Générateur de fonction.vi` dans le répertoire `LabVIEW\Activity`.
4. Sélectionnez une **fréquence d'échantillonnage** de 1000 Hz, **amplitude** = 1, **échantillons** = 100, **fréquence** = 10, **mise à zéro de la phase** = MARCHE, et **source du signal** = **signal sinusoïdal**. Puisque **fréquence d'échantillonnage** = 1000 et **fréquence** = 10 Hz, chaque ensemble de 100 échantillons correspond à un cycle.
5. Exécutez le VI et observez le tracé obtenu.
6. Changez **échantillons** sur 200, 300 et 400. Combien de cycles du signal voyez-vous ? Expliquez pourquoi.
7. Avec **échantillons** égal à 100, changez **mise à zéro de la phase** sur ARRET. Voyez-vous une différence dans le tracé ?
8. Changez **fréquence** sur 10,01 Hz. Que se passe-t-il ? Pourquoi ?
9. Changez **mise à zéro de la phase** sur MARCHE. Que se passe-t-il maintenant ? Expliquez pourquoi.
10. Répétez les étapes 4 à 9 pour les différents signaux sélectionnés dans la commande **source du signal**.



Fin de l'exercice 12-3.

13

Traitement des signaux numériques

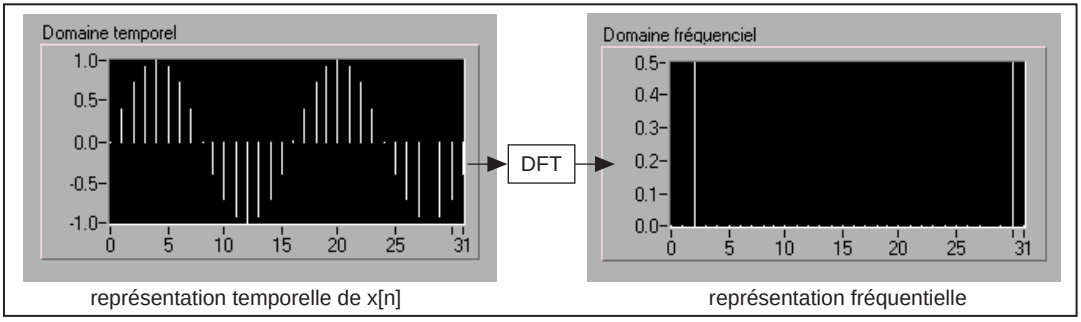
Ce chapitre décrit les fondements de la transformée rapide de Fourier (FFT) et de la transformée discrète de Fourier (DFT), ainsi que la façon de les utiliser dans l'analyse de spectre. Pour des exemples concernant l'utilisation des VIs de traitement de signaux numériques, consultez les exemples qui se trouvent dans la bibliothèque `examples\analysis\dspxmpl.llb`.

La transformée rapide de Fourier (FFT)

Les échantillons d'un signal obtenu avec une carte DAQ constituent la représentation du *domaine temporel* du signal. Cette représentation donne les amplitudes du signal aux instants de *temps* au moment où elles ont été échantillonnées. Cependant, dans de nombreux cas, vous désirez connaître la représentation fréquentielle d'un signal plutôt que les amplitudes des échantillons individuels. La représentation d'un signal en termes de ses composantes de fréquence individuelle est appelée la représentation du *domaine fréquentiel* du signal. La représentation du domaine fréquentiel peut vous donner plus d'informations sur le signal et le système qui l'a généré.

L'algorithme utilisé pour transformer des échantillons de données du domaine temporel dans le domaine fréquentiel est appelé la *transformée discrète de Fourier* ou DFT. La DFT établit la relation entre les échantillons d'un signal dans le domaine temporel et leur représentation dans le domaine fréquentiel. La DFT est largement utilisée dans l'analyse

spectrale, la mécanique appliquée, l'acoustique, l'imagerie médicale, l'analyse numérique, l'instrumentation et les télécommunications.



représentation temporelle de x[n] représentation fréquentielle

Supposons que vous ayez obtenu *N* échantillons d'un signal à partir d'une carte DAQ. Si vous appliquez la DFT à *N* échantillons de cette représentation temporelle du signal, le résultat est aussi une longueur de *N* échantillons, mais les informations obtenues concernent la représentation du domaine fréquentiel. La relation entre les *N* échantillons dans le domaine temporel et les *N* échantillons dans le domaine fréquentiel est expliquée ci-dessous.

Si le signal est échantillonné à une fréquence d'échantillonnage de f_s Hz, l'intervalle de temps entre les échantillons (c'est-à-dire, l'intervalle d'échantillonnage) est Δt, tel que :

$$\Delta t = \frac{1}{f_s}$$

Les échantillons sont notés x[i], $0 \le i \le N-1$ (vous avez un total de *N* échantillons). Lorsque la transformée discrète de Fourier, calculée par :

$$X_k = \sum_{i=0}^{N-1} x_i e^{-j2\pi ik/(N)} \qquad (13\text{-}1)$$

pour :

$$k = 0, 1, 2, \ldots, N-1$$

est appliquée à ces *N* échantillons, le résultat (*X*[k], 0 ð k ð N–1) est la représentation du domaine fréquentiel de x[i]. Remarquez que le domaine temporel *x* et le domaine fréquentiel *X* ont tous les deux un total de

N échantillons. Comme pour l'*espacement temporel* de Δt entre les échantillons de *x* dans le domaine temporel, vous avez un espacement de fréquence égal à :

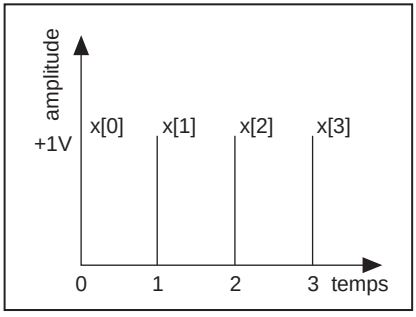
$$\Delta f = \frac{f_s}{N} = \frac{1}{N\Delta t}$$

entre les composantes de *X* dans le domaine fréquentiel. Δf est aussi appelé la *résolution de fréquence*. Pour augmenter la résolution de fréquence (diminuer Δf), vous devez soit augmenter le nombre d'échantillons *N* (avec *fs* constant) soit diminuer la fréquence d'échantillonnage *fs* (avec *N* constant).

Dans l'exemple suivant, vous utiliserez l'équation 13-1 pour calculer la DFT pour un signal D.C.

Exemple de calcul DFT

Dans la section suivante, vous verrez les fréquences exactes auxquelles correspondent les *N* échantillons de la DFT. Pour l'instant, supposons que *X*[0] corresponde à D.C., ou la valeur moyenne, du signal. Pour voir le résultat du calcul de la DFT d'un signal avec l'équation 13-1, considérez un signal D.C. ayant une amplitude constante de +1 V. Quatre échantillons de ce signal sont pris, comme indiqué dans la figure ci-dessous.



Chacun des échantillons a une valeur +1, selon la séquence temporelle :

x[0] = x[1] = x[3] = x[4] = 1

En utilisant l'équation 13-1 pour calculer la DFT de cette séquence et avec l'identité de Euler :

$$\exp(-i\theta) = \cos(\theta) - j\sin(\theta)$$

vous obtenez :

$$X[0] = \sum_{i=0}^{N-1} x_i e^{-j2\pi i0/(N))} = x[0] + x[1] + x[2] + x[3] = 4$$

$$X[1] = x[0] + x[1]\left(\cos\left(\frac{\pi}{2}\right) - j\sin\left(\frac{\pi}{2}\right)\right) + x[2](\cos(\pi) - j\sin(\pi)) + x[3]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) = (1 - j - 1 + j) = 0$$

$$X[2] = x[0] + x[1](\cos(\pi) - j\sin(\pi)) + x[2](\cos(2\pi) - j\sin(2\pi)) + x[3](\cos(3\pi) - j\sin(3\pi)) = (1 - 1 + 1 - 1) = 0$$

$$X[3] = x[0] + x[1]\left(\cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right)\right) + x[2](\cos(3\pi) - j\sin(3\pi)) + x[3]\left(\cos\left(\frac{9\pi}{2}\right) - j\sin\left(\frac{9\pi}{2}\right)\right) = (1 - j - 1 - j) = 0$$

Donc, excepté pour la composante DC, *X*[0], toutes les autres valeurs sont nulles, comme prévu. Cependant, la valeur calculée de *X*[0] dépend de la valeur de *N* (le nombre d'échantillons). Parce que vous avez *N* = 4, X[0] = 4. Si *N* = 10, vous aurez *X*[0] = 10. Cette dépendance de *X*[ ] par rapport à *N* se produit également pour les autres composantes de fréquence. Ainsi, vous divisez généralement la sortie DFT par *N*, de façon à obtenir l'amplitude correcte de la composante de fréquence.

Informations sur la phase et l'amplitude

Vous avez vu que *N* échantillons du signal d'entrée produisaient *N* échantillons de la DFT. Cela signifie que le nombre d'échantillons est le même dans les représentations temporelles et fréquentielles. Dans l'équation 13-1, que le signal d'entrée *x*[i] soit réel ou complexe, *X*[k] est toujours complexe (bien que la partie imaginaire puisse être nulle). Donc, parce que la DFT est complexe, elle contient deux informations : l'amplitude et la phase. Pour des signaux réels (*x*[i] réel), tels que ceux

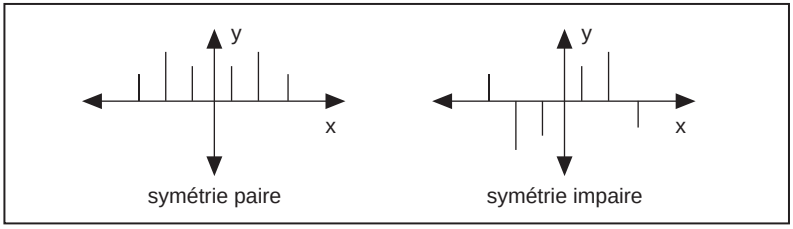
obtenus à partir de la sortie d'une voie de carte DAQ, il apparaît que la DFT est symétrique avec les propriétés suivantes :

$$| X[k] | = | X[N-k] |$$

et

$$\text{phase } (X[k]) = -\text{ phase } (X[N-k])$$

Pour décrire cette symétrie, on dit que l'amplitude de X[k] est *symétrique paire*, et que la phase (X[k]) est *symétrique impaire*. Un signal symétrique pair est un signal symétrique par rapport à l'axe des Y, alors qu'un signal symétrique impair est un signal symétrique par rapport à l'origine. Ceci est représenté dans les figures suivantes.



Cette symétrie permet la répétition des informations contenues dans les *N* échantillons de la DFT. A cause de cette répétition d'informations, seulement la moitié des échantillons de la DFT a réellement besoin d'être calculée ou affichée, alors que l'autre moitié peut être obtenue par cette répétition.

Remarque ***Si le signal d'entrée est complexe, la DFT n'est pas symétrique et vous ne pouvez pas utiliser cette astuce.***

Espacement de fréquence entre les échantillons DFT/FFT

Si l'intervalle d'échantillonnage est Δt secondes, et si le premier échantillon ($k = 0$) est pris à t = 0 seconde, le k^{ème} ($k > 0$, k entier) échantillon de données est pris à t = $k\Delta t$ secondes. De la même manière, si la résolution de fréquence est Δf *Hz*($\Delta f = f_s / N$) le k^{ème} échantillon de la DFT se produit à une fréquence de $k\Delta f$ Hz. (En réalité, comme vous le verrez bientôt, ceci n'est valide que pour la première moitié des composantes de la fréquence. L'autre moitié représente des composantes négatives de la fréquence.) Selon que le nombre d'échantillons, *N*, est pair ou impair, vous pouvez avoir une interprétation différente de la fréquence correspondant au $k^{ème}$ échantillon de la DFT.

Par exemple, supposons que *n* soit pair et que $p = N/2$. Le tableau suivant montre la fréquence à laquelle correspond chaque élément de format de la séquence de sortie complexe *X*.

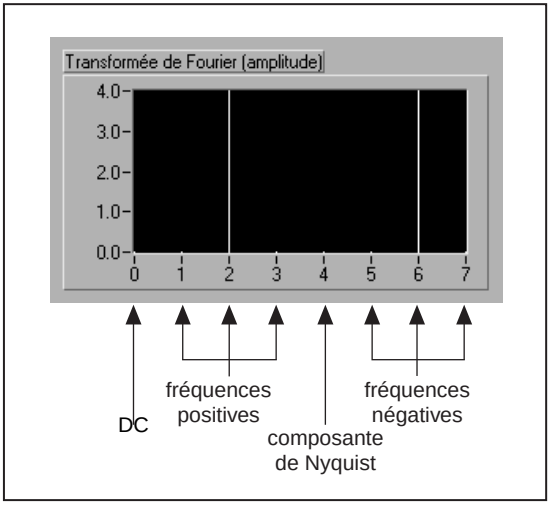
Remarquez que le $p^{ème}$ élément, X[p], correspond à la fréquence Nyquist. Les entrées négatives dans la deuxième colonne au-delà de la fréquence Nyquist représentent des fréquences *négative*s.

Par exemple, si $N = 8$, $p = N/2 = 4$, alors :

X[0]	DC
X[1]	Δf
X[2]	$2\Delta f$
X[3]	$3\Delta f$
X[4]	$4\Delta f$ (fréquence de Nyquist)
X[5]	$-3\Delta f$
X[6]	$-2\Delta f$
X[7]	$-\Delta f$

Ici, X[1] et X[7] ont la même amplitude, X[2] et X[6} ont la même amplitude, et X[3] et X[5] ont la même amplitude. La différence est que X[1], X[2] et X[3] correspondent à des composantes de fréquence positives, alors que X[5], X[6] et X[7] correspondent à des composantes de fréquence négatives. Remarquez que X[4] correspond à la fréquence de Nyquist.

L'illustration suivante représente cette séquence complexe pour $N = 8$.



Une telle représentation, où vous voyez à la fois les fréquences positives et négatives, est appelée la transformée *bilatérale*.

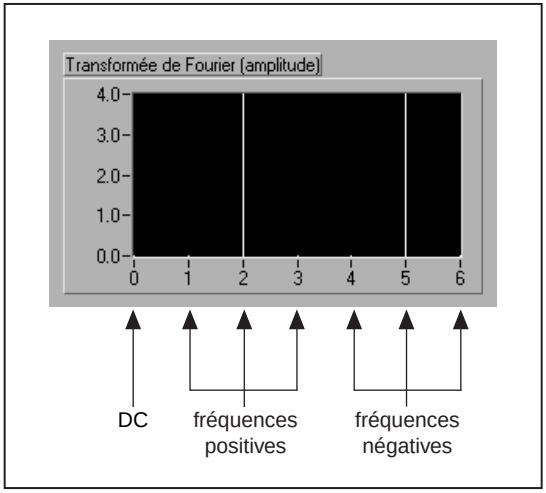
Remarquez que lorsque *N* est impair, il n'y a pas de composante à la fréquence de Nyquist.

Par exemple, si N= 7, p = (N–1)/2 = (7–1)/2 = 3, vous avez :

X[0]	DC
X[1]	Δf
X[2]	$2\Delta f$
X[3]	$3\Delta f$
X[4]	$-3\Delta f$
X[5]	$-2\Delta f$
X[6]	$-\Delta f$

A présent X[1] et X[6] ont la même amplitude, X[2] et X[5] ont la même amplitude, et X[3] et X[4] ont la même amplitude. Cependant, X[1], X[2] et X[3] correspondent à des composantes de fréquence positives, alors que X[4], X[5] et X[6] correspondent à des composantes de fréquence négatives. Comme *N* est impair, il n'y a pas de composante à la fréquence de Nyquist.

L'illustration suivante représente le tableau précédent pour $N = 7$.



Il y a aussi une transformée bilatérale, parce que vous avez à la fois des fréquences positives et négatives.

Transformées rapides de Fourier

L'implémentation directe de la DFT sur N échantillons de données nécessite environ N^2 opérations complexes et demande beaucoup de temps. Cependant, lorsque la taille de la séquence est une puissance de 2 :

$$N = 2^m \quad \text{pour } m = 1, 2, 3,\ldots$$

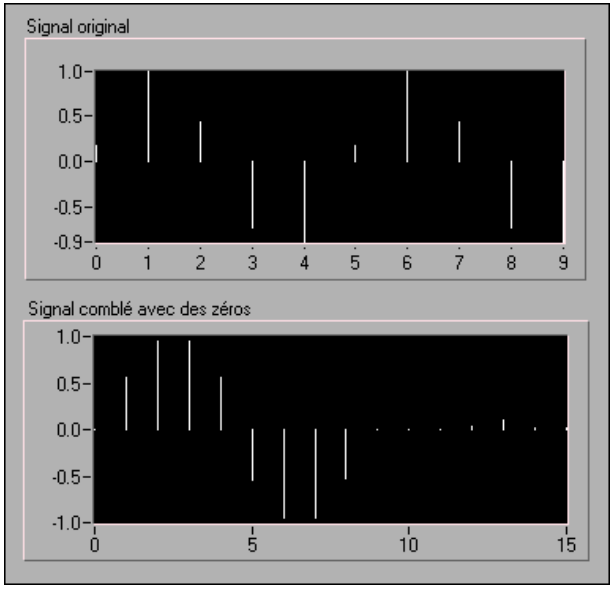
vous pouvez implémenter le calcul de la DFT avec $N \log_2(N)$ opérations environ. Cela rend le calcul de la DFT beaucoup plus rapide, et la littérature DSP appelle ces algorithmes des transformées de Fourier rapides (FFT). La FFT n'est rien d'autre qu'un algorithme rapide pour le calcul de la DFT lorsque le nombre d'échantillons (N) est une puissance de 2.

Parmi les avantages de la FFT, il y a la vitesse et les capacités de mémoire, car le VI peut calculer la FFT "sur place", c'est-à-dire que des buffers de mémoire supplémentaires ne sont pas nécessaires pour calculer la sortie. La taille de la séquence d'entrée, cependant, doit être une puissance de 2. La DFT peut traiter efficacement n'importe quelle séquence de taille, mais la DFT est plus lente que la FFT et utilise davantage de mémoire, parce qu'elle doit affecter des buffers supplémentaires pour le stockage de résultats intermédiaires pendant le traitement.

Combler en utilisant des zéros

Une technique employée pour rendre la taille de séquence d'entrée égale à une puissance de 2 est d'ajouter des zéros à la fin de la séquence pour que le nombre total d'échantillons soit égal à la puissance de 2 la plus proche.

Par exemple, si vous avez 10 échantillons d'un signal, vous pouvez ajouter six zéros pour rendre le nombre total d'échantillons égal à 16 (= 2^4). Ceci est représenté ci-dessous.



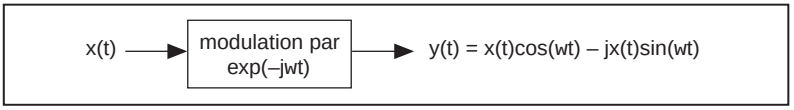
L'addition de zéros à la fin du signal dans le domaine temporel n'affecte pas le spectre du signal. En plus de transformer le nombre total d'échantillons en puissance de deux de manière à calculer plus rapidement en utilisant la FFT, combler en utilisant des zéros aide aussi à augmenter la résolution en fréquence (rappelez-vous que Δf = fs/N) en augmentant le nombre d'échantillons, *N*.

VIs FFT dans la bibliothèque d'analyse

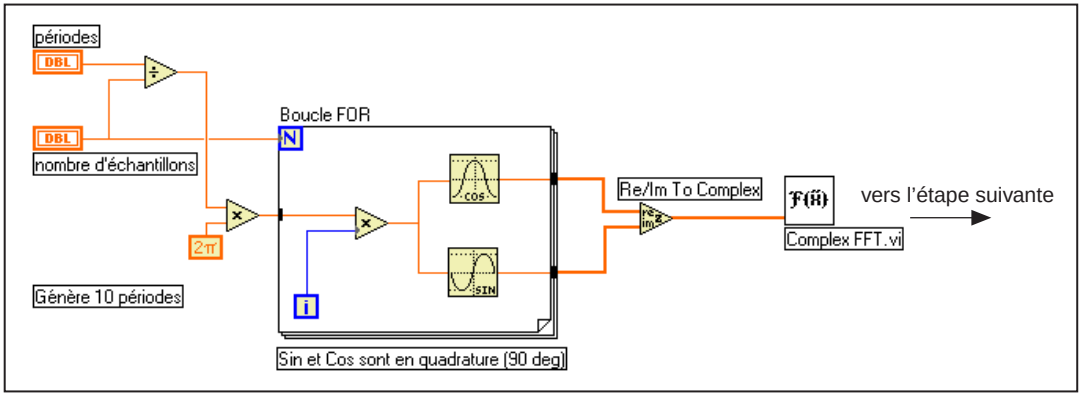
La bibliothèque d'analyse contient deux VIs qui calculent la FFT d'un signal. Il s'agit du VI **FFT réel (Real FFT)** et du VI **FFT complexe (Complex FFT)**.

La différence entre les deux VIs est que le VI **FFT réel** calcule la FFT d'un signal de valeur réelle, alors que le VI **FFT complexe** calcule la FFT d'un signal de valeur complexe. Gardez cependant à l'esprit que les sorties des deux VIs sont complexes.

La plupart des signaux réels ont des valeurs réelles, et vous pouvez donc utiliser le VI **FFT réel** pour la plupart des applications. Bien sûr, vous pouvez aussi utiliser le VI **FFT complexe** en définissant la partie imaginaire du signal nulle. Un exemple d'une application dans laquelle vous pouvez utiliser le VI **FFT complexe** est une application ayant un signal avec des composantes réelles et imaginaires. Un tel type de signal se produit fréquemment dans le domaine des télécommunications, où vous modulez un signal par une exponentielle complexe. Le processus de modulation par une exponentielle complexe donne un signal complexe, comme montré ci-dessous.



Le diagramme ci-dessous montre une version simplifiée de la façon dont vous pouvez générer 10 cycles d'un signal complexe :

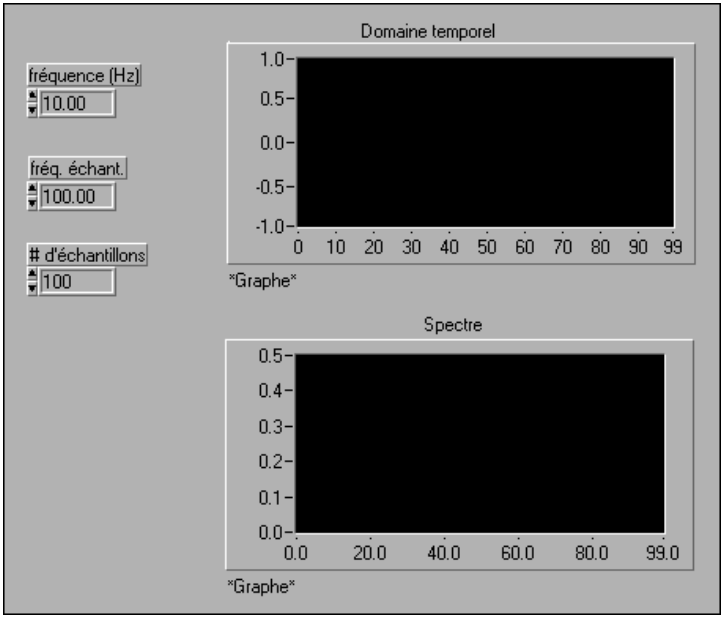


Exercice 13-1. Utilisation du VI FFT réel (Real FFT)

Dans cet exercice, votre objectif est d'afficher la transformée de Fourier unilatérale et bilatérale d'un signal en utilisant le VI FFT réel (Real FFT), et d'observer l'effet du repliement dans le spectre de fréquence.

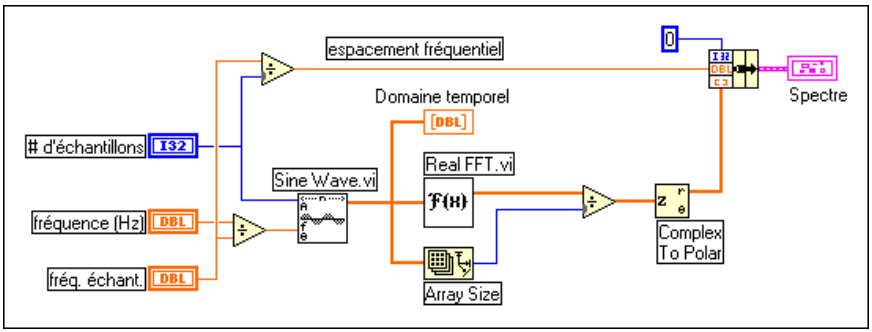
Face-avant

1. Construisez la face-avant présentée dans l'illustration suivante.



Diagramme

2. Construisez le diagramme présenté dans l'illustration suivante.



La fonction **"Taille d'un tableau" (Array Size)** (palette **Fonctions» Tableau**) met à l'échelle la sortie de la FFT par le **nombre d'échantillons** de façon à obtenir l'amplitude des composantes de fréquence correcte.

La fonction **"Signal sinusoïdal"** (**Sine Wave**) (palette **Fonctions» Analyse»Génération de signal**) génère un **signal sinusoïdal** de domaine temporel.

La fonction **"FFT réel"** (**Real FFT**) (palette **Fonctions»Analyse» Traitement des signaux numériques**) calcule la FFT des échantillons de données d'entrée.

La fonction **"Polaire en complexe" (Complex to Polar)** (palette **Fonctions»Numérique»Complexe**) sépare la sortie complexe de la FFT dans ses parties réelle et imaginaire (amplitude et phase). L'unité de la phase est le radiant. Vous n'affichez ici que l'amplitude de la FFT.

L'espacement de fréquence, Δf, est trouvé en divisant la **fréq. d'échantillonnage** par le **nombre d'échantillons**.

3. Enregistrez ce VI sous `FFT bilatérale.vi` dans le répertoire `LabVIEW\Activity`.
4. Sélectionnez **fréquence (Hz)** = 10, **fréq. d'échantillonnage** = 100, et **nombre d'échantillons** = 100. Exécutez le VI.

   Observez les tracés du signal temporel et le spectre de fréquence. **Fréq. d'échantillonnage** = **nombre d'échantillons** = 100, vous échantillonnez donc effectivement pendant 1 seconde. Ainsi, le nombre de cycles du **signal sinusoïdal** que vous voyez dans le signal temporel est égal à la **fréquence(Hz)** que vous sélectionnez. Dans ce

cas, vous observez 10 cycles. (Si vous mettez la **fréquence (Hz)** sur 5, vous observez 5 cycles).

FFT bilatérale

5. Examinez le spectre de fréquence (la transformée de Fourier). Vous remarquez deux pics, un à 10 Hz et l'autre à 90 Hz. Le pic à 90 Hz est en réalité la fréquence négative de 10 Hz. Le tracé que vous voyez est appelé la *FFT bilatérale* parce qu'il montre à la fois les fréquences positive et négatives.
6. Exécutez le VI avec **fréquence (Hz)** = 10 et avec **fréquence (Hz)** = 20. Pour chaque cas, remarquez le décalage dans les deux pics du spectre.

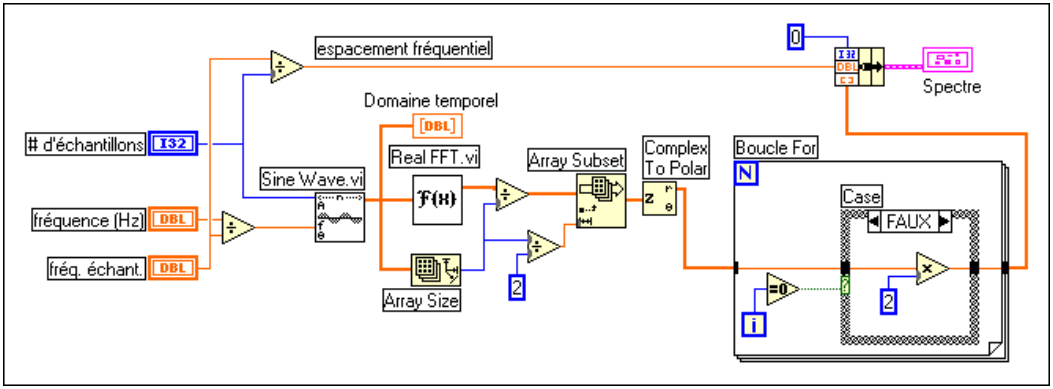
Remarque ***Observez aussi le tracé du domaine temporel pour les fréquences (Hz) = 10 et 20. Laquelle donne la meilleure représentation du signal sinusoïdal ? Pourquoi ?***

7. fs = 100 Hz, vous ne pouvez donc échantillonner précisément que les signaux ayant une fréquence < 50 Hz (fréquence de Nyquist = fs/2). Changez la **fréquence (Hz)** sur 48 Hz. Vous devez voir les pics à ± 48 Hz sur le tracé du spectre.
8. Faites passer maintenant la **fréquence (Hz)** sur 52 Hz. Y a-t-il une différence entre le résultat de l'étape 5 et ce que vous voyez sur les tracés maintenant ? Etant donné que 52 > Nyquist, la fréquence 52 est repliée sur |100 – 52| = 48 Hz.
9. Placez la **fréquence (Hz)** sur 30 Hz et 70 Hz, puis exécutez le VI. Y a-t-il une différence entre les deux cas ? Expliquez pourquoi.

FFT unilatérale

10. Modifiez le diagramme du VI comme montré ci-dessous. Vous avez vu que la FFT possède une répétition des informations car elle contient des informations sur les fréquences positives et négatives. Cette modification représente désormais seulement la moitié des points de la FFT (seulement les composantes de fréquence positive). Cette représentation est appelée la *FFT unilatérale*. Remarquez que vous devez multiplier les composantes de fréquence positive par deux

pour obtenir l'amplitude correcte. La composante D.C., cependant, reste inchangée.



La fonction "Egal à 0 ?" (palette **Fonctions»Comparaison**) teste l'indice de tableau. S'il est égal à zéro, il correspond à la composante D.C. et ne doit pas être multiplié par deux.

11. Exécutez le VI avec les valeurs suivantes : **fréquence (Hz)** = 30, **fréq. d'échantillonnage** = 100, **nombre d'échantillons** = 100.
12. Enregistrez le VI sous `FFT unilatérale.vi` dans le répertoire `LabVIEW\Activity`.
13. Changez la valeur de la **fréquence (Hz)** sur 70 et exécutez le VI. Remarquez-vous une différence entre ce résultat et celui de l'étape 9 ?

Fin de l'exercice 13-1.

Le spectre de puissance

Vous avez vu que la DFT (ou FFT) d'un signal réel est un nombre complexe, ayant une partie réelle et imaginaire. La *puissance* dans chaque composante de fréquence représentée par la DFT/FFT peut être obtenue en mettant au carré l'amplitude de cette composante de fréquence. Ainsi, la puissance de la $k^{ème}$ composante de fréquence (le $k^{ème}$ élément de la DFT/FFT) est donnée par $k^{ème}$par $|X[k]|^2$. Le tracé montrant la puissance dans chacune des composantes de fréquence est appelé le *spectre de puissance*. Parce que la DFT/FFT d'un signal réel est symétrique, la puissance à une fréquence positive de $k\Delta f$ est la même que la puissance à la fréquence négative correspondante de $-k\Delta f$ (composantes DC et Nyquist non incluses). La puissance totale des composantes DC et Nyquist sont, respectivement $|X[0]|^2$ et $\left|X\left[\frac{N}{2}\right]\right|^2$.

Perte d'informations de phase

Comme la puissance est obtenue en mettant au carré l'amplitude de la DFT/FFT, le spectre de puissance est toujours réel. Le problème est que les informations concernant la phase sont perdues. Si vous voulez des informations de phase, vous devez utiliser la DFT/FFT, qui vous donne une sortie complexe.

Vous pouvez utiliser le spectre de puissance dans des applications où les informations de phase ne sont pas nécessaires (par exemple, pour calculer la puissance harmonique d'un signal). Vous pouvez appliquer une entrée sinusoïdale à un système non linéaire et voir la puissance des harmoniques à la sortie du système.

Espacement de fréquence entre échantillons

Vous pouvez utiliser le VI **Spectre de puissance (Power Spectrum)** dans la sous-palette **Analyse»Traitement des signaux numériques** pour calculer le spectre de puissance des échantillons de données du domaine temporel. Comme la DFT/FFT, le nombre d'échantillons de la sortie du VI **Spectre de puissance** est le même que le nombre d'échantillons de données appliqués à l'entrée. L'espacement de fréquence entre les échantillons de sortie est Δf = fs/N.

En résumé

La représentation temporelle (valeurs d'échantillon) d'un signal peut être convertie dans le domaine fréquentiel à l'aide d'un algorithme appelé la transformée discrète de Fourier (DFT). Pour calculer rapidement la DFT, un algorithme appelé la transformée rapide de Fourier (FFT) est utilisé. Vous pouvez utiliser cet algorithme lorsque le nombre d'échantillons de signal est une puissance de deux.

La sortie de la DFT/FFT conventionnelle est dite bilatérale parce qu'elle contient des informations sur les fréquences positives et négatives. Cette sortie peut être convertie dans une DFT/FFT unilatérale en utilisant seulement la moitié des points de sortie DFT/FFT. L'espacement de fréquence entre les échantillons de la DFT/FFT est $\Delta f = fs/N$.

Le spectre de puissance peut être calculé à partir de la DFT/FFT en mettant au carré l'amplitude des composantes de fréquence individuelles. Le VI **Spectre de puissance (Power Spectrum)** de la bibliothèque d'analyse avancée fait cela automatiquement pour vous. Les unités de la sortie de ce VI sont des $Vrms^2$. Cependant, le spectre de puissance ne fournit pas d'information sur la phase.

La DFT, la FFT et le spectre de puissance sont utiles pour mesurer la fréquence de signaux stationnaires ou transitoires. La FFT fournit les fréquences du signal pour la durée totale pendant laquelle le signal est acquis. Pour cette raison, la FFT la plus souvent utilisée pour l'analyse de signal stationnaire (lorsque le signal ne change pas significativement de fréquence pendant la durée de l'acquisition), ou lorsque vous voulez seulement l'énergie moyenne de chaque fréquence. Une grande partie des problèmes de mesure tombe dans cette catégorie. Pour mesurer des informations de fréquence qui changent pendant l'acquisition, vous devez utiliser le toolkit Joint time-frequency analysis (JTFA) ou le toolkit Wavelet and filter banks designer (WFBD).

14

Fenêtres de lissage

Ce chapitre explique comment l'utilisation des fenêtres prévient la perte spectrale et améliore l'analyse de signaux acquis. Pour des exemples d'utilisation des VIs Analyse de fenêtres, voyez les exemples qui se trouvent dans la bibliothèque `examples\analysis\windxmpl.llb`.

Introduction aux fenêtres de lissage

Dans les applications d'échantillonnage de signal pratiques, vous pouvez seulement obtenir un enregistrement fini du signal, même lorsque vous observez avec précaution le théorème de Nyquist et les conditions d'échantillonnage. Malheureusement pour le système temps discret, l'enregistrement d'échantillonnage fini aboutit à un signal tronqué qui a des caractéristiques spectrales différentes du signal temps continu original. Ces discontinuités produisent la perte d'informations spectrales, donnant lieu à un spectre temps discret qui est une version étalée du spectre temps continu original.

Une manière simple d'améliorer les caractéristiques spectrales d'un signal échantillonné est d'utiliser les fenêtres de lissage. Lorsque vous utilisez l'analyse spectrale ou de Fourier sur des données d'une longueur finie, vous pouvez utiliser les fenêtres pour minimiser les fronts de transition de vos signaux tronqués, réduisant ainsi la perte spectrale. Utilisé de cette manière, les fenêtres de lissage fonctionnent comme des filtres passe-bas prédéfinis et à bande étroite.

Perte spectrale et fenêtres de lissage

Lorsque vous utilisez la DFT/FFT pour trouver la représentation en fréquence d'un signal, on suppose que vos données sont une période unique d'un signal se répétant périodiquement. Ceci est représenté dans la figure 14-1. La première période représentée est la période échantillonnée. Le signal correspondant à cette période est alors répété dans le temps pour produire le signal périodique.

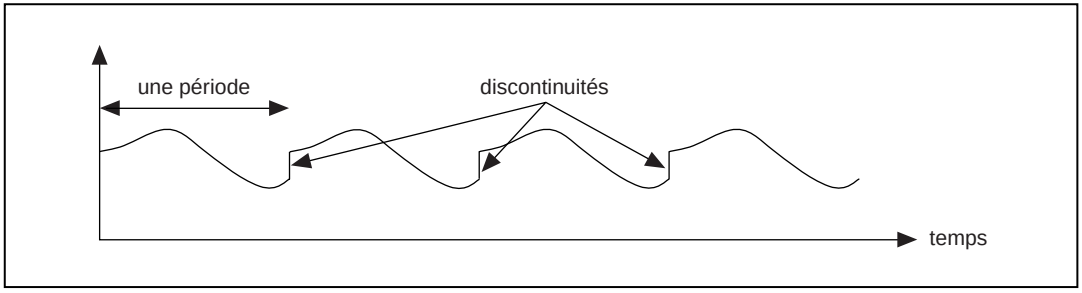


Figure 14-1. Signal périodique créé à partir d'une période échantillonnée

Comme présenté dans la figure précédente, parce que nous supposons le signal périodique, des discontinuités se produisent entre les périodes successives. Ceci survient lorsque vous échantillonnez un nombre non entier de cycles. Ces discontinuités *artificielles* se révèlent être de très hautes fréquences dans le spectre du signal, fréquences qui n'étaient pas présentes dans le signal original. Ces fréquences peuvent être bien plus hautes que la fréquence Nyquist, et comme vous l'avez vu précédemment, sont repliées quelque part entre 0 et fs/2. Le spectre que vous obtenez en utilisant la DFT/FFT n'est donc pas le spectre réel du signal original, mais une version étalée. Il apparaît comme si l'énergie d'une fréquence *s'était perdue* dans toutes les autres fréquences. Ce phénomène est appelé la *perte* **spectrale**.

La figure 14-2 montre un signal sinusoïdal et sa transformée de Fourier correspondante. Le signal du domaine temporel échantillonné est représenté dans le graphe 1. Comme la transformée de Fourier suppose la périodicité, vous devez répéter ce signal dans le temps, et le signal périodique ainsi obtenu est représenté dans le graphe 2. La représentation spectrale correspondante est affichée dans le graphe 3. L'enregistrement temporel du graphe 2 est périodique, mais sans discontinuité, son spectre est donc une ligne unique représentant la fréquence du signal sinusoïdal. La raison pour laquelle le signal du graphe 2 n'a pas de discontinuité est que

vous avez échantillonné un nombre entier de cycles (dans ce cas, 1) du signal temporel.

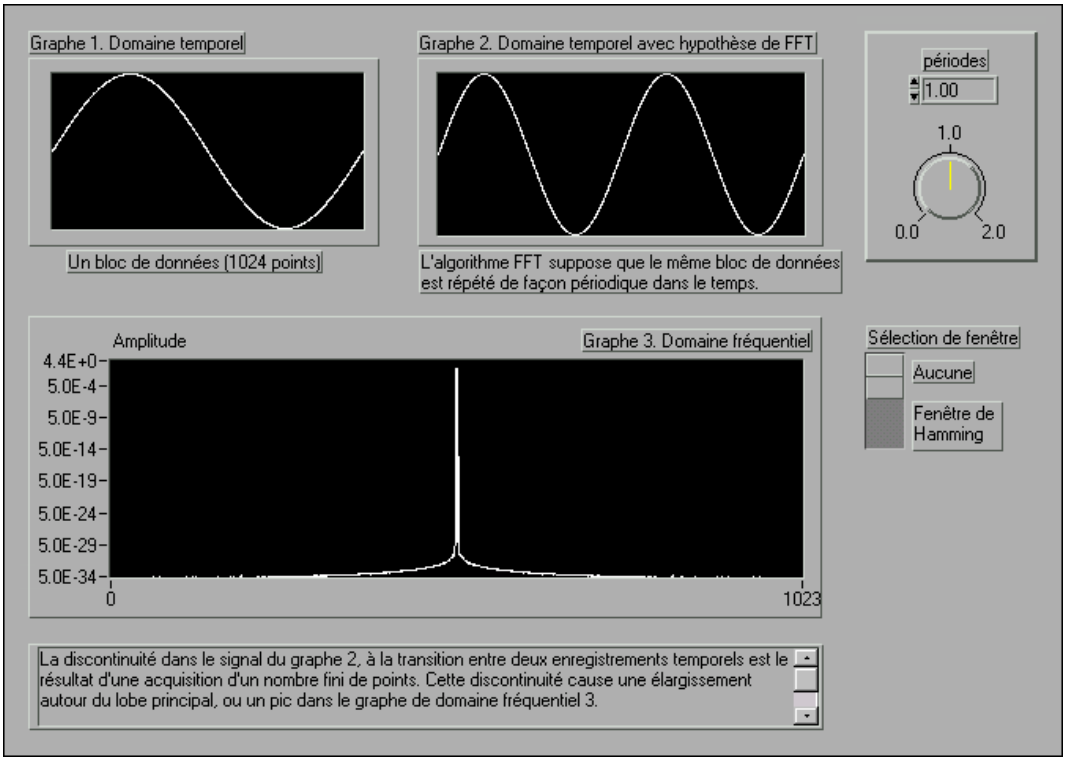


Figure 14-2. Signal sinusoïdal et transformée de Fourier correspondante

Dans la figure 14-3, vous voyez la représentation spectrale lorsque vous échantillonnez un nombre non entier de cycles du signal temporel (ici, 1,25). Le graphe 1 représente maintenant 1,25 cycle du signal sinusoïdal. Lorsque vous répétez ceci de façon périodique, le signal qui en résulte, comme présenté dans le graphe 2, comporte des discontinuités. Le spectre correspondant est représenté dans le graphe 3. Remarquez comment l'énergie est maintenant distribuée sur une large gamme de

fréquences. Cet étalement de l'énergie est la *perte spectrale*. L'énergie a quitté l'une des lignes FFT et s'est étalée sur toutes les autres lignes.

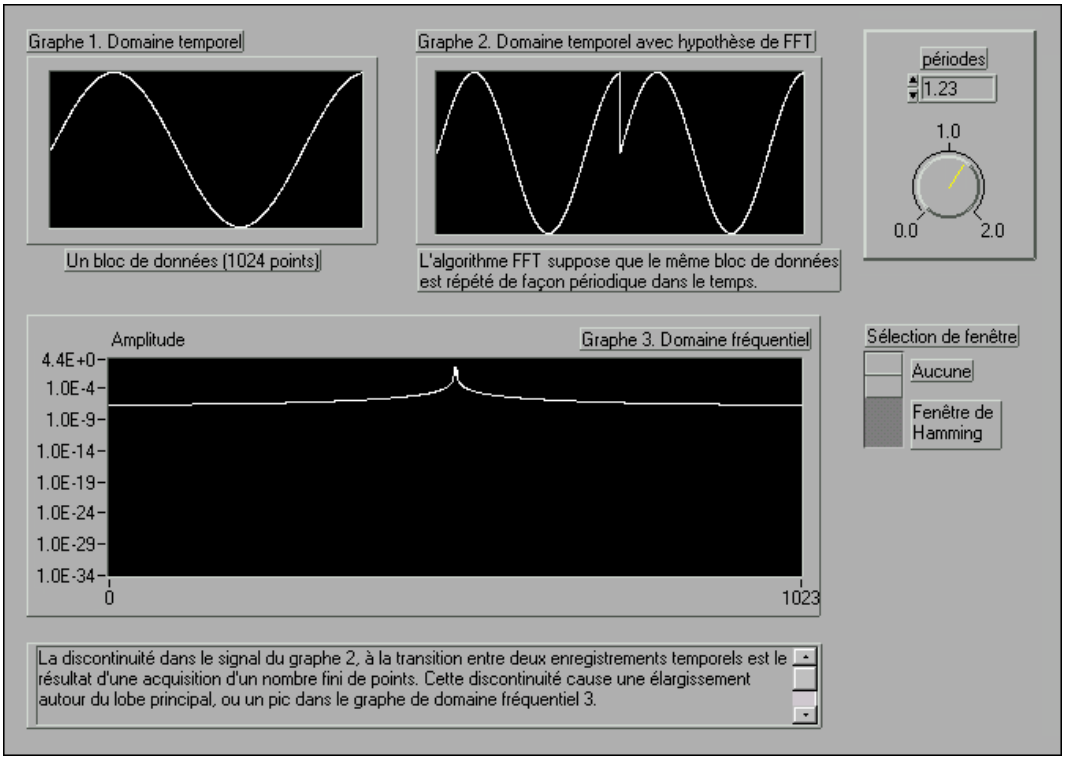


Figure 14-3. Représentation spectrale lors de l'échantillonnage d'un nombre non entier d'échantillons

La perte existe à cause de l'enregistrement temporel fini du signal d'entrée. Pour maîtriser la perte, une solution est de prendre un enregistrement temporel infini, de moins l'infini à plus l'infini. La FFT calcule ensuite une ligne unique à la fréquence correcte. Attendre pendant un temps infini est, cependant, impossible en pratique. Aussi, parce que vous êtes limité par un enregistrement temporel fini, une autre technique, appelée le *fenêtrage*, est utilisée pour réduire la perte spectrale.

La quantité de perte spectrale dépend de l'amplitude de la discontinuité. Plus la discontinuité est grande, plus la perte est grande, et vice versa. Vous pouvez utiliser le fenêtrage pour réduire l'amplitude des discontinuités aux limites de chaque période. Il consiste à multiplier l'enregistrement temporel par une fenêtre de longueur finie, dont l'amplitude varie doucement et graduellement vers zéro sur les bords. Ceci est présenté dans

la figure 14-4, où le signal temporel original est fenêtré à l'aide d'une fenêtre de *Hanning*. Remarquez que le signal temporel du signal fenêtré se rapproche graduellement de zéro aux extrémités. Ainsi, en réalisant l'analyse spectrale ou de Fourier sur des données de longueur finie, vous pouvez utiliser les fenêtres pour minimiser les bords de transition de votre signal échantillonné. Une fonction de rafraîchissement de fenêtre appliquée aux données avant qu'elles ne soient transformées en domaine fréquentiel minimise la perte spectrale.

Remarquez que si l'enregistrement temporel contient un nombre entier de cycles, comme présenté dans la figure 14-2, la supposition de périodicité n'aboutit pas à des discontinuités, et il n'y a donc pas de perte spectrale. Le problème se pose seulement lorsque vous avez un nombre non entier de cycles.

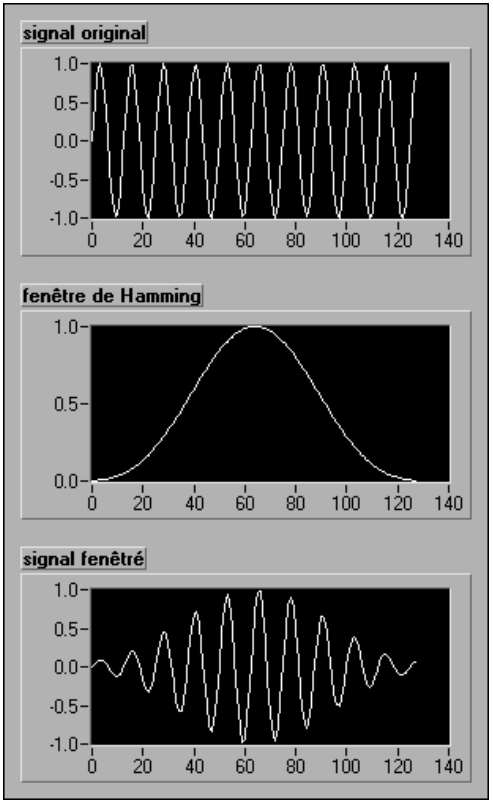


Figure 14-4. Signal temporel fenêtré en utilisant une fenêtre de Hamming

Applications du fenêtrage

Vous pouvez utiliser le fenêtrage pour plusieurs raisons, notamment pour :

- Définir la durée de l'observation.
- Réduire la perte spectrale.
- Séparer un petit signal d'amplitude d'un signal d'amplitude plus large avec des fréquences très proches l'une de l'autre.

Caractéristiques des différents types de fonctions de fenêtrage

Appliquer une fenêtre à (fenêtrage) un signal dans le domaine temporel équivaut à multiplier le signal par la fonction fenêtre. Comme la multiplication dans le domaine temporel est équivalente à la convolution dans le domaine fréquentiel, le spectre du signal fenêtré est une convolution du spectre du signal original avec le spectre de la fenêtre. Ainsi, le fenêtrage change la forme du signal dans le domaine temporel, de même qu'il affecte le spectre que vous voyez.

De nombreux types de fenêtres différents sont disponibles dans la bibliothèque d'analyse LabVIEW. Selon votre application, une fenêtre peut être plus utile que les autres. Quelques-unes de ces fenêtres sont :

Rectangulaire (aucun)

La fenêtre rectangulaire a une valeur de 1 dans son intervalle de temps. Mathématiquement, elle peut être écrite comme :

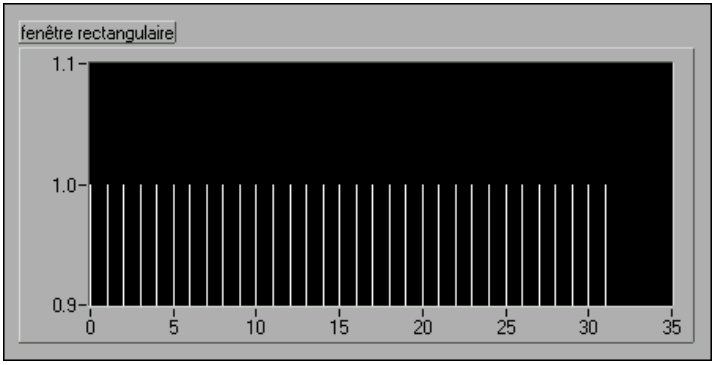
$$w[n] = 1,0$$

pour

$$n = 0, 1, 2.....N-1$$

où *N* est la longueur de la fenêtre. Appliquer une fenêtre rectangulaire équivaut à ne pas utiliser de fenêtre, car la fonction rectangulaire tronque simplement le signal à l'intérieur d'un intervalle de temps fini. La fenêtre rectangulaire subit la perte spectrale la plus grande.

La fenêtre rectangulaire pour $N = 32$ est présentée dans l'illustration suivante :



La fenêtre rectangulaire est utile pour analyser des régimes transitoires qui ont une durée inférieure à celle de la fenêtre. Elle est aussi utilisée pour le *traçage d'ordre*, où la fréquence d'échantillonnage effective est proportionnelle à la vitesse de l'axe dans des machines tournantes. Dans cette application, elle détecte le mode principal de vibration de la machine et ses harmoniques.

Hanning

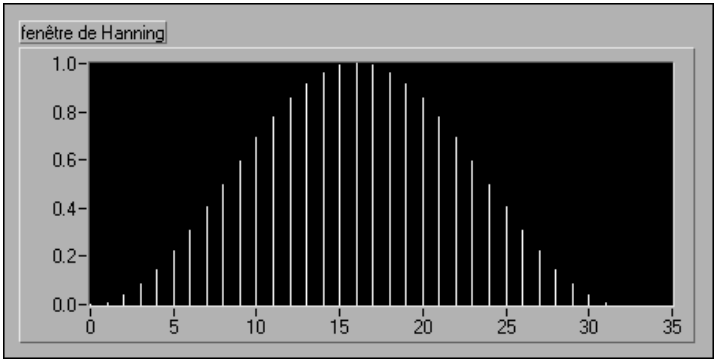
Cette fenêtre a une forme similaire à celle de la moitié d'un cycle de signal cosinusoïdal. L'équation qui la définit est :

$$w(n) = 0{,}5 - 0{,}5\cos(2\pi n/N)$$

pour

$$n = 0, 1, 2, ...N-1$$

Une fenêtre de Hanning avec N = 32 est représentée ci-dessous.



La fenêtre de Hanning est utile pour analyser des régimes transitoires qui ont une durée supérieure à celle de la fenêtre, et aussi pour des applications d'intérêt général.

Hamming

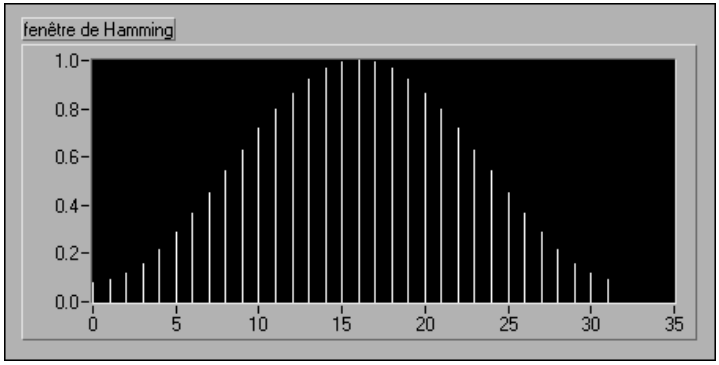
Cette fenêtre est une version modifiée de la fenêtre de Hanning. Sa forme est également similaire à celle d'un signal cosinusoïdal. Elle peut être définie comme :

$$w(n) = 0{,}54 - 0{,}46\cos(2\pi n/N)$$

pour

$$n = 0, 1, 2, ...N-1$$

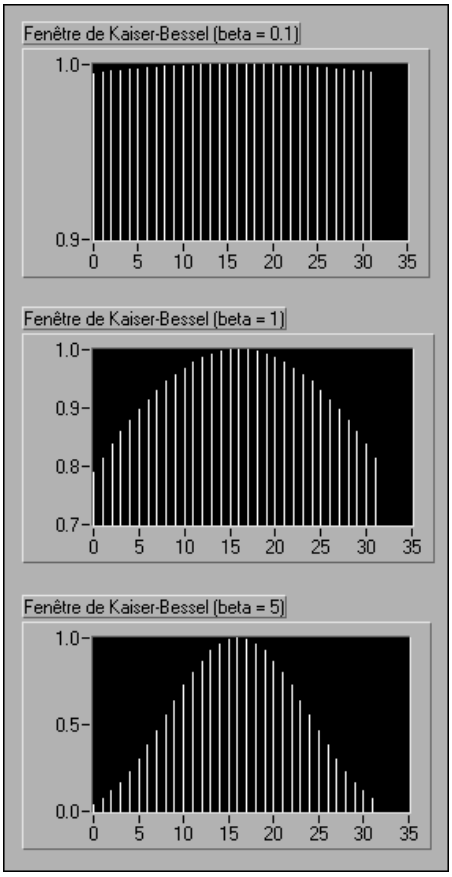
Une fenêtre de Hamming avec N = 32 est présentée ci-dessous.



Vous pouvez constater que les fenêtres de Hanning et Hamming se ressemblent plus ou moins. Cependant, remarquez que dans le domaine temporel, la fenêtre de Hamming ne se rapproche pas autant de zéro sur les bords que la fenêtre de Hanning.

Kaiser-Bessel

Cette fenêtre est une fenêtre “flexible” dont la forme peut être modifiée par l’utilisateur en réglant le paramètre *bêta*. Ainsi, selon votre application, vous pouvez changer la forme de la fenêtre pour contrôler la perte spectrale. La fenêtre de Kaiser-Bessel pour les différentes valeurs de *bêta* est présentée ci-dessous.



Remarquez que pour les petites valeurs de bêta, la forme ressemble à celle d’une fenêtre rectangulaire. En réalité, pour bêta = 0,0, vous obtenez une fenêtre rectangulaire. Lorsque vous augmentez bêta, la fenêtre se termine en pointe sur les côtés.

Cette fenêtre est utile pour détecter deux signaux qui ont presque la même fréquence, mais des amplitudes tout à fait différentes.

Triangle

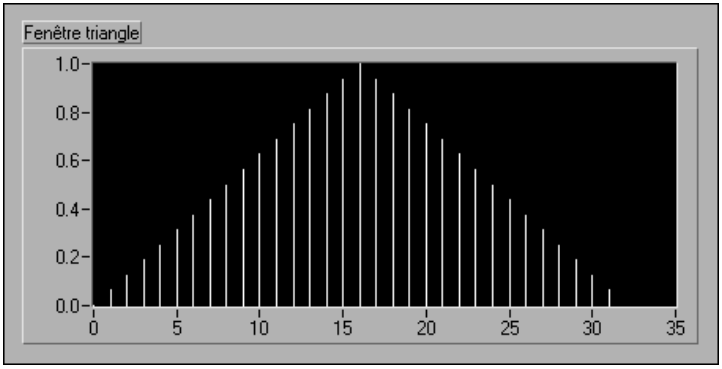
La forme de cette fenêtre est celle d'un triangle. Elle est donnée par :

$$w[n] = 1 - | (2n-N) / N |$$

pour

$$n = 0, 1, 2, ..., n-1$$

Une fenêtre triangulaire pour N = 32 est présentée ci-dessous.



A profil plat

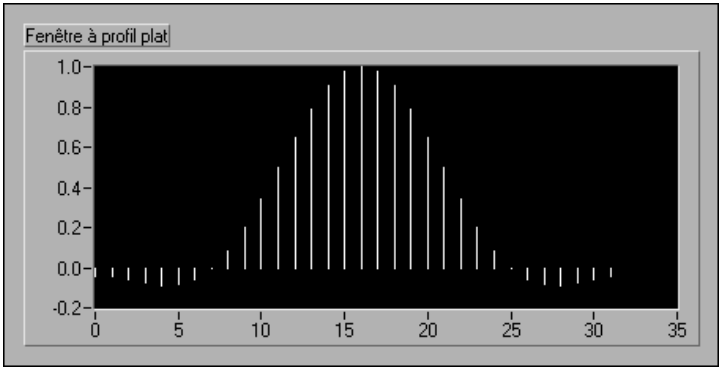
Cette fenêtre possède la meilleure précision d'amplitude de toutes les fonctions de fenêtre. L'augmentation de la précision de l'amplitude (± 0,02 dB pour des signaux exactement entre des cycles intégraux) se fait aux dépens de la sélectivité de la fréquence. La fenêtre à profil plat est surtout utile pour mesurer précisément l'amplitude de composantes de fréquence unique avec peu d'énergie spectrale dans le signal. La fenêtre à profil plat peut être définie comme :

$$w(n) = a0 - a1*\cos(2\pi n/N) + a2*\cos(4\pi n/N)]$$

où

$$a0 = 0,2810638602$$
$$a1 = 0,5208971735$$
$$a2 = 0,1980389663$$

Une fenêtre à profil plat est présentée ci-dessous.



Exponentielle

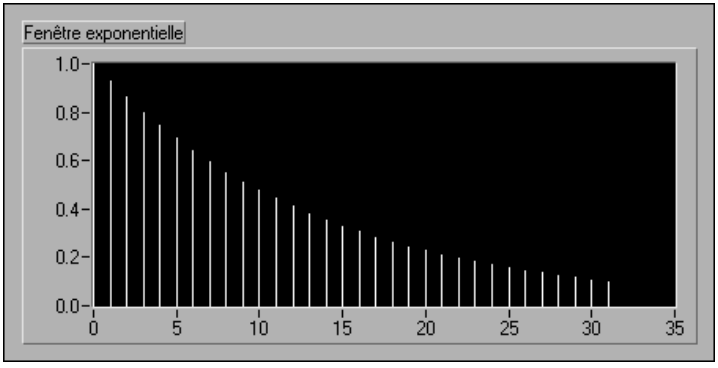
La forme de cette fenêtre est celle d'une exponentielle décroissante. Elle peut être exprimée mathématiquement comme :

$$w[n] = e^{\left(\frac{n \ln(f)}{N-1}\right)} = f^{\left(\frac{n}{N-1}\right)}$$

pour

$$n = 0, 1, 2.....N - 1$$

où *f* est la valeur finale. La valeur initiale de la fenêtre est 1, et elle décroît graduellement vers 0. La valeur finale de l'exponentielle peut être réglée entre 0 et 1. La fenêtre exponentielle pour N = 32, avec la valeur finale spécifiée comme 0,1, est représentée ci-dessous.



Cette fenêtre est utile pour l'analyse de régimes transitoires (signaux qui n'existent que pendant une courte période de temps) dont la durée est plus grande que la longueur de la fenêtre. Cette fenêtre peut être appliquée aux

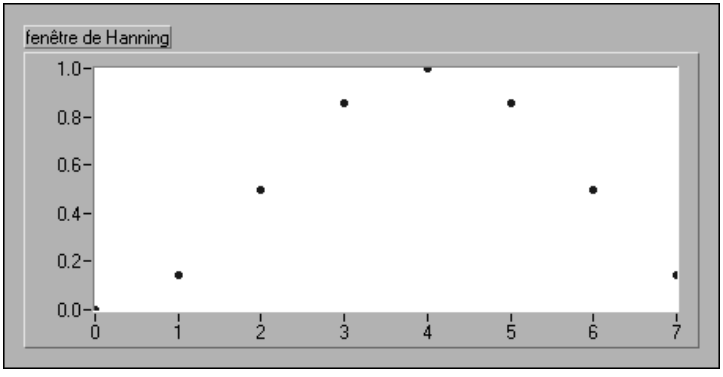
signaux décroissant exponentiellement, tels que la réponse de structures avec un amortisseur faible qui sont excitées par un impact (par exemple, un marteau).

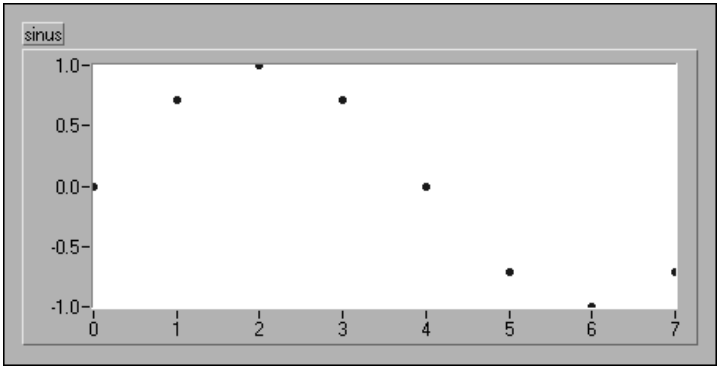
Fenêtres pour l'analyse spectrale et fenêtres pour la conception de coefficients

Les VIs de fenêtre implémentés dans la bibliothèque d'analyse de LabVIEW sont conçus pour des applications d'analyse de spectre. Dans ces applications, le signal d'entrée est fenêtré en le faisant passer à travers un des VIs de fenêtre. Le signal fenêtré est ensuite transféré sur un VI basé sur la DFT pour l'analyse et l'affichage dans le domaine fréquentiel.

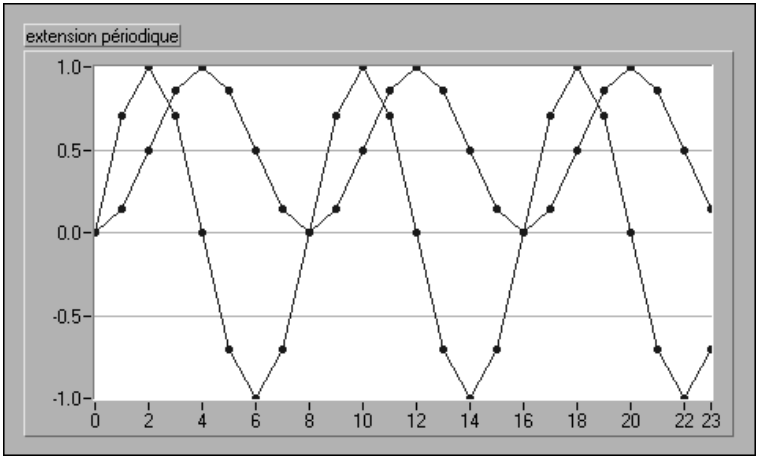
Les fonctions de fenêtre conçues pour l'analyse de spectre doivent être *DFT-paires*, un terme défini par Fredric J. Harris dans son article *On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform (De l'utilisation des fenêtres pour l'analyse harmonique avec la transformée discrète de Fourier)* (*Proceedings of the IEEE*, Volume 66, No.1, janvier 1978). Une fonction de fenêtre est DFT-paire si son produit scalaire (produit intérieur) avec les cycles entiers de séquences sinusoïdales est égal à zéro. Une autre façon d'imaginer une séquence DFT-paire est de concevoir une DFT sans composante imaginaire.

Les figures suivantes illustrent la fenêtre de Hanning et un cycle d'un motif sinusoïdal pour une taille d'échantillon de 8. Les figures ci-dessous montrent que la fenêtre de Hanning, DFT-paire, n'est pas symétrique par rapport à son milieu et que son dernier point n'est pas égal à son premier point, tout à fait comme le cycle complet d'un motif sinusoïdal.





Enfin, la DFT considère les séquences d'entrée comme périodiques, c'est-à-dire que le signal analysé est réellement une concaténation du signal d'entrée. L'illustration suivante montre trois cycles des séquences précédentes, démontrant l'extension périodique rafraîchie de la fenêtre DFT-paire et le motif sinusoïdal à cycle unique.



Un autre type d'application du fenêtrage est la conception de filtre RIF. A ce sujet, reportez-vous à la section *Filtres RIF fenêtrés*, au chapitre 16, *Filtrage*. Cette application a besoin de fenêtres symétriques par rapport à leur centre.

Les équations suivantes de la fonction de fenêtre de Hanning illustrent la différence entre la fonction de fenêtre DFT-paire (analyse de spectre) et la fonction de fenêtre symétrique (conception de coefficients).

La fonction de fenêtre de Hanning pour l'analyse de spectre est :

$$w[i] = 0,5\left(1 - \cos\left(\frac{2\pi i}{N}\right)\right)$$

pour

$$i=0,1,2,\ldots,N-1$$

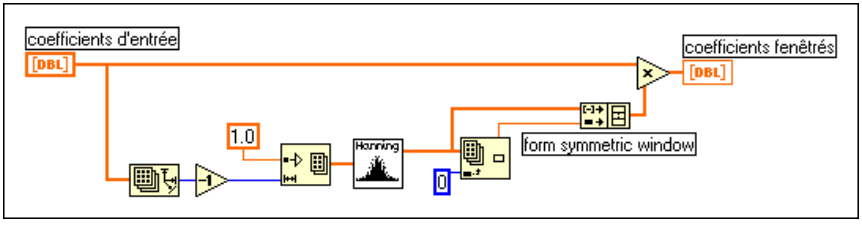
La fonction de fenêtre de Hanning pour la conception de coefficients symétriques est :

$$w[i] = (0,5)\left(1 - \cos\left(\frac{2\pi i}{N-1}\right)\right)$$

pour

$$i=0,1,2,\ldots,N-1$$

Les deux équations ci-dessus montrent que vous pouvez implémenter des fonctions de fenêtre symétriques en modifiant légèrement l'utilisation des sfonctions de fenêtre DFT-paires. L'illustration suivante présente un diagramme qui utilise le VI "Fenêtre de Hanning" (Hanning Window) pour implémenter le fenêtrage symétrique des coefficients filtres.



Voyez l'annexe A, *Références d'analyse*, pour plus d'informations sur les fenêtres de lissage.

Quel type de fenêtres utiliser ?

Maintenant que vous avez vu plusieurs des nombreux types de fenêtres disponibles, vous pouvez vous poser la question "Quel type de fenêtre dois-je utiliser ?". La réponse dépend du type de signal que vous avez et de ce que vous recherchez. Choisir une fenêtre correcte demande une connaissance préalable du signal que vous analysez. En résumé, le tableau suivant présente les différents types de signaux et les fenêtres appropriées que vous pouvez utiliser avec eux.

Type de signal	Fenêtre
Transitoires dont la durée est inférieure à la longueur de la fenêtre	Rectangulaire
Transitoires dont la durée est supérieure à la longueur de la fenêtre	Exponentielle, Hanning
Applications générales	Hanning
Repérage d'ordre	Rectangulaire
Analyse de système (mesures de réponse en fréquence)	Hanning (pour l'excitation aléatoire), Rectangulaire (pour l'excitation pseudo-aléatoire)
Séparation de deux sons avec des fréquences très proches l'une de l'autre, mais avec des amplitudes largement différentes	Kaiser-Bessel
Séparation de deux sons avec des fréquences très proches l'une de l'autre, et avec des amplitudes presque identiques	Rectangulaire
Mesures précises d'amplitude de son unique	A profil plat

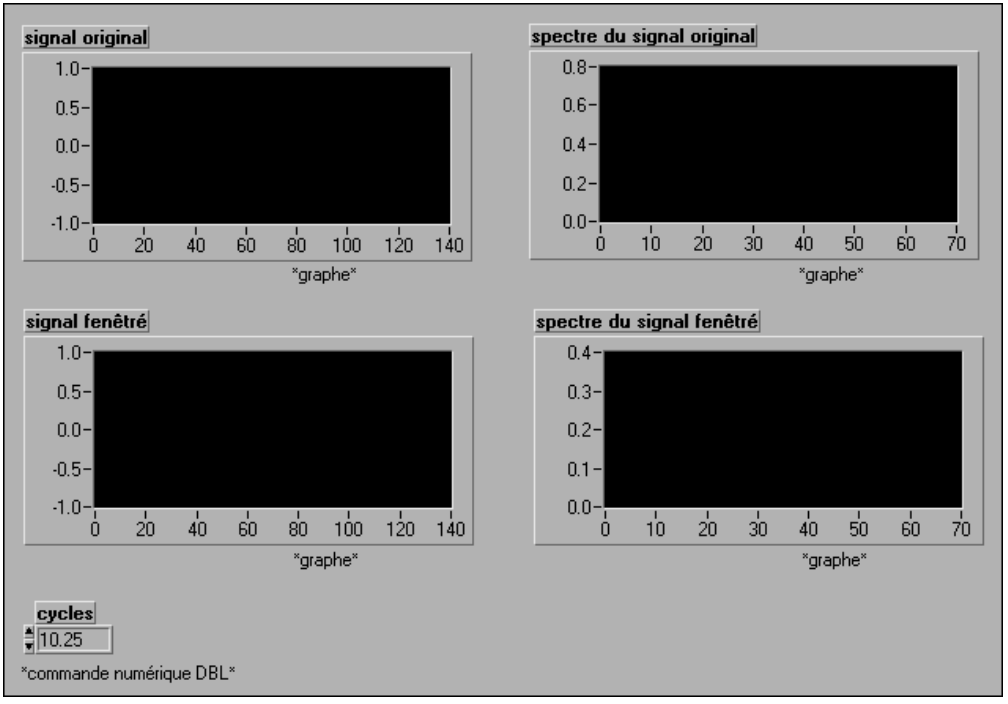
Dans de nombreux cas, vous n'aurez peut-être pas une connaissance préalable du signal suffisante, aussi vous devrez tester différentes fenêtres afin de trouver celle qui vous convient.

Exercice 14-1. Comparer un signal fenêtré avec un signal non fenêtré

Votre objectif est d'observer la différence (pour les domaines temporels et fréquentiels) entre un signal fenêtré et un signal qui ne l'est pas.

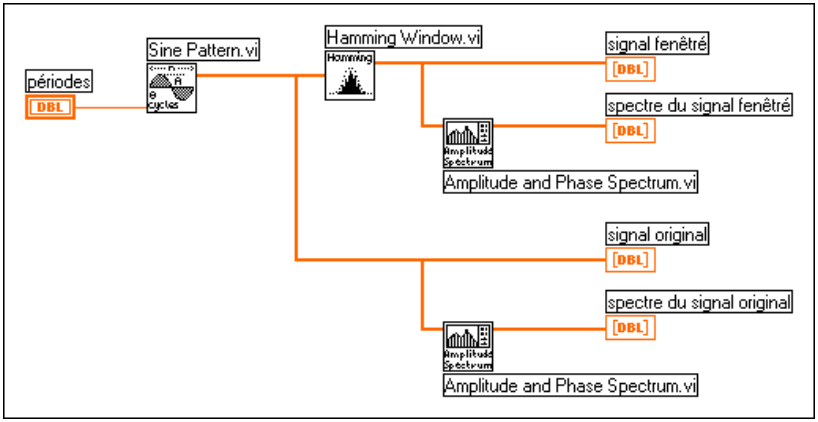
Face-avant

1. Ouvrez une nouvelle face-avant et créez les objets comme indiqué dans l'illustration suivante.



Diagramme

2. Construisez le diagramme indiqué dans l'illustration suivante.



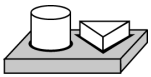
Le VI "Motif sinusoïdal" (**Sine Pattern**) (palette **Fonctions»Analyse» Génération de signal**) génère un signal sinusoïdal avec le nombre de cycles spécifié par la commande **cycles**.

Le signal temporel du signal sinusoïdal est fenêtré en utilisant le VI "Fenêtre Hamming" **(Hamming Window)** (palette **Fonctions»Analyse» Fenêtres**), et les signaux temporels fenêtrés et non fenêtrés sont affichés sur les deux tracés à gauche de la face-avant.

Le VI "Spectre de phase et d'amplitude" **(Amplitude and Phase Spectrum)** (palette **Fonctions»Analyse»Mesure**) obtient le spectre d'amplitude des signaux temporels fenêtrés et non fenêtrés. Ces signaux sont affichés sur les deux tracés sur la droite de la face-avant.

3. Enregistrez le VI sous `Signal fenêtré et défenêtré.vi` dans le répertoire `LabVIEW\Activity`.
4. Définissez le nombre de **cycles** sur 10 (un nombre entier) et exécutez le VI. Remarquez que le spectre du signal fenêtré est plus large que le spectre du signal non fenêtré. Les deux spectres sont toutefois concentrés près de 10 sur l'axe des X.

5. Changez le nombre de **cycles** sur 10,25 (un numéro non entier) et exécutez le VI. Remarquez que le spectre du signal non fenêtré est désormais plus étendu. C'est parce que vous avez un nombre de cycles non entier et que vous répétez le signal pour le rendre périodique, obtenant ainsi des discontinuités. Le spectre du signal fenêtré est toujours concentré, mais celui du signal non fenêtré est maintenant étalé sur toute la fréquence de domaine. (Il s'agit de la perte spectrale.)
6. Faites passer le nombre de **cycles** sur 10,5 et observez les tracés du domaine fréquentiel. La perte spectrale du signal original est clairement apparente.



Fin de l'exercice 14-1.

15

Analyse et mesure de spectre

Ce chapitre montre comment déterminer le spectre d'amplitude et de phase, développer un analyseur de spectre, et déterminer la distorsion harmonique totale (THD) de vos signaux. Pour des exemples d'utilisation des VIs de mesure, consultez les exemples qui se trouvent dans la bibliothèque `examples\analysis\measure\measxmpl.llb`.

Introduction aux VIs de mesure

Plusieurs VIs de mesure réalisent des transformations couramment utilisées de domaine temporel en domaine fréquentiel, telles que le spectre de phase et d'amplitude, le spectre de puissance du signal, la fonction de transfert de réseau, etc. D'autres VIs de mesure interagissent avec des VIs qui réalisent des fonctions comme le fenêtrage du domaine temporel mis à l'échelle et l'estimation de la fréquence et de la puissance.

Vous pouvez utiliser les VIs de mesure pour les applications suivantes :

- Les applications d'analyse de spectre
  - Spectre d'amplitude et de phase
  - Spectre de puissance
  - Fenêtre du domaine temporel mise à l'échelle
  - Estimation de la puissance et de la fréquence
  - Mesures de distorsion harmonique totale (THD) et d'analyse harmonique
- Applications d'analyse à deux voies et en réseau
  - Réponse impulsionnelle
  - Fonctions de réseau (y compris la cohérence)
  - Spectre de puissance croisé

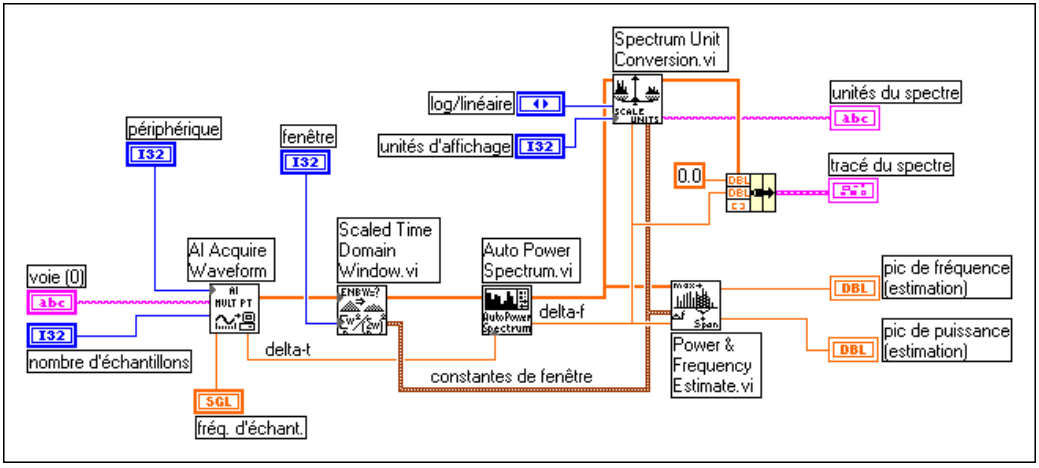
La DFT, la FFT et le spectre de puissance servent à mesurer les composantes fréquentielles des signaux stationnaires ou transitoires. La FFT fournit la composante fréquentielle moyenne du signal pour la durée de l'acquisition. Pour cette raison, vous utilisez la FFT surtout pour l'analyse de signaux stationnaires (lorsque les composantes fréquentielles

du signal ne changent pas de façon significative pendant l'acquisition), ou lorsque vous voulez seulement l'énergie moyenne de chaque raie de fréquence. De nombreux problèmes de mesure tombent dans cette catégorie. Pour mesurer les informations de fréquence qui varient pendant l'acquisition, vous devez utiliser les VIs d'analyse de joint temps-fréquence (Joint time-frequency analysis), comme le spectrogramme Gabor.

Les VIs de mesure sont construits au-dessus des VIs de traitement de signaux et possèdent les caractéristiques suivantes, qui régissent le comportement des instruments d'analyse de fréquence de banc traditionnels.

- On suppose une entrée de signal de domaine temporel, du monde réel.
- Les sorties sont mises à l'échelle en amplitude et en phase, et en unités si nécessaire, prêtes pour une représentation graphique immédiate.
- Des spectres unilatéraux de DC à : $\frac{\text{fréquence d'échantillonnage}}{2}$
- Une conversion de la période d'échantillonnage en un intervalle de fréquence pour la représentation graphique avec des unités d'axe des X appropriées (en Hz).
- Des corrections pour la fenêtre utilisée s'appliquent si nécessaire.
- Des fenêtres sont mises à l'échelle pour que chaque fenêtre donne le même résultat concernant les pics d'amplitude du spectre à l'intérieur des contraintes de précision d'amplitude.
- Des représentations des spectres de puissance ou d'amplitude dans différents formats d'unités, comme les décibels et les unités de densité spectrales, telles que V^2/Hz, $V/\sqrt{Hz}$, etc.

En général, vous pouvez connecter directement les VIs de mesure à la sortie des VIs d'acquisition de données et aux graphes au moyen d'un cluster d'axe, comme le montre le diagramme d'analyseur de spectre suivant.



Les exemples de mesure incluent :

- Des exemples de spectre d'amplitude
- Des exemples d'analyse de signal dynamique simulé
- Des exemples de distorsion harmonique totale (THD)

Vous pouvez utiliser les exemples suivants avec le matériel de National Instruments.

- L'analyseur de spectre simple et l'analyseur de spectre (Simple Spectrum Analyzer et Spectrum Analyzer) : les deux fonctionnent avec tout matériel d'entrée analogique (utilisez le matériel d'acquisition de signaux dynamiques pour des mesures de bonne qualité).
- L'analyseur de signaux dynamiques et l'analyseur de réseau (Dynamic Signal Analyzer et Network Analyzer) : les deux fonctionnent avec du matériel d'acquisition de signal dynamique (DSA).

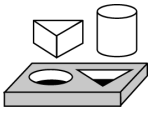
Vous apprendrez

- A utiliser les VIs de mesure pour réaliser différentes opérations de traitement de signaux.
- Comment calculer le spectre de fréquence (amplitude et phase) d'un signal du domaine temporel, avec les unités appropriées.
- Comment calculer la réponse de fréquence d'un système, en traitant les signaux de réponse et d'impulsion, avec les unités appropriées.
- Comment calculer la fonction de cohérence et comment l'utiliser pour comprendre vos mesures de réponse de fréquence.
- Comment déterminer la distorsion harmonique totale présente dans un signal.

Analyse de spectre

Calcul du spectre d'amplitude et de phase d'un signal

Dans beaucoup d'applications, connaître les composantes fréquentielles d'un signal donne un aperçu du système qui a généré le signal. Vous pouvez utiliser les informations obtenues pour analyser les composantes fréquentielles de sons, calibrer des instruments, estimer les bruits et les vibrations générés par des machines, etc. Le prochain exercice vous explique comment utiliser le VI de spectre de phase et d'amplitude pour mesurer l'amplitude et la phase d'un signal.

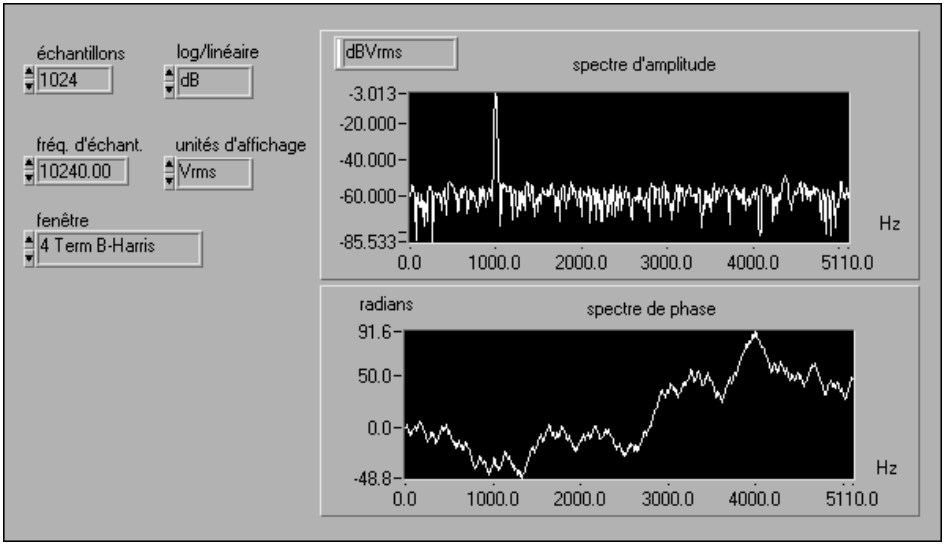


Exercice 15-1. Utiliser le VI de spectre de phase et d'amplitude

Dans cet exercice, votre objectif est de calculer le spectre d'amplitude et de phase d'un signal.

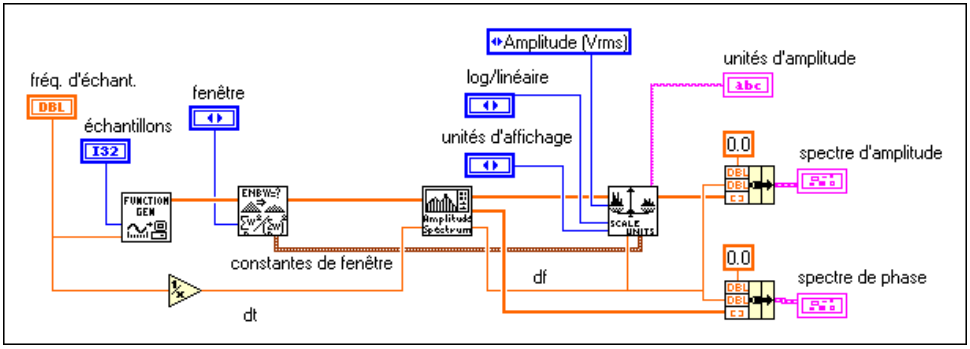
Face-avant

1. Ouvrez le VI "Exemple de spectre d'amplitude" (Amp Spectrum Example) qui se trouve dans la bibliothèque `examples\analysis\measure\measxmpl.llb`. Le signal est généré par le VI "Générateur de fonction simple", qui simule un générateur multi-fonction avec du bruit blanc additif.



Diagramme

2. Ouvrez et examinez le diagramme.

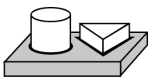


Le VI “Spectre d’amplitude et de phase” (Amplitude and Phase Spectrum) calcule le spectre d’amplitude et le spectre de phase d’un signal du domaine temporel. Les connexions à ce VI sont présentées ci-dessous.

Le signal du domaine temporel d’entrée est appliqué à la commande Signal (V). L’amplitude et la phase du spectre du signal d’entrée sont disponibles aux sorties, respectivement, Amp Spectrum Mag (Vrms) et Amp Spectrum Phase (radiant). Le VI Conversion d’unité de spectre (Spectrum Unit Conversion) est utilisé pour convertir la sortie originale Vrms du spectre d’amplitude et de phase en n’importe quelle autre unité (Vrms, Vpk,

Vrms^2, Vpk^2, $\text{Vrms}\sqrt{\text{Hz}}$, $\text{Vpk}\sqrt{\text{Hz}}$, Vrms^2/Hz et Vpk^2/Hz). Les quatre dernières unités sont la densité spectrale d'amplitude ($\text{Vrms}\sqrt{\text{Hz}}$, $\text{Vpk}\sqrt{\text{Hz}}$) et la densité spectrale de puissance (Vrms^2/Hz et Vpk^2/Hz). Le cluster de sortie **Constantes de fenêtre** dans le VI "Fenêtre de domaine temporel mis à l'échelle" (Scaled Time Domain Window.vi) contient des constantes pour la fenêtre sélectionnée nécessaires aux mesures de densité spectrale.

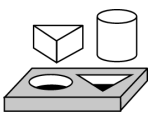
3. Exécutez le VI.
4. Exécutez l'exemple Spectre d'amplitude (Amp Spectrum example) avec la face-avant "Générateur de fonction simple" (Simple Function Generator) toujours ouverte, pour que vous puissiez changer le type de fréquence et de signal simulés de même que le niveau de bruit et d'amplitude du signal. Remarquez les changements dans le spectre d'amplitude.



Fin de l'exercice 15-1.

Calcul de la réponse en fréquence d'un système

Mesurer le contenu de la fréquence de signaux individuels est utile, mais la réponse en fréquence de systèmes est couramment utilisée pour analyser le comportement de toutes sortes de réseau, de l'impédance de composantes électriques à l'analyse de la fréquence vibratoire naturelle de structures dynamiques. La réponse en fréquence caractérise tout à fait la façon dont un réseau répond à une entrée donnée.

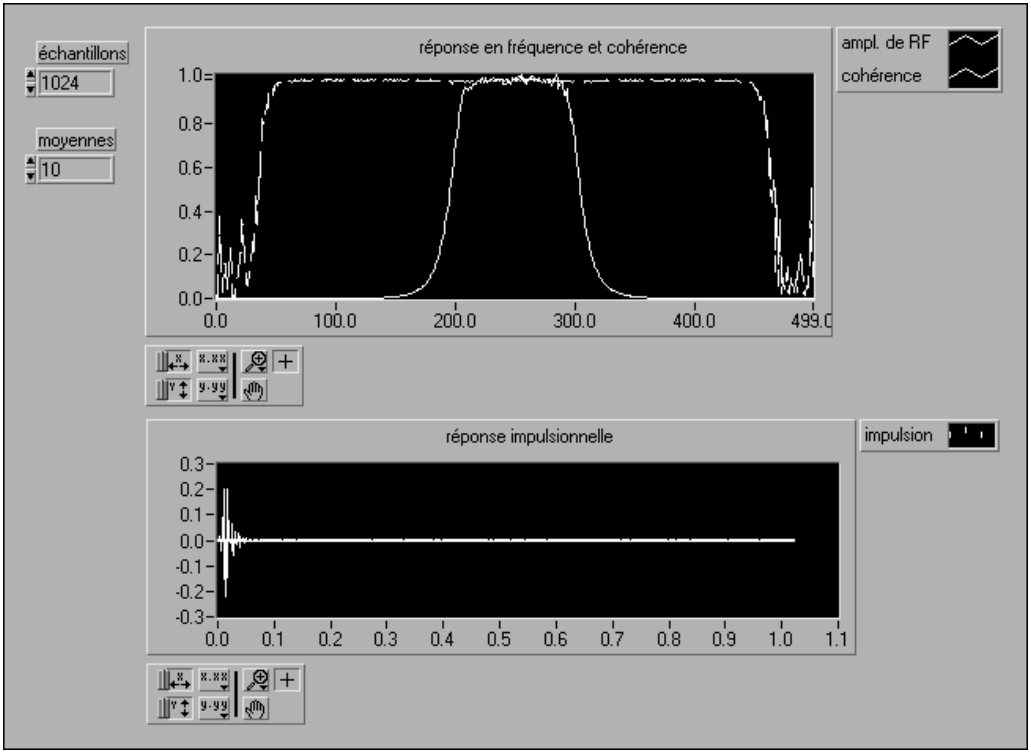


Exercice 15-2. Calculer la réponse en fréquence et impulsionnelle

Votre objectif est de calculer la réponse en fréquence et impulsionnelle d'un système, puis de calculer la fonction de cohérence, et de comprendre comment elle est utilisée pour valider vos mesures de réponses en fréquence.

Face-avant

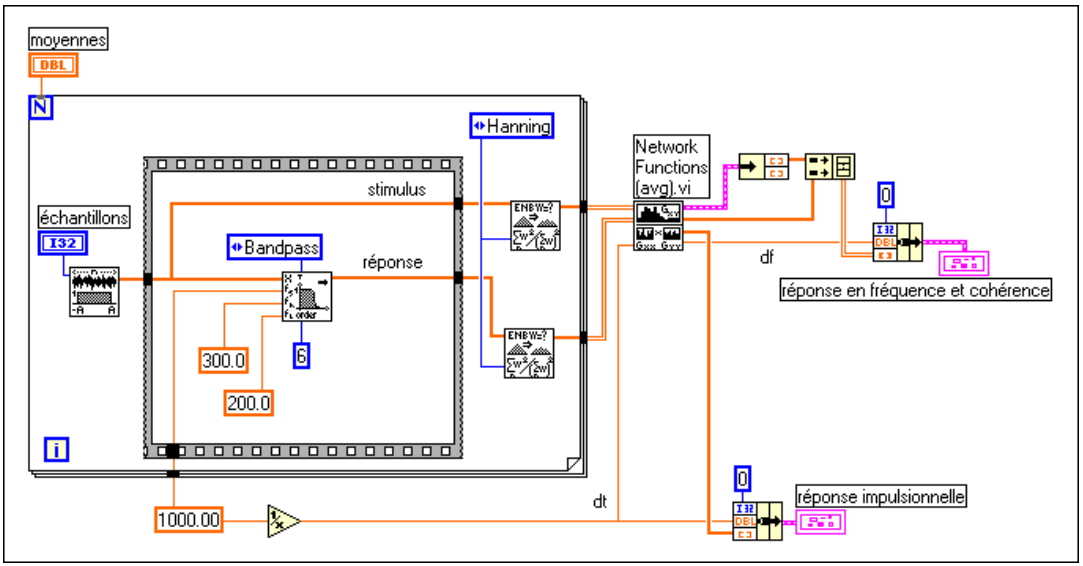
1. Ouvrez une nouvelle face-avant et ajoutez les objets comme présenté dans l'illustration suivante. Cette face-avant affiche l'amplitude de la réponse en fréquence et la fonction de réponse impulsionnelle pour un filtre passe-bande. La fonction de cohérence est tracée à la même échelle que l'amplitude de réponse en fréquence car c'est aussi une mesure spectrale.



Diagramme

2. Ouvrez le diagramme et modifiez-le comme indiqué dans l'illustration suivante. Nous mesurons ici la réponse système d'un filtre passe-bande (VI "Filtre Butterworth" [Butterworth Filter.vi]) en utilisant le bruit blanc (VI "Bruit blanc uniforme" [Uniform White Noise.vi]) comme l'impulsion du système, et en recueillant la sortie du filtre comme la réponse du système. L'impulsion et la réponse sont fenêtrées par la fenêtre de Hanning (VI" Fenêtre de domaine temporel mise à l'échelle" [Scaled Time Domain Window.vi]) et le système entier est surveillé par un certain de nombre de cadres, ou *moyennes*.

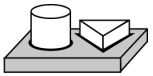
L'impulsion et la réponse sont alors envoyées au VI "Fonctions réseau (moyenne)" (Network Functions [avg]) où sont effectués tous les calculs réels liés à la réponse en fréquence du système.



Le VI "Fonctions réseau (moyenne)" (Network Functions [avg].vi) calcule la réponse en fréquence (amplitude et phase), le spectre de puissance croisée (amplitude et phase), la fonction de cohérence et la réponse impulsionnelle. En augmentant le nombre de cadres des données d'entrée et de sortie (augmentation des moyennes sur la face-avant), les estimations des fonctions de réponse du système s'améliorent. Dans ce diagramme, seules sont tracées l'amplitude de la réponse de fréquence, la cohérence et la réponse impulsionnelle.

La fonction de cohérence mesure le degré de la corrélation du signal de sortie avec le signal d'entrée, donnant ainsi une indication de la validité de votre estimation de réponse en fréquence. Le bruit injecté et le comportement du système, non linéaire à certaines fréquences, font que la fonction de cohérence baisse en dessous de l'unité à ces fréquences. Pour un bruit de système non corrélé, plus on utilise des moyennes, plus la fonction de cohérence approche l'unité et meilleure est l'estimation de la réponse en fréquence. Dernier point à noter concernant la fonction de cohérence : elle n'est définie que lorsque vous calculez la moyenne de plus d'un cadre de donnée d'entrée et de sortie. Pour une moyenne seule, la

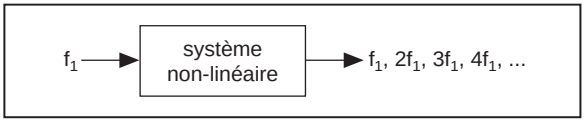
cohérence est l'unité à toutes les fréquences, même lorsque vos estimations de la réponse en fréquence semblent pauvres.



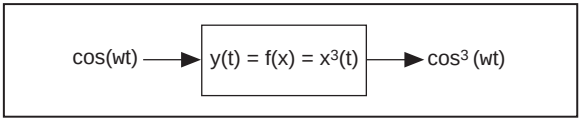
Fin de l'exercice 15-2.

Distorsion harmonique

Lorsqu'un signal, *x(t)*, d'une fréquence particulière (par exemple, f1) est transmis à un système non linéaire, la sortie du système contient non seulement la fréquence d'entrée (f1), mais aussi ses harmoniques (f2 = 2*f1, f3 = 3*f1, f4 = 4*f1, etc). Le nombre d'harmoniques et leurs amplitudes correspondantes générées, dépendent du degré de non linéarité du système. En général, plus la non linéarité est importante, plus l'amplitude des harmoniques est élevée, et vice versa.



Un exemple de système non linéaire est un système où la sortie *y(t)* est le signal d'entrée *x(t)* au cube.



Ainsi, si l'entrée est :

$$x(t) = \cos(\omega t),$$

la sortie est :

$$x^3(t) = 0{,}5*\cos(\omega t) + 0{,}25*[\cos(\omega t) + \cos(3\omega t)]$$

Par conséquent, la sortie contient non seulement la fréquence d'entrée fondamentale de ω, mais aussi la troisième harmonique 3ω.

Distorsion harmonique totale

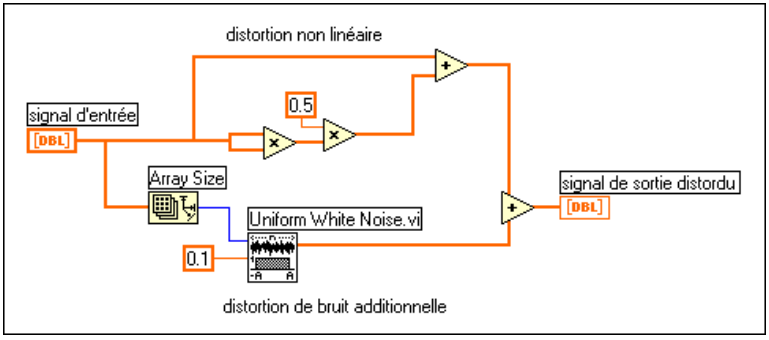
Pour déterminer la distorsion non linéaire introduite par un système, vous devez mesurer les amplitudes des harmoniques introduites par le système par rapport à l'amplitude du fondamental. La distorsion harmonique est une mesure relative des amplitudes des harmoniques comparées à l'amplitude du fondamental. Si l'amplitude du fondamental est A_1, et que les amplitudes des harmoniques sont A_2 (deuxième harmonique), A_3 (troisième harmonique), A_4 (quatrième harmonique), ...A_N (*Nième* harmonique), la distorsion harmonique totale (THD) est

$$\text{THD} = \text{racine carrée de } (A_1^2 + A_2^2 + A_3^2 + \ldots A_N^2)/A_1$$

et le pourcentage de distorsion harmonique total (% THD) est :

$$\% \text{ THD} = 100*\text{racine carrée de } (A_1^2 + A_2^2 + A_3^2 + \ldots\ldots A_N^2)/A_1$$

Dans le prochain exercice, vous générerez un signal sinusoïdal et le ferez passer dans un système non linéaire. Le diagramme du système non linéaire est présenté ci-dessous.



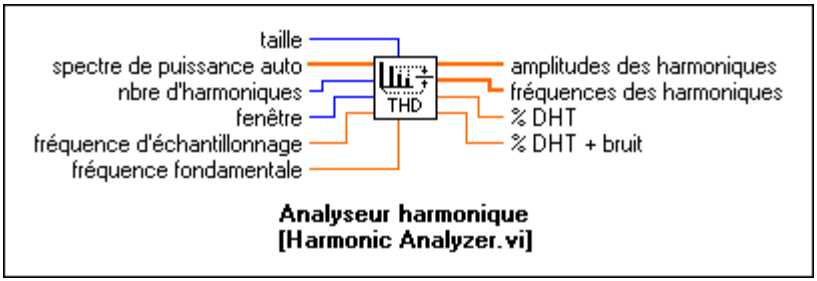
Vérifiez dans le diagramme que si l'entrée est $x(t) = \cos(\omega t)$, la sortie est :

$$\begin{aligned} y(t) &= \cos(\omega t) + 0{,}5\cos^2(\omega t) + 0{,}1n(t) \\ &= \cos(\omega t) + [1 + \cos(2\omega t)]/4 + 0{,}1n(t) \\ &= 0{,}25 + \cos(\omega t) + 0{,}25\cos(2\omega t) + 0{,}1n(t) \end{aligned}$$

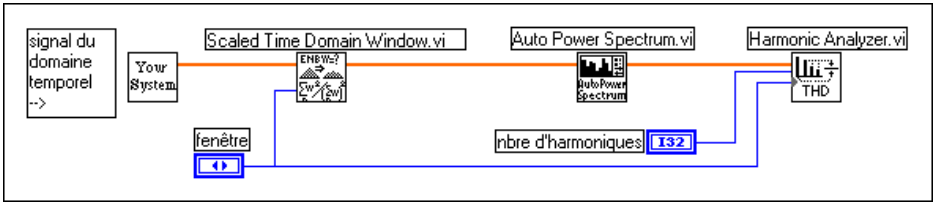
Ainsi, ce système non linéaire génère une composante continue supplémentaire de même que le deuxième harmonique.

Utilisation du VI Analyseur harmonique

Vous pouvez utiliser le VI “Analyseur harmonique” (Harmonic Analyzer.vi) pour calculer le pourcentage de THD présente dans le signal à la sortie du système non linéaire. Le VI retourne les composantes harmoniques et fondamentales (leurs amplitudes et les fréquences correspondantes) présentes dans le spectre de puissance appliqué à son entrée, puis calcule le pourcentage de distorsion harmonique totale (%THD) et le pourcentage de distorsion harmonique totale plus le bruit (%THD + bruit). Les connexions avec le VI “Analyseur harmonique” sont présentées ci-dessous.



Pour utiliser ce VI, vous devez lui donner le spectre de puissance du signal pour lequel vous voulez que le VI calcule la distorsion THD. Ainsi, dans cet exemple, vous devez faire les connexions suivantes.



Le VI “Fenêtre de domaine temporel mise à l’échelle” (Scaled Time Domain Window.vi) applique une fenêtre à la sortie y(t) du système non linéaire (**Votre système**). Elle passe ensuite au VI “Spectre de puissance automatique” (Auto Power Spectrum.vi), qui envoie le spectre de puissance de y(t) au VI “Analyseur harmonique” (Harmonic Analyzer.vi), qui calcule les amplitudes et les fréquences des harmoniques, la THD et le pourcentage de distorsion THD.

Vous pouvez préciser le nombre d’harmoniques à trouver par le VI dans la commande **Nombre d’harmoniques**. Leurs amplitudes et les fréquences correspondantes sont retournées dans les indicateurs de tableau **Amplitudes harmoniques** et **Fréquences harmoniques**.

Remarque ***Le nombre spécifié dans la commande Nombre d'harmoniques inclut la composante fondamentale. Donc, si vous entrez une valeur de 2 dans la commande Nombre d'harmoniques, vous allez trouver le fondamental (de fréquence f1 par exemple) et la deuxième harmonique (de fréquence f2 = 2*f1). Si vous entrez une valeur de N, le VI trouve la fondamentale et les (N−1) harmoniques suivantes.***

Quelques-unes des autres commandes sont expliquées ci-dessous :

La **fréquence fondamentale** est une estimation de la fréquence de la composante fondamentale. Si elle est laissée sur zéro (la valeur par défaut), le VI utilise la fréquence de la composante non DC avec la plus haute amplitude comme la fréquence fondamentale.

La **fenêtre** est le type de fenêtre que vous appliquez à votre signal temporel original. C'est la fenêtre que vous sélectionnez dans le VI "Fenêtre de domaine temporel mise à l'échelle" (Scaled Time Domain Window.vi). Pour une estimation précise de la THD, nous vous recommandons de choisir une fonction de fenêtre. Le paramètre par défaut est la fenêtre uniforme.

La **fréquence d'échantillonnage** est la fréquence d'échantillonnage d'entrée en Hz.

La sortie *% THD + bruit* nécessite davantage d'explications. Les calculs pour *% THD + bruit* sont presque identiques aux calculs pour *% THD*, sauf que la puissance de bruit est aussi ajoutée à celle des harmoniques. Elle se calcule par :

$$\%\ \text{THD + bruit} = 100 * \text{racine carrée}(\text{somme}(\text{APS}))/\text{A1}$$

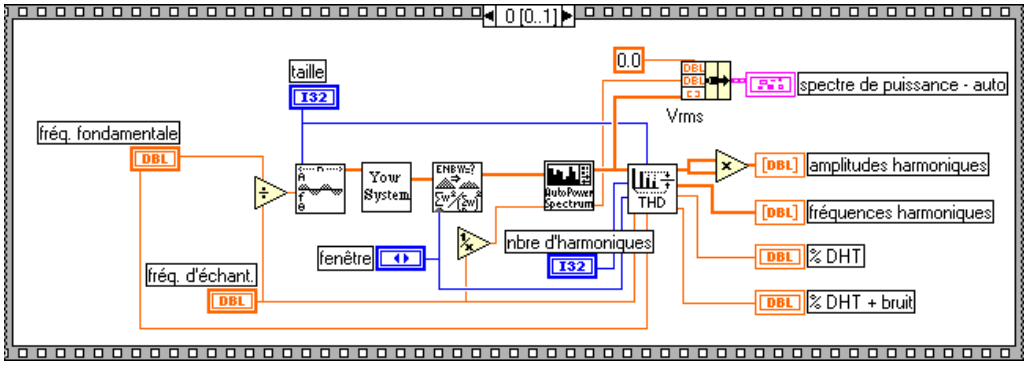
où somme(APS) est la somme des éléments du spectre de puissance automatique, moins les éléments proches de la composante et de l'indice de la fréquence fondamentale.

Exercice 15-3. Calculer la distorsion harmonique

Votre objectif est d'utiliser le VI d'analyseur harmonique pour les calculs de distorsion harmonique.

Diagramme

1. Ouvrez le VI d'exemple **THD (THD Example)** dans la bibliothèque `examples\analysis\measxmpl.llb` et affichez le diagramme.

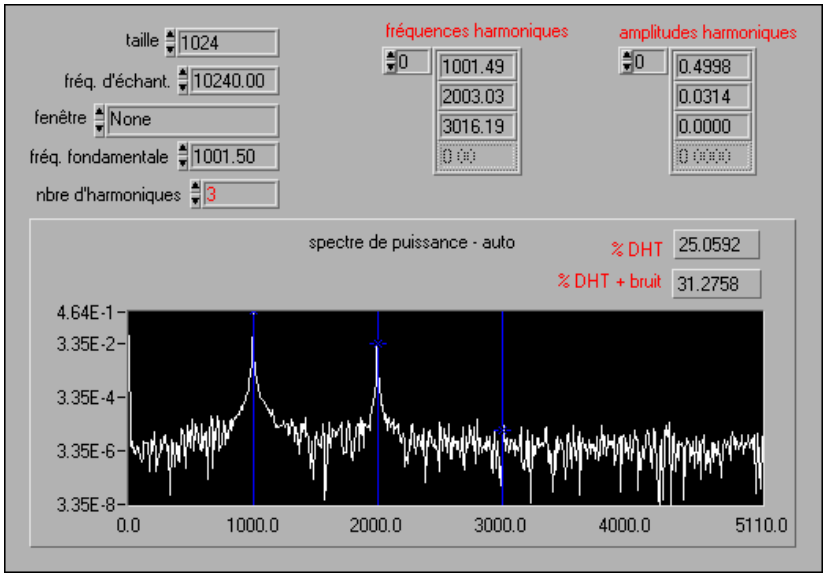


Une partie de ce diagramme vous est déjà familière. Votre système est le système non linéaire que vous avez vu précédemment. Sa sortie est fenêtrée, et le spectre de puissance est calculé et donné au VI d'analyseur harmonique.

Le VI de signal sinusoïdal génère une fréquence fondamentale spécifiée dans la commande **fréquence fondamentale**.

Face-avant

2. Affichez la face-avant. Dans la partie inférieure, vous voyez un tracé du spectre de puissance de la sortie du système non linéaire. En haut à droite, se trouvent les indicateurs de tableau pour les fréquences et les amplitudes du fondamental et de ses harmoniques. La taille du tableau dépend de la valeur entrée de la commande **Nombres d'harmoniques**.



3. Modifiez la valeur de la **fréquence fondamentale** sur 1000, **nombre d'harmoniques** sur 2 et exécutez le VI plusieurs fois. A chaque fois, remarquez les valeurs dans les indicateurs de sortie (**Fréquences harmoniques**, **Amplitudes harmoniques**, **% THD**, et **% THD + bruit**).

   Pourquoi obtenez-vous des valeurs différentes chaque fois que vous exécutez le VI ?

   Laquelle des valeurs, % THD ou % THD + bruit, est la plus grande ? Pouvez-vous expliquer pourquoi ?

4. Exécutez le VI en utilisant différentes fenêtres et observez les pics dans le spectre de puissance.

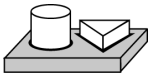
   Quelle fenêtre donne les pics les plus étroits ? Les plus larges ? Pouvez-vous expliquer pourquoi ?

5. Modifiez la fréquence fondamentale sur 3000 et exécutez le VI.

   Pourquoi obtenez-vous une erreur ?

   Idée : considérez la relation entre la fréquence de Nyquist et la fréquence des harmoniques.

6. Lorsque vous avez terminé, fermez le VI et quittez LabVIEW.



Fin de l'exercice 15-3.

En résumé

Vous avez vu que les VIs de mesure peuvent réaliser des opérations de mesure classiques. Certaines de ces opérations incluent le calcul du spectre d'amplitude et de phase d'un signal, et la quantité de distorsion harmonique. D'autres VIs calculent les propriétés d'un système telles que sa fonction de transfert, sa réponse impulsionnelle, le spectre de puissance croisée entre les signaux d'entrée et de sortie, etc. Les VIs prêts à l'utilisation disponibles pour réaliser ces mesures se trouvent dans la sous-palette **Analyse»Mesure**.

16

Filtrage

Ce chapitre explique comment filtrer les fréquences parasites des signaux grâce à des filtres à réponse impulsionnelle infinie (RII), des filtres à réponse impulsionnelle finie (RIF) et des filtres non linéaires. Pour des exemples d'utilisation de VIs de filtre d'analyse, consultez la bibliothèque `examples\analysis\fltrxmpl.llb`.

Introduction aux fonctions de filtrage numérique

La conception de filtre analogique constitue l'un des domaines les plus importants de la conception électronique. Bien qu'il existe des manuels présentant des conceptions de filtre analogique simples et déjà testées, la conception de filtre est souvent réservée aux spécialistes, car elle requiert des connaissances approfondies en mathématique et une compréhension des processus impliqués dans le système affectant le filtre.

Les outils modernes de traitement de signaux numériques et d'échantillonnage permettent de remplacer les filtres analogiques par des filtres numériques dans les applications qui requièrent souplesse et programmabilité, notamment dans le domaine de l'électro-acoustique, des télécommunications, de la géophysique et de la surveillance médicale.

Les filtres numériques offrent les avantages suivants par rapport aux filtres analogiques :

- Ils sont programmables par logiciel.
- Ils sont stables et prévisibles.
- Ils ne varient pas avec la température ou l'humidité et ne requièrent pas des composantes de précision.
- Ils offrent un meilleur rapport performances/coût.

Vous pouvez utiliser des filtres numériques dans LabVIEW pour contrôler des paramètres tels que l'ordre de filtre, les fréquences de coupure, la quantité d'ondulation et l'atténuation de la bande d'arrêt.

Les VIs de filtres numériques décrits dans cette section suivent le principe des instruments virtuels. Les VIs gèrent en interne tous les problèmes de

conception, les calculs, la gestion de mémoire et le filtrage de données réelles, qui deviennent ainsi transparents pour l'utilisateur. Vous n'avez donc pas besoin d'être un expert en théorie des filtres numériques pour traiter les données.

Les paragraphes suivants sur la théorie d'échantillonnage sont destinés à vous faire mieux comprendre les paramètres de filtres et leur relation avec les paramètres d'entrée.

D'après le théorème d'échantillonnage, vous pouvez reconstruire un signal temporel continu à partir d'échantillons discrets et régulièrement espacés, si la fréquence d'échantillonnage est au moins deux fois supérieure à la plus haute fréquence du signal temporel. Supposons que vous puissiez échantillonner le signal temporel concerné à des intervalles Δt espacés de façon égale, sans perdre d'information. Le paramètre Δt représente l'intervalle d'échantillonnage.

Vous pouvez obtenir la fréquence d'échantillonnage f_s à partir de l'intervalle d'échantillonnage

$$f_s = \frac{1}{\Delta t}$$

ce qui signifie que, d'après le théorème d'échantillonnage, la plus haute fréquence pouvant être traitée par le système numérique est

$$f_{Nyq} = \frac{f_s}{2}$$

La plus haute fréquence pouvant être traitée par le système est appelée la fréquence de Nyquist. Ceci concerne également les filtres numériques. Par exemple, si votre intervalle d'échantillonnage est

$$\Delta t = 0{,}001 \text{ s}$$

alors la fréquence d'échantillonnage est

$$f_s = 1.000 \text{ Hz}$$

et la plus haute fréquence pouvant être traitée par le système est donc

$$f_{Nyq} = 500 \text{ Hz}$$

Les types d'opérations de filtrage suivants sont basés sur les techniques de conception de filtre :

- Fenêtres de lissage
- Filtres à réponse impulsionnelle infinie (RII) ou filtres numériques récursifs
- Filtres à réponse impulsionnelle finie (RIF) ou filtres numériques non récursifs
- Filtres non linéaires

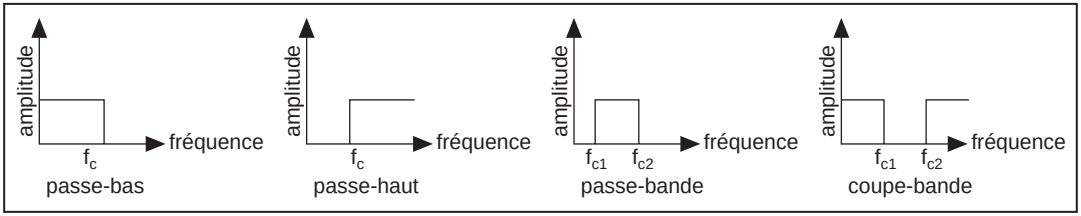
Le reste de ce chapitre présente brièvement la théorie des techniques RII, RIF et non linéaires. Les VIs de filtres numériques correspondant à chaque technique sont également traités. Reportez-vous au chapitre 14, *Fenêtres de lissage*, pour obtenir des informations concernant les VIs utilisés pour effectuer un lissage de fenêtres.

Filtres idéaux

Les filtres modifient ou éliminent les fréquences parasites. Selon la gamme de fréquences qu'ils laissent passer ou atténuent, les filtres se classent dans l'une des catégories suivantes :

- Un *filtre passe-bas* laisse passer les basses fréquences et atténue les hautes fréquences.
- Un *filtre passe-haut* laisse passer les hautes fréquences et atténue les basses fréquences.
- Un *filtre passe-bande* laisse passer une certaine bande de fréquences.
- Un *filtre coupe-bande* atténue une certaine bande de fréquences.

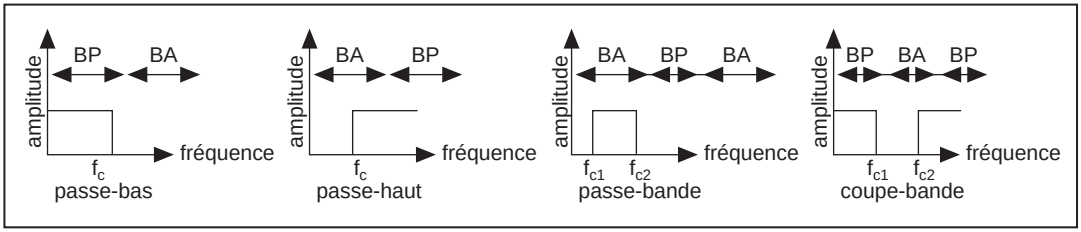
La réponse en fréquence idéale de ces filtres est représentée ci-dessous :



Remarquez que le filtre passe-bas laisse passer toutes les fréquences au-dessous de f_c, tandis que le filtre passe-haut fait passer toutes les fréquences au-dessus de f_c. Le filtre passe-bande laisse passer toutes les fréquences entre f_{c1} et f_{c2}, tandis que le filtre coupe-bande atténue toutes les fréquences

entre f_{c1} et f_{c2}. Les points de fréquence f_c, f_{c1} et f_{c2} sont appelés les fréquences de coupure du filtre. Lorsque vous concevez des filtres, vous devez préciser ces fréquences de coupure.

La gamme de fréquences qui passe à travers le filtre est appelée la *bande passante* (*Passband* [PB]) du filtre. Le gain d'un filtre idéal est d'une unité (0 dB) dans la bande passante, afin que l'amplitude du signal n'augmente ni ne diminue. La *bande d'arrêt* (*Stopband* [SB]) correspond à la gamme de fréquences qui sont rejetées (atténuées). La bande passante et la bande d'arrêt des différents types de filtres sont présentées ci-dessous :

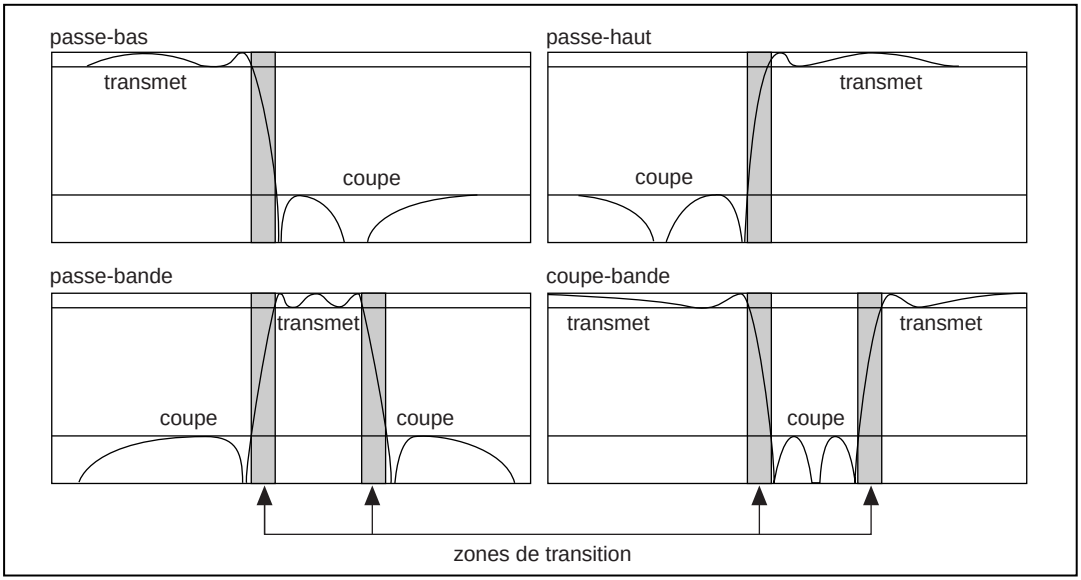


Tandis que les filtres passe-bas et passe-haut possèdent chacun une bande passante et une bande d'arrêt, remarquez que le filtre passe-bande possède une bande passante et deux bandes d'arrêt, et que le filtre coupe-bande comprend deux bandes passantes et une bande d'arrêt.

Filtres réels (non idéaux)

La bande de transition

Dans l'idéal, le gain d'un filtre doit être d'une unité (0 dB) dans la bande passante et de zéro (moins l'infini) dans la bande d'arrêt. Cependant, dans des applications réelles, ces critères ne peuvent pas tous être remplis. En pratique, il y a toujours une région de transition finie entre la bande passante et la bande d'arrêt. Dans cette région, le gain du filtre passe graduellement de 1 (0 dB) dans la bande passante à 0 (moins l'infini) dans la bande d'arrêt. Les diagrammes suivants représentent la bande passante, la bande d'arrêt et la région de transition (Transition Region [TR]) des différents types de filtres non idéaux. Remarquez que la bande passante est désormais la région de fréquence pour laquelle le gain du filtre varie de 0 dB à –3 dB.



Ondulation de bande passante et atténuation de bande d'arrêt

Dans de nombreuses applications, il est permis de faire légèrement varier le gain dans la bande passante (gain idéalement égal à une unité). Cette variation du gain dans la bande passante est appelée l'*ondulation de bande passante* et représente la différence entre le gain réel et le gain d'une unité désiré. L'*atténuation de bande d'arrêt* ne peut pas être infinie en pratique, aussi vous devez préciser une valeur qui vous convient. L'ondulation de bande passante et l'atténuation de bande d'arrêt sont mesurées en décibels (dB). Cette unité est définie par :

$$dB = 20*\log_{10}(A_o(f)/A_i(f))$$

où $\log_{10}$ représente le logarithme décimal à base 10, et $A_i(f)$ et $A_o(f)$ sont les amplitudes d'une fréquence particulière *f*, respectivement avant et après le filtrage.

Par exemple, pour une ondulation bande passante de –0,02 dB, la formule donne :

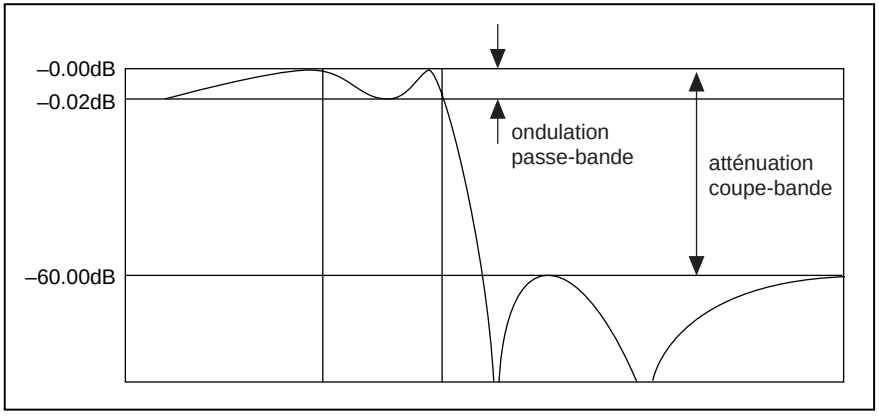
$$-0{,}02 = 20*\log_{10}(A_o(f)/A_i(f))$$
$$A_o(f)/A_i(f) = 10^{-0{,}001} = 0{,}9977$$

ce qui démontre que le rapport des amplitudes d'entrée et de sortie est proche de l'unité.

Si l'atténuation est de –60 dB dans la bande d'arrêt, nous obtenons

$$-60 = 20*\log_{10}(A_o(f)/A_i(f))$$
$$A_o(f)/A_i(f) = 10^{-3} = 0{,}001$$

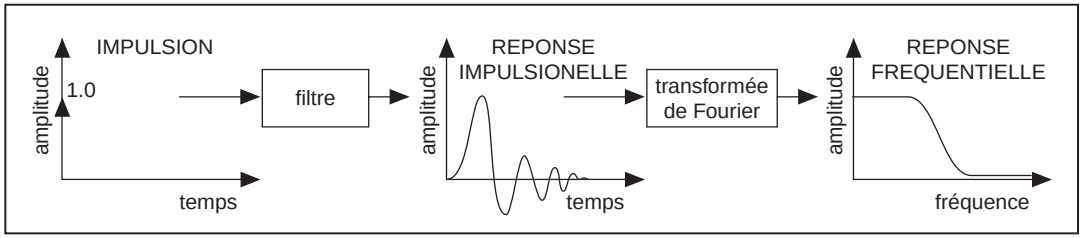
ce qui signifie que l'amplitude de sortie est un millième (1/1000) de l'amplitude d'entrée. La figure suivante (non à l'échelle) illustre ce concept.



Remarque ***L'atténuation est généralement exprimée en décibels sans le terme "moins", bien que l'on suppose habituellement une valeur en dB négative.***

Filtres RII et RIF

Les filtres peuvent être également classés selon leur réponse impulsionnelle. Qu'est-ce au juste qu'une réponse impulsionnelle ? La réponse d'un filtre à une entrée correspondant à une impulsion (*x[0]* = 1 et *x*[i] = 0 pour tout $i \neq 0$) est appelée la *réponse impulsionnelle* du filtre (voir la figure ci-après). La transformée de Fourier de la réponse impulsionnelle est appelée la *réponse en fréquence* du filtre. La réponse en fréquence d'un filtre vous indique quelle va être la sortie du filtre à différentes fréquences. En d'autres termes, cette réponse vous donne le gain du filtre à différentes fréquences. Pour un filtre idéal, le gain doit être égal à 1 dans la bande passante et à 0 dans la bande d'arrêt. Toutes les fréquences de la bande passante passent donc "telles quelles" à la sortie, mais il n'y aucune sortie pour les fréquences de la bande d'arrêt.



Si la réponse impulsionnelle du filtre décroît jusqu'à zéro après un intervalle de temps fini, il s'agit d'un filtre à *réponse impulsionnelle finie (RIF)*. Si la réponse impulsionnelle se prolonge indéfiniment, il s'agit d'un filtre à *réponse impulsionnelle infinie (RII)*. Le fait que la réponse impulsionnelle soit finie ou non (c'est-à-dire que le filtre soit de type RIF ou RII) dépend de la façon dont la sortie est calculée.

La différence fondamentale entre les filtres RIF et RII est la suivante : pour les filtres RIF, la sortie ne dépend que des valeurs d'entrée (présentes et passées), tandis que pour les filtres RII, la sortie dépend non seulement des valeurs d'entrée (présentes et passées), mais aussi des valeurs de sortie passées.

Considérons par exemple une caisse enregistreuse dans un supermarché. Soit $x[k]$ le coût du présent article acheté par un client et soit x(k–1) le prix de l'article précédent, avec $1 \leq k \leq N$, *où N* représente le nombre total d'articles. La caisse enregistreuse ajoute le coût de chaque article pour produire un total "cumulé". Ce total "cumulé" y[k], jusqu'au $k^{ème}$ article, est donné par

$$y[k] = x[k] + x[k–1] + x[k–2] + x[k–3] + ...+ x[1] \qquad (16\text{-}1A)$$

Le total de *N* articles est donc *y[N]*. Si *y[k]* est le total jusqu'au $k^{ème}$ article et *y[k–1]*, le total jusqu'au *(k–1)*^{ème} article, nous pouvons récrire l'équation 16-1A sous la forme

$$y[k] = y[k–1] + x[k] \qquad (16\text{-}1B)$$

Si l'on ajoute une taxe sur les ventes de 8,25%, les équations 16-1A et 16-1B deviennent

$$y[k] = 1{,}0825x[k] + 1{,}0825x[k–1] + 1{,}0825\ x[k–2] + 1{,}0825x[k–3] + ... + 1{,}0825x[1] \qquad (16\text{-}2A)$$

$$y[k] = y[k–1] + 1{,}0825x[k] \qquad (16\text{-}2B)$$

Remarquez que les deux équations 16-2A et 16-2B décrivent de façon similaire le comportement de la caisse enregistreuse. La différence est que l'équation 16-2A n'est implémentée que dans les entrées, tandis que l'équation 16-2B est implémentée dans les entrées et les sorties. L'équation 16-2A est une implémentation *non récursive* ou RIF, tandis que l'équation 16-2B est une implémentation *récursive* ou RII.

Coefficients de filtres

Dans l'équation 16-2A, la constante multiplicative de chaque terme est 1,0825. Dans l'équation 16-2B, les constantes multiplicatives sont 1 (pour *y[k–1]*) et 1,0825 (pour *x[k]*). Ces constantes multiplicatives sont appelées les *coefficients* du filtre. Pour un filtre RII, les coefficients multipliant les entrées sont appelés *coefficients directs* et ceux multipliant les sorties sont appelés *coefficients inverses*.

Les équations de la forme 16-1A, 16-1B, 16-2A ou 16-2B décrivant l'opération du filtre sont appelées équations à *différences*.

La non-linéarité de la réponse en phase des filtres RII est un désavantage. Si l'application ne nécessite pas d'informations concernant la phase, comme un contrôle de signal simple, les filtres RII peuvent s'avérer appropriés. Vous devez utiliser les filtres RIF pour les applications nécessitant des réponses en phase linéaire. La nature récursive des filtres RII les rend plus difficiles à concevoir et à implémenter.

Filtres à réponse impulsionnelle infinie

Les filtres à réponse impulsionnelle infinie (RII) sont des filtres numériques dont les réponses impulsionnelles peuvent théoriquement être infinies en longueur (durée). L'équation à différence générale caractérisant les filtres RII est

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N_b-1} b_j x_{i-j} - \sum_{k=1}^{N_a-1} a_k y_{i-k}\right) \qquad (16\text{-}3)$$

où N_b est le nombre de coefficients *directs* (b_j) et où N_a est le nombre de coefficients *inverses* (a_k).

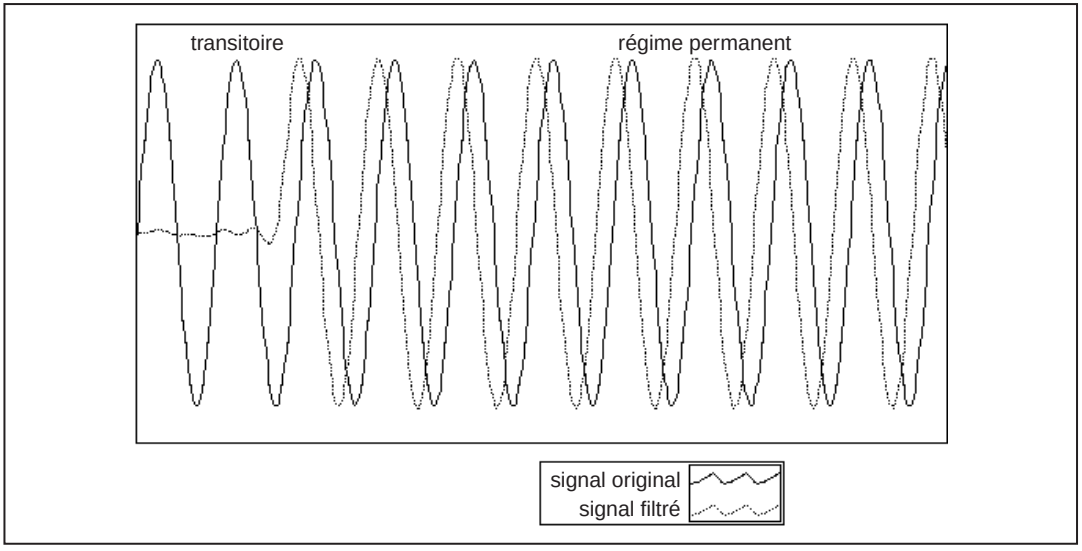
Dans la plupart des conceptions de filtres RII (et pour tous les filtres RII LabVIEW), le coefficient a_0 est 1. L'échantillon de sortie à l'indice d'échantillon présent i est la somme des entrées passées et présentes à l'échelle (x_i et x_{i-j} lorsque $\neq 0$) et des sorties passées à l'échelle (y_{i-k}). Pour cette raison, les filtres RII sont également appelés filtres récursifs ou filtres autorégressifs à moyenne mobile (ARMA [autoregressive moving average]).

La réponse du filtre RII général à une impulsion ($x_0 = 1$ et $x_i = 0$ pour tout $i \neq 0$) est appelée la réponse impulsionnelle du filtre. La réponse impulsionnelle du filtre décrite par l'équation 16-3 est en effet d'une longueur infinie pour des coefficients non nuls. Cependant, dans des applications de filtres pratiques, la réponse impulsionnelle de filtres RII stables décroît jusqu'à une valeur proche de zéro sur un nombre fini d'échantillons.

Les filtres RII de LabVIEW possèdent les propriétés suivantes :

- Les indices négatifs de l'équation 16-3 sont supposés nuls la première fois que vous appelez le VI.
- Comme l'état de filtre initial est supposé être zéro (indices négatifs), un transitoire proportionnel à l'ordre du filtre se produit avant que le filtre n'atteigne un état stable. Pour les filtres passe-bas et passe-haut, la durée de la réponse transitoire, ou retard, est égale à l'ordre du filtre.
- Retard = ordre.
- Pour les filtres passe-bande et coupe-bande, la durée de la réponse transitoire est deux fois l'ordre du filtre.
- Retard = 2 * ordre.

Vous pouvez éliminer cette réponse transitoire lors d'appels successifs en activant la mémoire d'état. Pour cela, mettez la commande **init/cont** du VI sur Vrai (TRUE) (filtrage continu).



Le nombre d'éléments de la séquence filtrée est égal au nombre d'éléments de la séquence d'entrée.

Le filtre conserve les valeurs d'état de filtre interne lorsque le filtrage est terminé.

L'avantage des filtres RII numériques sur les filtres à réponse impulsionnelle finie (RIF) repose sur le fait suivant : les filtres RII requièrent généralement moins de coefficients pour effectuer des opérations de filtrage similaires. Les filtres RII s'exécutent ainsi bien plus rapidement et ne nécessitent pas de mémoire supplémentaire, car ils s'exécutent de manière intégrée.

Le désavantage des filtres RII est que la réponse en phase est non linéaire. Si l'application ne requiert pas d'informations concernant la phase, comme un contrôle de signal simple, les filtres RII peuvent s'avérer appropriés. Vous devez utiliser des filtres RIF pour les applications nécessitant des réponses en phase linéaire.

Filtrage RII en cascade

Les filtres implémentés utilisant la structure définie directement par l'équation 16-4 sont appelés des filtres RII *à forme directe*. Les implémentations directes sont souvent sensibles aux erreurs introduites par la quantification des coefficients et par les limites de précision dues aux calculs. De plus, un filtre conçu pour être stable peut devenir instable si sa longueur (proportionnelle à l'ordre du filtre) de coefficients augmente.

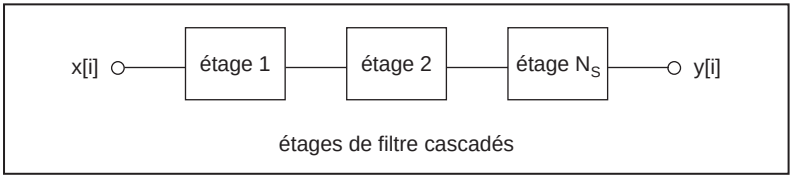
On peut obtenir une structure moins sensible en divisant la fonction de transfert à forme directe en des sections d'ordre inférieur, ou étages de filtres. La fonction de transfert à forme directe du filtre donnée par l'équation 16-4 (avec $a_0 = 1$) peut être écrite sous la forme d'un rapport de transformées en *z*, comme indiqué ci-après :

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_{N_b - 1} z^{-(N_b - 1)}}{1 + a_1 z^{-1} + \ldots + a_{N_a - 1} z^{-(N_a - 1)}} \qquad (16\text{-}4)$$

En factorisant l'équation 16-4 en des sections de second ordre, la fonction de transfert du filtre devient un produit de fonctions de filtres du second ordre

$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{1 + a_{1k} z^{-1} + a_{2k} z^{-2}} \qquad (16\text{-}5)$$

où $N_s = \lfloor N_a/2 \rfloor$ est le plus grand entier $\leq N_a/2$ et $N_a \geq Nb$. (N_s est le *nombre d'étages*). Cette nouvelle structure de filtre peut être décrite comme une *cascade* de filtres du second ordre.



Chaque étage individuel est implémenté en utilisant la structure de filtre à *forme directe II* car il nécessite un nombre minimum d'opérations arithmétiques et un nombre minimum d'éléments de retard (états de filtres internes). Chaque étage possède une entrée, une sortie et deux états antérieurs internes ($s_k[i–1]$ et $s_k[i–2]$).

Si *n* est le nombre d'échantillons dans la séquence d'entrée, l'opération de filtrage s'effectue selon les équations suivantes :

$$y_0[i] = x[i],$$

$$s_k[i] = y_{k-1}[i-1] - a_{1k}s_k[i-1] - a_{2k}s_k[i-2],\ k = 1, 2,..., N_s$$

$$y_k[i] = b_{Ok}s_k[i] + b_{1k}s_k[i-1] + b_{2k}s_k[i-2],\ k = 1, 2,..., N_s$$

$$y[i] = y_{Ns}[i]$$

pour chaque échantillon

$$i = 0, 1, 2,..., n-1.$$

Pour les filtres avec une seule fréquence de coupure (passe-bas et passe-haut), les étages de filtres de second ordre peuvent être conçus directement. Le filtre RII passe-bas ou passe-haut global contient des filtres de second ordre en cascade.

Pour les filtres possédant deux fréquences de coupure (passe-bande et coupe-bande), les étages de filtres du quatrième ordre représentent une forme plus naturelle. Le filtre RII passe-bande ou coupe-bande global contient des filtres du quatrième ordre en cascade. L'opération de filtrage pour des étages du quatrième ordre s'effectue conformément aux équations suivantes :

$$y_0[i] = x[i],$$

$$s_k[i] = y_{k-1}[i-1] - a_{1k}s_k[i-1] - a_{2k}s_k[i-2] - a_{3k}s_k[i-3] - a_{4k}s_k[i-4],\ k = 1, 2,..., N_s$$

$$y_k[i] = b_{0k}s_k[i] + b_{1k}s_k[i-1] + b_{2k}s_k[i-2] + b_{3k}s_k[i-3] + b_{4k}s_k[i-4],\ k = 1, 2,..., N_s$$

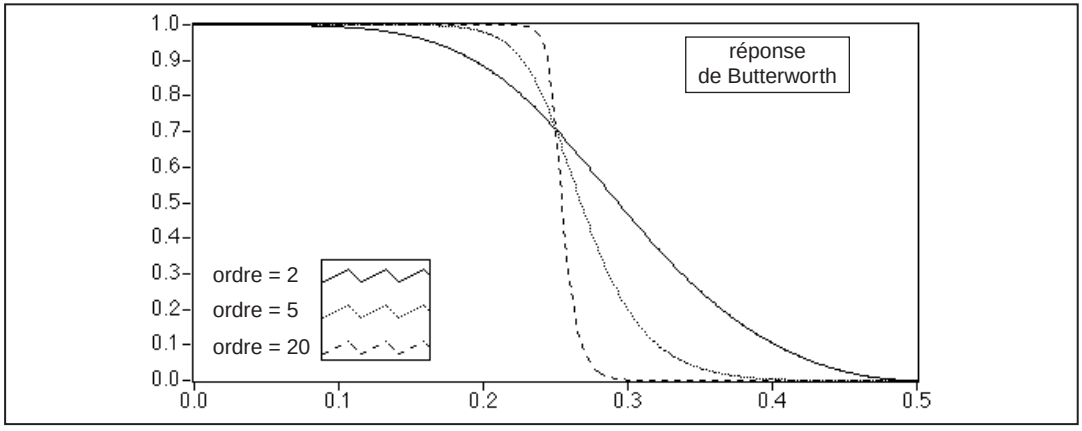
$$y[i] = y_{Ns}[i].$$

Remarquez que dans le cas d'étages de filtres du quatrième ordre, $N_s = \lfloor (N_a + 1)/4 \rfloor$.

Filtres de Butterworth

Une réponse continue à toutes les fréquences et une décroissance monotone à partir des fréquences de coupure précisées caractérisent la réponse en fréquence des filtres de Butterworth. Les filtres de Butterworth sont plats de façon maximale, avec une réponse idéale d'une unité dans la bande passante et de zéro dans la bande d'arrêt. La fréquence à demi-puissance ou la fréquence descendante de 3 dB correspond aux fréquences de coupure précisées.

L'illustration suivante présente la réponse d'un filtre de Butterworth passe-bas. L'avantage des filtres de Butterworth est une réponse en fréquence décroissante monotone continue. Une fois que vous établissez la fréquence de coupure, LabVIEW définit le *taux de variation de la pente* de la transition, qui est proportionnel à l'ordre du filtre. Des filtres de Butterworth d'ordre supérieur se rapprochent de la réponse idéale d'un filtre passe-bas.



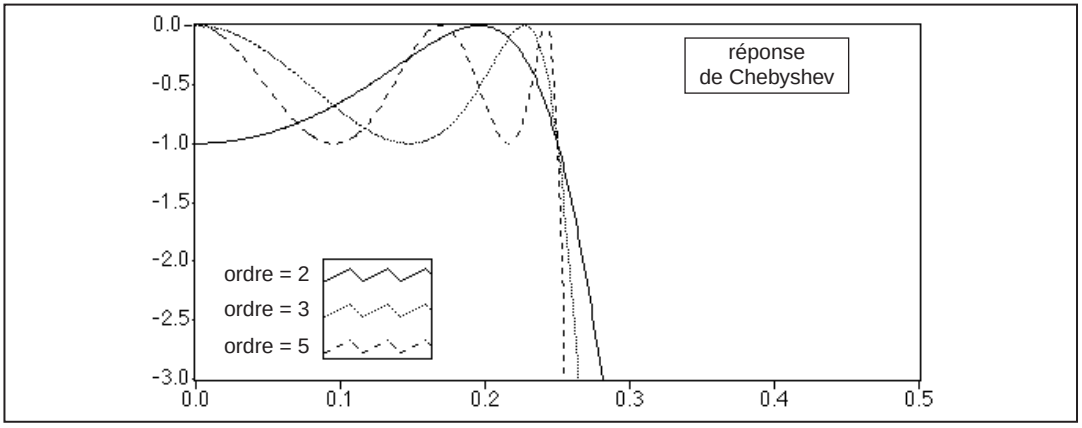
Filtres de Chebyshev

Les filtres de Butterworth ne fournissent pas toujours une bonne approximation de la réponse de filtre idéale à cause de la pente de diminution lente entre la bande passante (la portion intéressante du spectre) et la bande d'arrêt (la portion non désirée du spectre).

Les filtres de Chebyshev minimisent l'erreur maximum de la bande passante en rendant compte de la valeur absolue maximale de la différence entre le filtre idéal et la réponse de filtre que vous souhaitez (l'erreur tolérable maximum dans la bande passante). Les caractéristiques de la réponse en fréquence des filtres de Chebyshev sont une réponse

d’amplitude équi-ondulation dans la bande passante, une réponse d’amplitude monotone décroissante dans la bande d’arrêt et une pente de diminution plus abrupte que celle des filtres de Butterworth.

Le graphe suivant présente la réponse d’un filtre de Chebyshev passe-bas. Remarquez que la réponse équi-ondulation de la bande passante est contrainte par l’erreur d’ondulation tolérable maximum et que la pente de diminution abrupte apparaît dans la bande d’arrêt. L’avantage des filtres de Chebyshev sur les filtres de Butterworth réside dans le fait que les filtres de Chebyshev ont une transition plus abrupte entre la bande passante et la bande d’arrêt avec un filtre d’ordre inférieur. Ceci génère de plus petites erreurs absolues et des vitesses d’exécution plus rapides.

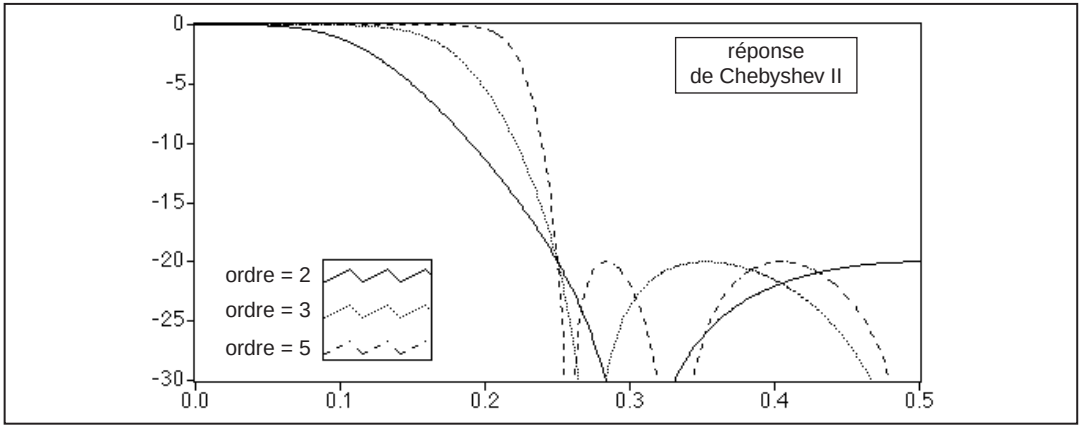


Filtres de Chebyshev inverses ou filtres de Chebyshev II

Les filtres de Chebyshev II, également appelés filtres de Chebyshev inverses ou filtres de Chebyshev de type II, sont similaires aux filtres de Chebyshev classiques, excepté que les filtres de Chebyshev II distribuent l’erreur sur la bande d’arrêt (au lieu de la bande passante), et les filtres de Chebyshev II sont plats de façon maximale dans la bande passante (au lieu de la bande d’arrêt).

Les filtres de Chebyshev II minimisent l’erreur maximum de la bande d’arrêt en rendant compte de la valeur absolue maximale de la différence entre le filtre idéal et la réponse de filtre que vous souhaitez. Les caractéristiques de la réponse en fréquence des filtres de Chebyshev II sont une réponse d’amplitude équi-ondulation dans la bande d’arrêt, une réponse d’amplitude monotone décroissante dans la bande passante et une pente de diminution plus abrupte que celle des filtres de Butterworth.

Le graphe suivant représente la réponse d'un filtre de Chebyshev II passe-bas. Remarquez que la réponse équi-ondulation de la bande d'arrêt est contrainte par l'erreur tolérable maximum et que la pente de diminution continue monotone apparaît dans la bande d'arrêt. L'avantage des filtres de Chebyshev II sur les filtres de Butterworth réside dans le fait que les filtres de Chebyshev II ont une transition plus abrupte entre la bande passante et la bande d'arrêt avec un filtre d'ordre inférieur. Cette différence correspond à une plus petite erreur absolue et à une vitesse d'exécution plus rapide. Un avantage des filtres de Chebyshev II sur les filtres de Chebyshev classiques est que les filtres de Chebyshev II distribuent l'erreur dans la bande d'arrêt au lieu de la bande passante.

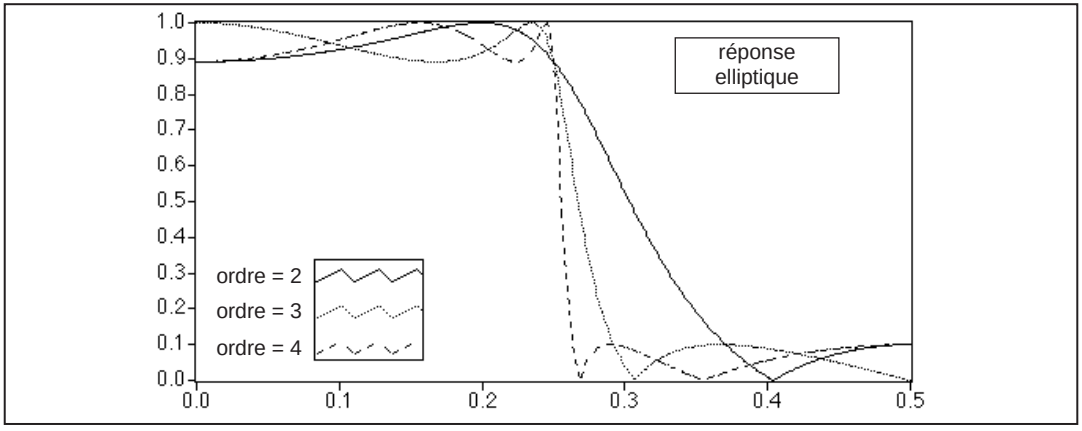


Filtres elliptiques (ou de Cauer)

Les filtres elliptiques minimisent l'erreur maximum en la distribuant sur la bande passante et la bande d'arrêt. Les équi-ondulations de la bande passante et la bande d'arrêt caractérisent la réponse d'amplitude des filtres elliptiques. Si on la compare à la conception des filtres de Butterworth ou de Chebyshev de même ordre, la conception elliptique fournit la transition la plus abrupte entre la bande passante et la bande d'arrêt. C'est la raison pour laquelle les filtres elliptiques sont couramment utilisés.

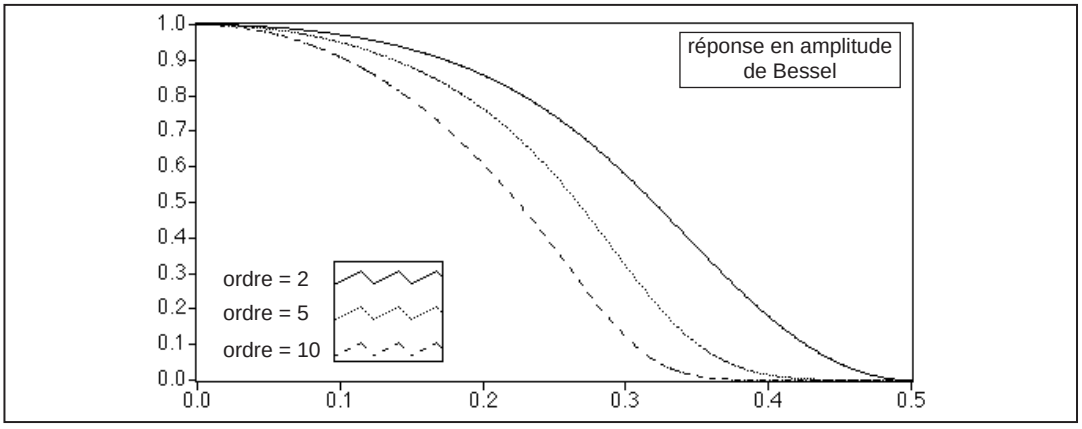
Le graphe suivant représente la réponse d'un filtre elliptique passe-bas. Remarquez que l'ondulation de la bande passante et de la bande d'arrêt est contrainte par la même erreur tolérable maximum (comme spécifié par la

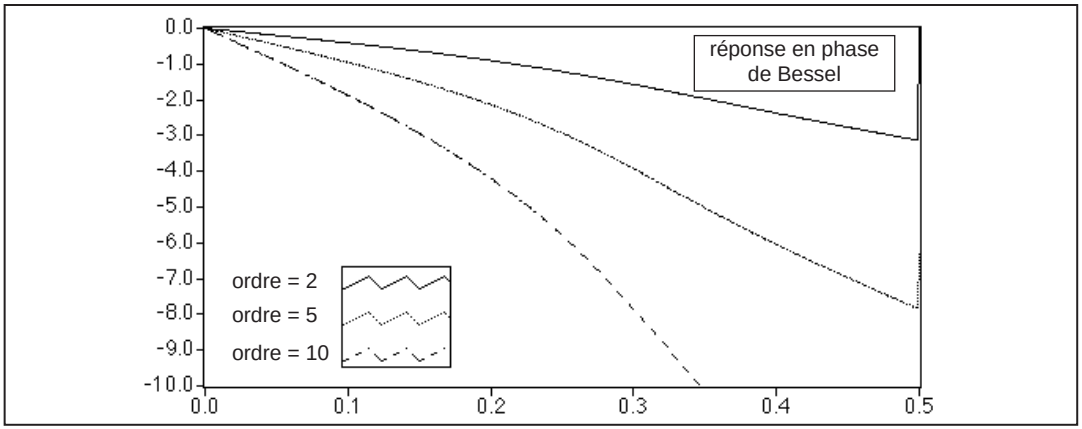
quantité d'ondulation en dB). Remarquez également le front de transition abrupt même pour les filtres elliptiques d'ordre faible.



Filtres de Bessel

Vous pouvez utiliser les filtres de Bessel pour réduire la distorsion de phase non linéaire inhérente à tous les filtres RII. Dans le cas de filtres d'ordre élevé et ceux avec une pente de diminution plus abrupte, cette condition est plus prononcée, en particulier dans les régions de transition des filtres. Les filtres de Bessel ont une réponse plate maximale en amplitude et en phase. De plus, la réponse en phase dans la bande passante des filtres de Bessel (la région qui nous intéresse) est presque linéaire. Tout comme les filtres de Butterworth, les filtres de Bessel nécessitent des ordres élevés pour minimiser l'erreur et sont peu utilisés pour cette raison. Vous pouvez également obtenir une réponse en phase linéaire en utilisant des conceptions de filtres RIF. Les graphes suivants présentent la réponse d'un filtre de Bessel passe-bas. Remarquez que la réponse est continue à toutes les fréquences, et monotone décroissante en amplitude et en phase. Remarquez également que la phase de la bande passante est presque linéaire.





Filtres à réponse impulsionnelle finie

Les filtres RIF sont des filtres numériques dont la réponse impulsionnelle est finie. Les filtres RIF sont également appelés filtres non récursifs, filtres de convolution ou filtres à moyenne mobile, car la sortie d'un filtre RIF peut s'exprimer sous la forme d'une convolution finie

$$y_i = \sum_{k=0}^{n-1} h_k x_{i-k}$$

où x représente la séquence d'entrée à filtrer, y représente la séquence de sortie filtrée et h représente les coefficients de filtre RIF.

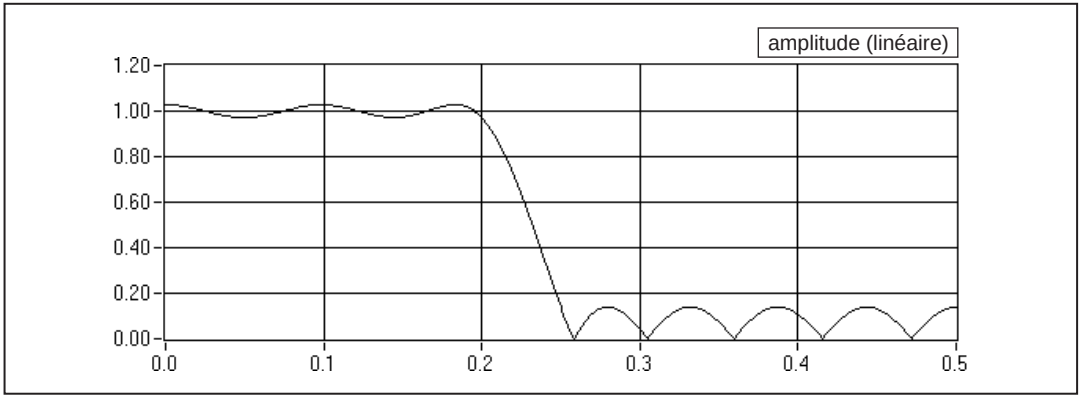
La liste suivante répertorie les caractéristiques principales des filtres RIF :

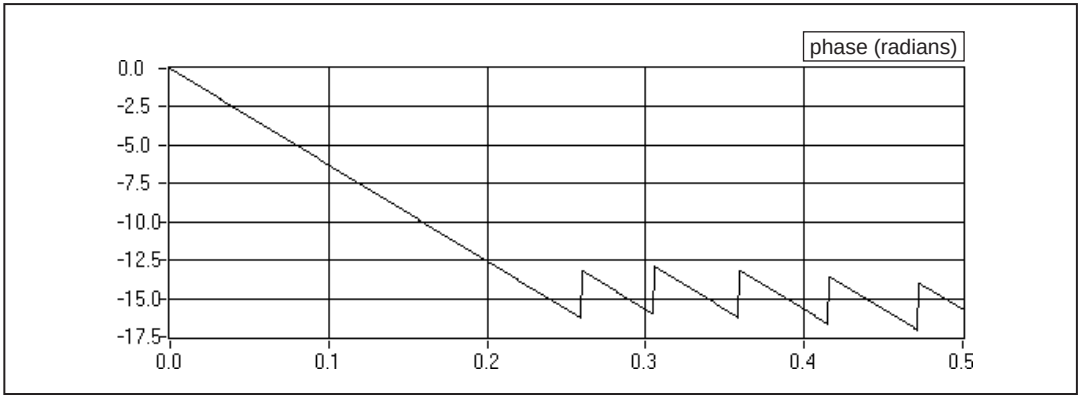
- Ils peuvent assurer une phase linéaire grâce à la symétrie des coefficients de filtre.
- Ils sont toujours stables.
- Vous pouvez exécuter la fonction de filtrage en utilisant la convolution et, comme telle, associer généralement un retard à la séquence de sortie

$$\text{retard} = \frac{n-1}{2}$$

où *n* est le nombre de coefficients de filtre RIF.

Les graphes suivants représentent une amplitude et une réponse en phase typiques des filtres RIF en fonction de la fréquence normalisée.





Les discontinuités de la réponse en phase résultent des discontinuités introduites lorsque vous calculez la réponse d'amplitude en utilisant la

valeur absolue. Remarquez que les discontinuités de la phase sont de l'ordre de *pi*. La phase est cependant clairement linéaire. Consultez l'Annexe A, *Références d'analyse*, pour plus d'informations à ce sujet.

Vous pouvez concevoir des filtres RIF en faisant l'approximation de la réponse en fréquence désirée spécifiée d'un système temporel discret. Les techniques les plus courantes donnent une approximation de la réponse d'amplitude désirée tout en maintenant une réponse en phase linéaire.

Conception de filtres RIF par fenêtrage

La méthode la plus simple pour concevoir des filtres RIF à phase linéaire est le *fenêtrage*. Pour concevoir un filtre RIF par fenêtrage, commencez par une réponse en fréquence idéale, calculez sa réponse impulsionnelle, puis tronquez la réponse impulsionnelle pour obtenir un nombre fini de coefficients. Pour répondre à la contrainte de phase linéaire, maintenez la symétrie par rapport au point central des coefficients. La troncature de la réponse impulsionnelle idéale mène à l'effet appelé "phénomène de Gibbs" : il s'agit d'un comportement oscillatoire proche des transitions abruptes (fréquences de coupure) dans la réponse en fréquence des filtres RIF.

Vous pouvez réduire les effets du phénomène de Gibbs en lissant la troncature de la réponse impulsionnelle idéale grâce à une fonction de lissage de fenêtre. En réduisant les coefficients RIF à chaque extrémité, vous pouvez diminuer la hauteur des lobes latéraux dans la réponse en fréquence. Cette méthode présente toutefois un inconvénient, car le lobe principal s'élargit, créant ainsi une région de transition plus large au niveau des fréquences de coupure. La sélection d'une fonction de fenêtrage est ainsi similaire au choix entre les filtres RII de Chebyshev et de Butterworth, dans le sens où c'est un compromis entre les niveaux des lobes latéraux près des fréquences de coupure et la largeur de la région de transition.

La conception des filtres RIF par fenêtrage est simple et requiert peu de calculs : elle représente la méthode la plus rapide pour concevoir des filtres RIF. Toutefois, il ne s'agit pas nécessairement de la meilleure méthode pour concevoir des filtres RIF.

Conception de filtres RIF optimum grâce à l'algorithme de Parks-McClellan

L'algorithme de Parks-McClellan offre une technique de conception de filtre RIF optimum, afin de concevoir le meilleur filtre possible pour un nombre de coefficients donné. Une telle conception réduit les effets indésirables pouvant survenir au niveau des fréquences de coupure. Elle offre également un meilleur contrôle sur les erreurs d'approximation dans différentes bandes de fréquence : contrôle impossible avec la méthode de fenêtrage.

L'utilisation de l'algorithme de Parks-McClellan pour concevoir des filtres RIF nécessite de nombreux calculs. Cette méthode produit toutefois des filtres RIF optimum grâce à des techniques itératives très longues.

Conception de filtres bande étroite RIF

Avec les techniques classiques de conception de filtres RIF dotés de bandes passantes particulièrement étroites, les longueurs de filtre qui en résultent peuvent s'avérer très importantes. Or il faut souvent passer beaucoup de temps pour concevoir et implémenter des filtres RIF longs, avec un risque plus élevé d'inexactitude numérique. Dans certains cas, les techniques de conception de filtre classiques, comme l'algorithme de Parks-McClellan, risquent de faire entièrement échouer la conception.

Vous pouvez utiliser un algorithme très efficace, appelé la technique de conception de filtre IFIR (Interpolated Finite Impulse Response - Réponse impulsionnelle finie interpolée) pour concevoir des filtres bande étroite RIF. Cette technique produit des filtres bande étroite qui nécessitent moins de coefficients (et donc moins de calculs) que les filtres conçus par l'application directe de l'algorithme de Parks-McClellan. LabVIEW utilise également cette technique pour produire des filtres large bande et passe-bas (fréquence de coupure proche de celle de Nyquist) et des filtres passe-haut (fréquence de coupure proche de zéro). Pour plus d'informations sur la conception des filtres RIF, consultez le livre intitulé *Multirate Systems et Filter Banks* de P.P. Vaidyanathan, ou le document sur les filtres à réponse impulsionnelle finie interpolée de Neuvo et al., indiqué à l'Annexe A, *Références d'analyse*, de ce manuel.

Filtres RIF fenêtrés

Utilisez le paramètre **type de filtre** des VIs RIF pour sélectionner le type de filtre RIF fenêtré souhaité : passe-bas, passe-haut, passe-bande ou coupe-bande. La liste suivante présente les deux VIs RIF apparentés :

- VI “Coefficients fenêtrés RIF” (FIR Windowed Coefficients.vi) : génère les coefficients fenêtrés (ou non fenêtrés).
- VI “Filtres fenêtrés RIF” (FIR Windowed Filters.vi) : filtre l’entrée en utilisant les coefficients fenêtrés (ou non fenêtrés).

Filtres RIF optimum

Vous pouvez utiliser l’algorithme de Parks-McClellan pour concevoir des coefficients de filtre RIF optimum à phase linéaire, puisque le filtre résultant correspond de façon optimale aux spécifications du filtre pour un nombre de coefficients donné. Le VI de Parks-McClellan prend comme entrée un tableau de description de bande, chacun contenant des informations décrivant la réponse souhaitée pour la bande donnée. Le VI retourne les coefficients RIF avec l’ondulation calculée, qui est une mesure de la déviation du filtre résultant des spécifications du filtre idéal.

Quatre VIs utilisent le VI de Parks-McClellan afin d’implémenter des filtres pour lesquels le niveau d’ondulation de la bande passante et de la bande d’arrêt sont égaux : passe-bas équi-ondulation, passe-haut équi-ondulation, passe-bande équi-ondulation et coupe-bande équi-ondulation.

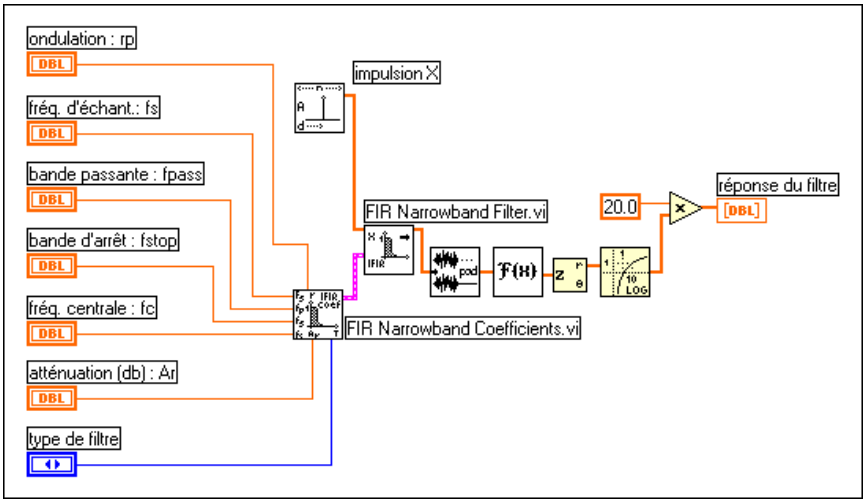
Filtres bande étroite RIF

Vous pouvez concevoir des filtres bande étroite RIF en utilisant le VI “Coefficients bande étroite RIF” (FIR Narrowband Coefficients.vi), puis implémenter le filtrage en utilisant le VI “Filtre bande étroite RIF” (FIR Narrowband Filter.vi). La conception et l’implémentation sont des opérations séparées, car de nombreux filtres bande étroite ont de longues durées de conception, mais un processus de filtrage réel très rapide et efficace. Gardez ceci à l’esprit lorsque vous créez vos diagrammes de filtrage bande étroite.

Les paramètres requis pour la spécification de filtre bande étroite sont : le type de filtre, la fréquence d’échantillonnage, les fréquences de la bande passante et de la bande d’arrêt, l’ondulation bande passante (échelle linéaire) et l’atténuation de bande d’arrêt (décibels). Pour des filtres passe-bande et coupe-bande, les fréquences de bande passante et de bande d’arrêt font référence aux largeurs de bande, et vous devez préciser un paramètre de fréquence centrale supplémentaire. Vous pouvez également concevoir

des filtres passe-bas large bande (fréquence de coupure proche de celle de Nyquist) et des filtres passe-haut large bande (fréquence de coupure proche de zéro) en utilisant les VIs de filtre bande étroite.

L'illustration suivante vous indique comment utiliser le VI "Coefficients bande étroite RIF" (FIR Narrowband Coefficients.vi) et le VI "Filtre bande étroite RIF" (FIR Narrowband Filter.vi) pour évaluer la réponse d'un filtre bande étroite à une impulsion.



Filtres non linéaires

Les fenêtres de lissage, les filtres RII et les filtres RIF sont linéaires car ils satisfont aux principes de superposition et de proportion

$$L\{ax(t) + by(t)\} = aL\{x(t)\} + bL\{y(t)\},$$

où a et b sont des constantes, *x(t)* et *y(t)* sont des signaux, L{•} est une opération de filtrage linéaire, et où leurs entrées et sorties sont liées par l'opération de convolution.

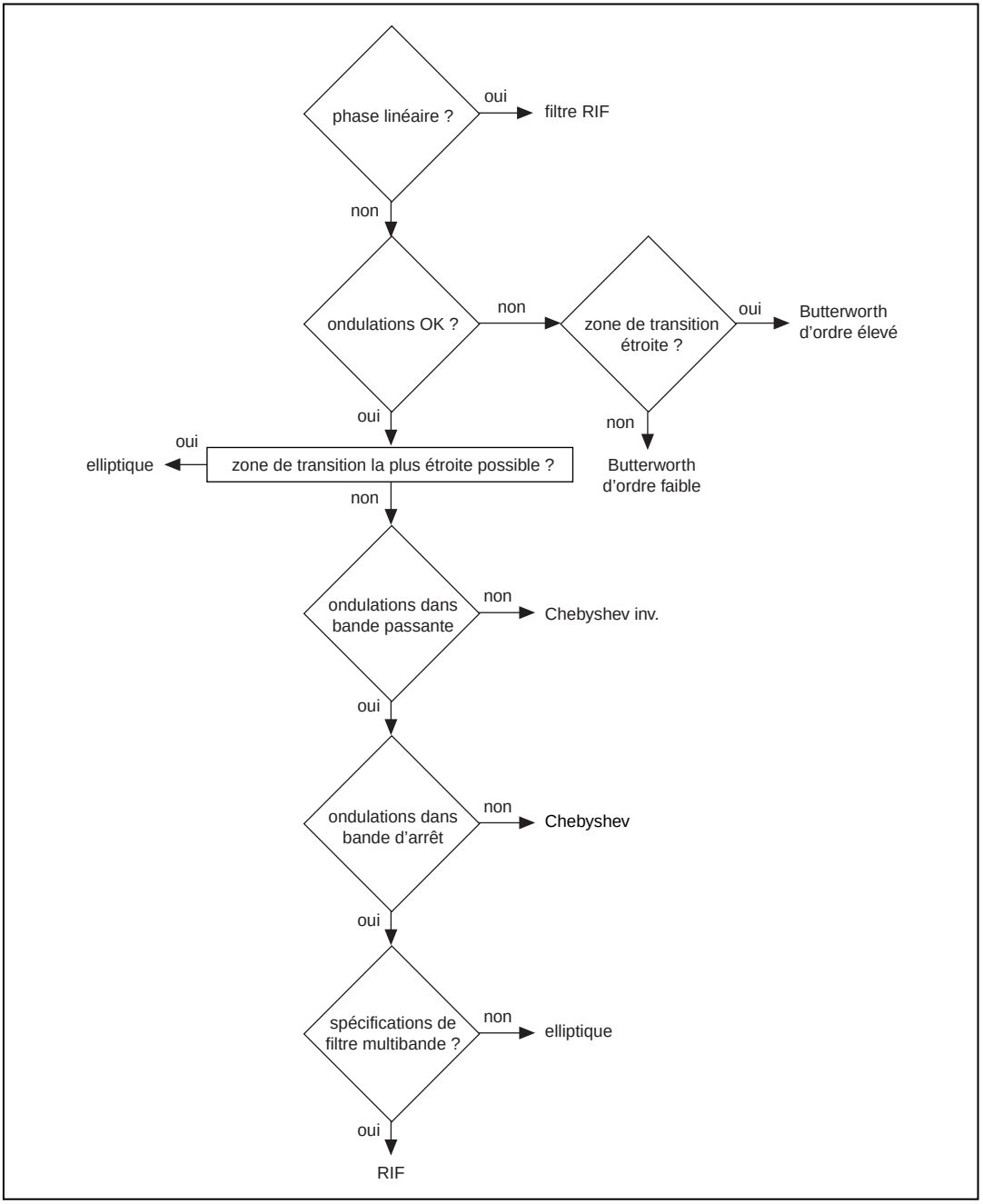
Un filtre non linéaire ne satisfait pas aux conditions précédentes et vous ne pouvez pas obtenir ses signaux de sortie par l'opération de convolution, car un ensemble de coefficients ne peut pas caractériser la réponse impulsionnelle du filtre. Les filtres non linéaires ont des caractéristiques de filtrage spécifiques, difficiles à obtenir en utilisant des techniques linéaires. Le filtre médian est un filtre non linéaire qui combine les caractéristiques

des filtres passe-bas (pour éliminer le bruit des hautes fréquences) et celles des hautes fréquences (pour détecter les arêtes).

Comment choisir le type de filtre à utiliser ?

Maintenant que vous avez appris les différents types de filtres et leurs caractéristiques, la question est de savoir quelle conception de filtre convient le mieux à votre application. Voici quelques-uns des facteurs usuels affectant le choix d'un filtre convenable : si vous avez besoin d'une phase linéaire, si vous pouvez tolérer des ondulations et s'il faut une bande de transition étroite. L'organigramme suivant donne les grandes lignes pour sélectionner le filtre adéquat.

Gardez à l'esprit qu'en pratique, vous devrez sans doute essayer plusieurs options avant de trouver la meilleure.

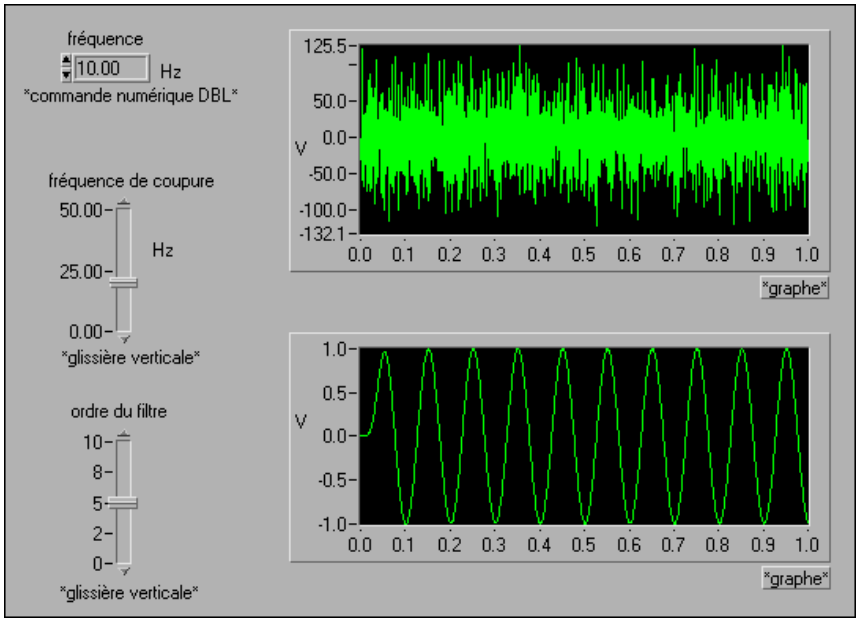


Exercice 16-3. Extraire un signal sinusoïdal

Votre objectif est de filtrer des échantillons de données qui contiennent un bruit de haute fréquence et un signal sinusoïdal.

Dans cet exercice, vous combinez un signal sinusoïdal généré par le VI “**Motif sinus”** (**Sine Pattern**.vi) avec un bruit de haute fréquence. (Le bruit de haute fréquence est obtenu en filtrant un bruit blanc uniforme passe-haut avec un filtre de Butterworth). Le signal combiné est ensuite filtré en passe-bas par un autre filtre de Butterworth afin d’extraire le signal sinusoïdal.

Face-avant



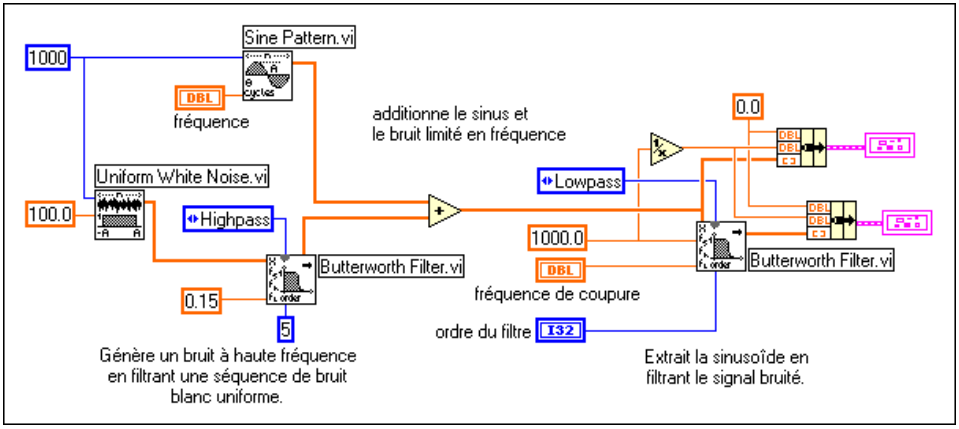
1. Ouvrez un nouveau VI et construisez la face-avant comme indiqué ci-dessus.
   a. Sélectionnez une commande numérique dans la palette **Numérique»Commandes** et nommez-la *Fréquence*.
   b. Sélectionnez une **glissière verticale** dans la palette **Numérique»Commandes** et nommez-la *Fréquence de coupure*.

c. Sélectionnez une autre **glissière verticale** dans la palette **Numérique»Commandes** et nommez-la *Ordre du filtre*.

d. Sélectionnez un g**raphe oscilloscopique** dans la palette **Numérique»Graphe** pour afficher le signal bruité, puis un autre g**raphe oscilloscopique** pour afficher le signal d'origine.

Diagramme

2. Construisez le diagramme comme indiqué ci-dessous.



Le VI "Motif sinus" (**Sine Pattern**.vi) (palette **Fonctions»Analyse»Génération de signal**) génère un signal sinusoïdal à la fréquence désirée.

Le VI "Bruit blanc uniforme" (Uniform White Noise.vi) (palette **Fonctions»Analyse»Génération de signal**) génère un bruit blanc uniforme qui est ajouté au signal sinusoïdal.

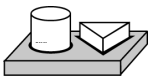
Le VI "Filtre de Butterworth" (**Butterworth Filte**r.vi) (palette **Fonctions»Analyse»Filtres**) filtre le bruit en passe-haut.

Remarquez que vous générez 10 cycles du signal sinusoïdal et qu'il y a 1000 échantillons. De plus, la fréquence d'échantillonnage du VI **"Filtre de Butterworth"** (**Butterworth Filte**r.vi) sur la droite est spécifiée à 1000 Hz. Vous générez donc efficacement un signal de 10 Hz.

3. Enregistrez le VI sous le nom `Extract the Sine Wave.vi` dans le répertoire `LabVIEW\Activity`.

4. Revenez sur la face-avant. Sélectionnez une **Fréquence** de 10 Hz, une **Fréquence de coupure** de 25 Hz et un **Ordre de filtre** de 5. Exécutez le VI.

5. Réduisez l'**Ordre de filtre** à 4, 3 et 2, puis observez la différence dans le signal bruité. Expliquez ce qui se passe lorsque vous diminuez l'ordre du filtre.
6. Lorsque vous avez terminé, enregistrez le VI sous le nom `Extract the Sine Wave.vi` dans la bibliothèque `Dig.filt.llb`.
7. Fermez le VI.



Fin de l'exercice 16-3.

En résumé

Vous venez d'apprendre, d'après les caractéristiques de la réponse en fréquence, que les filtres réels diffèrent en pratique des filtres idéaux. Pour des filtres réels, le gain n'est pas toujours égal à 1 dans la bande passante, l'atténuation de la bande d'arrêt n'est pas toujours moins l'infini, et il existe une région de transition de largeur finie. La largeur de la région de transition est liée à l'ordre du filtre : elle décroît quand l'ordre croît.

Vous avez également appris les bases concernant les filtres numériques RIF et RII. La sortie des filtres RIF ne dépend que des valeurs d'entrée présentes et passées, tandis que la sortie des filtres RII dépend à la fois des valeurs d'entrée (présentes et passées) et des valeurs de sortie (passées). Vous avez vu la réponse en fréquence de différentes conceptions de filtres RII et vous avez classé ces réponses en fonction de la présence d'ondulations dans la bande passante et/ou la bande d'arrêt. A cause de la dépendance de sa sortie sur les sorties passées, un transitoire apparaît à la sortie d'un filtre RII chaque fois que le VI est appelé. Ce transitoire peut être éliminé après le premier appel au VI si sa commande **init/cont** est mise sur la valeur Vrai (TRUE).

17

Ajustement de courbe

Ce chapitre décrit comment extraire des informations d'un ensemble de données afin d'en obtenir une description fonctionnelle. Pour des exemples d'utilisation des VIs de régression, consultez la bibliothèque `examples\analysis\regressn.llb`.

Introduction à l'ajustement de courbe

L'analyse d'ajustement de courbe est une technique utilisée pour extraire un ensemble de paramètres ou de coefficients de courbe d'un ensemble de données, afin d'obtenir une description fonctionnelle de l'ensemble des données. L'algorithme qui ajuste une courbe à un ensemble de données particulier est connu sous le nom de "méthode des moindres carrés". Cette méthode est traitée dans la plupart des manuels d'initiation aux probabilités et aux statistiques. L'erreur est définie sous la forme

$$e(a) = [f(x,a) - y(x)]^2 \qquad (17\text{-}1)$$

où $e(a)$ est l'erreur, $y(x)$ est l'ensemble des données observé, $f(x,a)$ est la description fonctionnelle de l'ensemble des données et a est l'ensemble des coefficients de courbe décrivant le mieux celle-ci.

Par exemple, soit $a = \{a_0, a_1\}$. Une ligne peut alors être décrite par la fonction

$$f(x,a) = a_0 + a_1 x$$

L'algorithme des moindres carrés détermine l'inconnue a en résolvant le système

$$\frac{\partial}{\partial a} e(a) = 0 \qquad (17\text{-}2)$$

Pour résoudre ce système, définissez et résolvez le système jacobien obtenu en développant l'équation 17-2. Après avoir déterminé l'inconnue a du système, vous pouvez obtenir une estimation de l'ensemble de données observé pour toute valeur de x en utilisant la description fonctionnelle $f(x, a)$.

Dans LabVIEW, les VIs d'ajustement de courbe définissent et résolvent automatiquement le système jacobien, puis retournent l'ensemble de coefficients qui décrit le mieux votre ensemble de données. Vous pouvez ainsi vous concentrer sur la description fonctionnelle de vos données et ne pas vous soucier de la résolution de l'équation 17-2.

Deux séquences d'entrée, Valeurs X et Valeurs Y, représentent l'ensemble des données $y(x)$. Un échantillon ou point de l'ensemble de données est

$$(x_i, y_i),$$

où x_i est le $i^{ème}$ élément des Valeurs X de la séquence et y_i, le $i^{ème}$ élément des Valeurs Y de la séquence.

En général, pour chaque type d'ajustement de courbe prédéfini, il existe deux types de VIs, sauf indication contraire. Le premier type ne retourne que les coefficients, de sorte que vous pouvez manipuler plus en profondeur les données. L'autre type retourne les coefficients, la courbe attendue ou ajustée correspondante, et l'erreur quadratique moyenne (MSE). Comme il s'agit d'un système discret, le VI calcule l'erreur quadratique moyenne, qui est une mesure relative des erreurs résiduelles entre les valeurs de courbe prévues et les valeurs réelles observées, en utilisant la formule

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(f_i - y_i)^2 \qquad (17\text{-}3)$$

où f est la séquence représentant les valeurs ajustées, y est la séquence représentant les valeurs observées et n, le nombre de points d'échantillons observé.

La bibliothèque d'analyse offre des algorithmes d'ajustement de courbe linéaires et non linéaires. Les différents types d'ajustement de courbe dans LabVIEW sont décrits ci-dessous :

- *Ajustement linéaire* : ajuste des données expérimentales en une ligne droite de la forme $y = mx + c$.

$$y[i]=a_0+a_1*x[i]$$

- *Ajustement exponentiel* : ajuste les données en une courbe exponentielle de la forme $y = aexp(bx)$

$$y[i] = a_0*\exp(a_1*x[i])$$

- *Ajustement polynomial général* : ajuste les données en une courbe polynomiale de la forme

$$y = a + bx + cx\text{^}2 + ...$$
$$y[i] = a_0 + a_1 * x[i] + a_2 * x[i]^2 ...$$

- *Ajustement linéaire général* **:** ajuste les données en

$$y[i] = a_0 + a_1 * f_1(x[i]) + a_2 * f_2(x[i]) + ...$$

où y[i] est une combinaison linéaire des paramètres a_0, a_1, a_2... L'ajustement linéaire général présente également des algorithmes sélectionnables pour obtenir une meilleure précision et une plus grande exactitude. Par exemple, $y = a_0 + a_1 * \sin(x)$ est un ajustement linéaire parce que *y* est en relation linéaire avec les paramètres a_0 et a_1. Les ajustements polynomiaux sont toujours des ajustements linéaires pour cette même raison. Cependant, des algorithmes spéciaux peuvent être conçus pour forcer l'ajustement polynomial à accélérer le processus d'ajustement et pour améliorer l'exactitude.

- *Ajustement de Levenberg-Marquardt non linéaire* : ajuste les données en

$$y[i] = f(x[i], a_0, a_1, a_2...)$$

où a_0, a_1, a_2... sont les paramètres. Cette méthode est la plus générale et ne requiert pas une relation linéaire de *y* avec a_0, a_1, a_2.... Elle peut être utilisée pour ajuster des courbes linéaires ou non linéaires, mais elle est presque toujours utilisée pour ajuster une courbe non linéaire, car la méthode d'ajustement linéaire générale est mieux adaptée à l'ajustement de courbe linéaire. La méthode de Levenberg-Marquardt ne garantissant pas toujours un résultat correct, il est absolument nécessaire de vérifier les résultats.

Applications d’ajustement de courbe

Il y a de nombreuses applications pratiques d’ajustement de courbe, notamment :

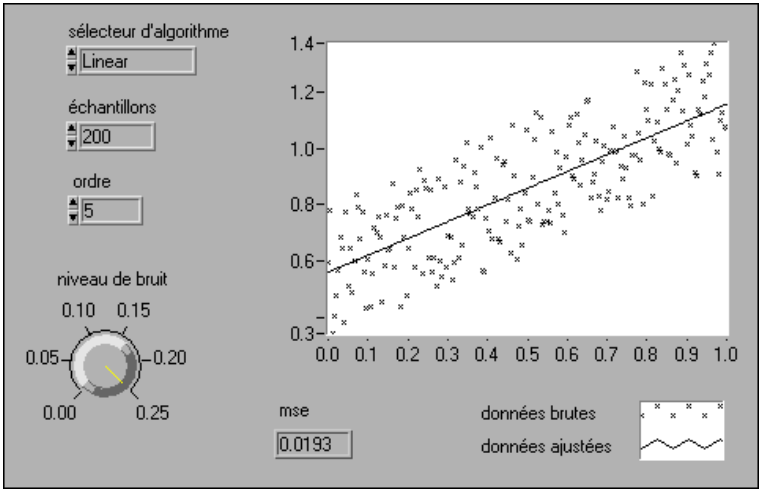
- L’élimination du bruit généré par les mesures.
- Le remplissage des points manquants (par exemple, si une ou plusieurs mesures manquent ou ont été enregistrées de façon incorrecte).
- L’interpolation (estimation des données entre des points ; par exemple, lorsque la durée entre deux mesures n’est pas suffisamment courte).
- L’extrapolation (estimation des données au-delà des points ; par exemple, lorsque vous recherchez la valeur de données se trouvant avant ou après la prise des mesures).
- La dérivation de données numériques (par exemple, si vous devez trouver la dérivée des points. Les données discrètes peuvent être modélisées par un polynôme, et l’équation polynomiale qui en résulte peut être dérivée).
- L’intégration de données numériques (par exemple, pour trouver la zone sous une courbe lorsque vous n’avez que les points discrets de la courbe).
- L’obtention de la trajectoire d’un objet, en se basant sur des mesures discrètes de sa vitesse (dérivée première) ou de son accélération (dérivée seconde).

Exercice 17-1. Utiliser les VIs d'ajustement de courbe

Votre objectif est d'utiliser et de comparer les VIs d'ajustement de courbe linéaire, exponentiel et polynomial afin d'obtenir l'ensemble des coefficients des moindres carrés représentant le mieux un ensemble de points.

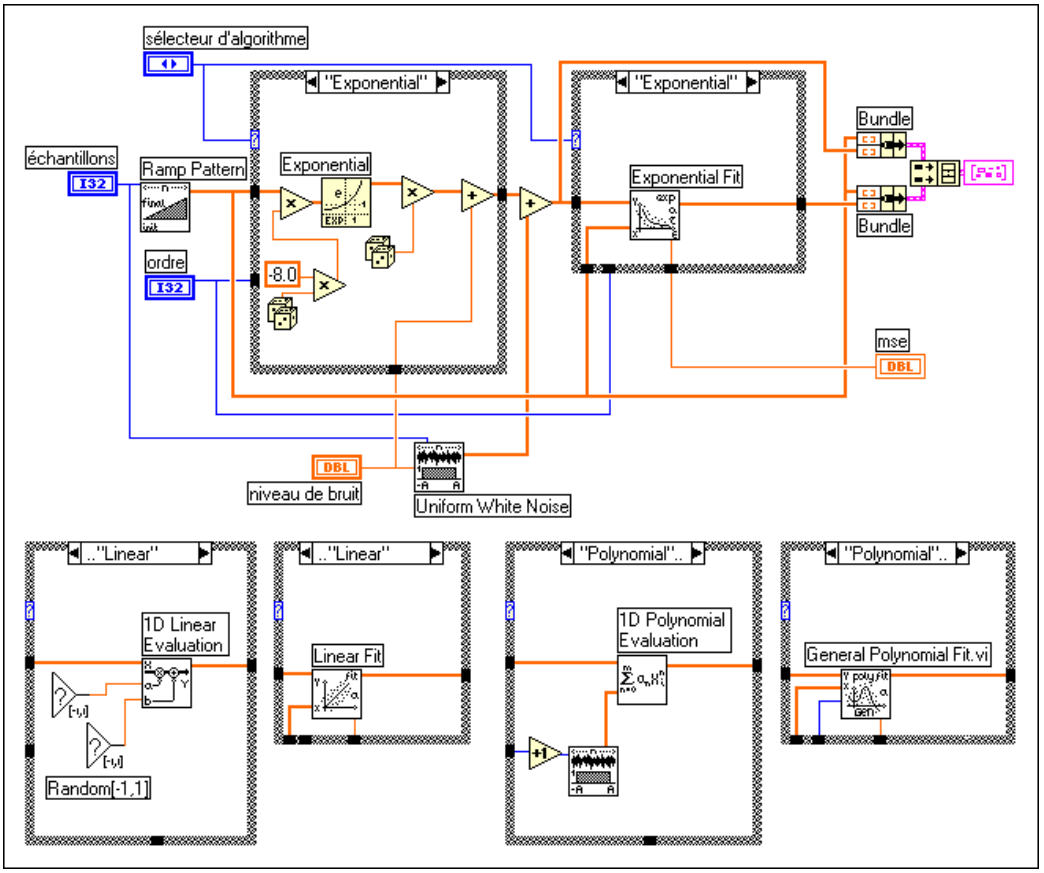
Face-avant

1. Ouvrez le VI "Démo régressions" (Regressions Demo.vi) de la bibliothèque `regressn.llb`. La face-avant et le diagramme sont déjà construits.



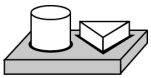
Ce VI génère des échantillons de données "bruités" qui sont approximativement linéaires, exponentiels ou polynomiaux. Le VI utilise ensuite les VIs d'ajustement de courbe d'analyse correspondants pour déterminer les paramètres de la courbe qui correspondent le mieux à ces points. (A ce stade, ne vous souciez pas de la façon dont sont générés les échantillons de données du bruit). Vous pouvez contrôler l'amplitude du bruit grâce à la commande **Niveau de bruit** de la face-avant.

Diagramme



2. Sélectionnez *Linéaire* dans la commande **Sélecteur d'algorithme**, puis mettez la commande **Niveau de bruit** sur 0,1 environ. Exécutez le VI. Observez l'étendue des points et la courbe ajustée (ligne droite).
3. Faites des essais avec différentes valeurs données aux commandes **Ordre** et **Niveau de bruit**. Que remarquez-vous ? Dans quel sens l'erreur quadratique moyenne varie-t-elle ?
4. Faites passer la commande **Sélecteur d'algorithme** sur *Exponentiel* et exécutez le VI. Faites des essais avec différentes valeurs des commandes **Ordre** et **Niveau de bruit**. Que remarquez-vous ?
5. Faites passer la commande **Sélecteur d'algorithme** sur *Polynomial* et exécutez le VI. Faites des essais avec différentes valeurs des commandes **Ordre** et **Niveau de bruit**. Que remarquez-vous ?

6. En particulier, lorsque la commande **Sélecteur d'algorithme** est mise sur *Polynomial*, faites passer la commande **Ordre** sur 0 et exécutez le VI. Optez ensuite pour la valeur 1 et exécutez le VI. Expliquez vos observations.
7. D'après vos observations relatives aux étapes 2, 3, 4 et 5, avec quel algorithme (linéaire, exponentiel, polynomial) la commande **Ordre** s'avère-t-elle la plus efficace ? Pourquoi ?
8. Fermez le VI. N'enregistrez pas les changements.



Fin de l'exercice 17-1.

Théorie de l'ajustement linéaire général (moindre carré)

Le problème de l'ajustement linéaire général (moindre carré) peut être décrit comme suit.

Etant donné un ensemble de données d'observation, recherchons un ensemble de coefficients correspondant au "modèle" linéaire.

$$y_i = b_o x_{i0} + \ldots + b_{k-1} x_{ik-1}$$

$$= \sum_{j=0}^{k-1} b_j x_{ij} \qquad \text{i=0, 1, ... , n–1,} \tag{17-4}$$

où *B* est l'ensemble de **Coefficients**, *n* est le nombre d'éléments dans les **Valeurs Y** et le nombre de rangées de **H**, et où *k* est le nombre de **Coefficients**.

x_{ij} représente vos données d'observation contenues dans **H**.

$$H = \begin{bmatrix} x_{00} & x_{01}\cdots & x_{0k-1} \\ x_{10} & x_{11}\cdots & x_{1k-1} \\ . & & \\ . & & \\ . & & \\ . & & \\ x_{n-10} & x_{n-12}\cdots & x_{n-1k-1} \end{bmatrix}$$

L'équation 17-4 peut également s'écrire sous la forme $Y = HB$.

Ceci est un modèle de régression linéaire multiple, qui utilise plusieurs variables $x_{i0}, x_{i1}, \ldots, x_{ik-1}$ pour décrire une seule variable y_i. A l'opposé, les VIs d'ajustement linéaire, exponentiel et polynomial sont tous basés sur une seule variable explicative, qui utilise une variable pour en décrire une autre.

Dans la plupart des cas, il y a davantage de données d'observation que de coefficients. Les équations 17-4 ne possèdent peut-être pas de solution. Le problème d'ajustement correspond alors à la recherche du coefficient B qui minimise la différence entre les données observées y_i et les valeurs prévues :

$$z_i = \sum_{j=0}^{k-1} b_j x_{ij}$$

Ce VI utilise la méthode plane du moindre chi carré pour obtenir les coefficients de l'équation 17-4, c'est-à-dire pour trouver la solution, B, qui minimise la quantité :

$$\chi^2 = \sum_{i=0}^{n-1}\left(\frac{y_i - z_i}{\sigma_i}\right)^2 = \sum_{i=0}^{n-1}\left(\frac{y_i - \sum_{j=0}^{k-1} b_j x_{ij}}{\sigma_i}\right)^2 = |H_0 B - Y_0|^2 \qquad (17\text{-}5)$$

où

$$h_{oij} = \frac{x_{ij}}{\sigma_i},\ y_{oi} = \frac{y_i}{\sigma_i},\ i=0, 1, \ldots, n-1;\ j=0, 1, \ldots, k-1.$$

Dans cette équation, σ_i est l'**Ecart-type**. Si les erreurs de mesure sont indépendantes et distribuées normalement avec un écart-type constant $\sigma_i = \sigma$, l'équation précédente représente également l'estimation des moindres carrés.

Il y a différentes façons de minimiser χ^2. Une façon consiste à rendre les dérivées partielles de χ^2 par rapport à $b_0, b_1, \ldots, b_{k-1}$ égales à zéro.

$$\begin{cases} \dfrac{\partial \chi^2}{\partial b_0} = 0 \\ \dfrac{\partial \chi^2}{\partial b_1} = 0 \\ \vdots \\ \dfrac{\partial \chi^2}{\partial b_{k-1}} = 0 \end{cases}$$

On peut dériver les équations précédentes pour obtenir :

$$H_0^T H_0 B = H_0^T Y \qquad (17\text{-}6)$$

où H_0^T est la transposée de $\mathbf{H}_0$.

Les équations 17-6 sont également appelées équations normales des problèmes de moindres carrés. Vous pouvez les résoudre en utilisant les algorithmes de factorisation LU ou de Cholesky, mais la solution des équations normales est sujette aux erreurs d'arrondi.

Une autre façon (recommandée) de minimiser χ^2 est de rechercher par la méthode des moindres carrés la solution des équations

$$H_0 B = Y_0.$$

Vous pouvez utiliser une factorisation QR ou SVD pour trouver la solution *B*. Pour la factorisation QR, vous pouvez choisir les méthodes

Householder, Givens et Givens2 (cette dernière étant également appelée la méthode de Givens rapide).

Différents algorithmes peuvent vous donner des précisions différentes, et dans certains cas, si un algorithme ne peut pas résoudre l'équation, un autre algorithme le pourra peut-être. Vous pouvez essayer différents algorithmes pour trouver celui qui convient le mieux à vos données d'observation.

La matrice des covariances est calculée grâce à la formule

$$C = (H_0^T H_0)^{-1}$$

Le meilleur ajustement Z est donné par

$$z_i = \sum_{j=0}^{k-1} b_j x_{ij}$$

L'erreur quadratique moyenne (mse) est obtenue grâce à la formule suivante :

$$\boldsymbol{mse} = \frac{1}{n} \sum_{i=0}^{n-1} \left(\frac{y_i - z_i}{\sigma_i} \right)^2$$

L'ajustement polynomial avec une seule variable explicative peut être considéré comme un cas particulier de régression multiple. Si les ensembles de données d'observation sont $\{x_i, y_i\}$, où $i = 0, 1, \ldots, n-1$, le modèle d'ajustement polynomial est

$$y_i = \sum_{j=0}^{k-1} b_j x_i^j = b_0 + b_1 x_i + b_2 x_i^2 + \ldots + b_{k-1} x_i^{k-1} \qquad (17\text{-}7)$$

$$i = 0, 1, 2, \ldots, n-1$$

En comparant les équations 17-4 et 17-7, on remarque que $x_{ij} = x_i^j$. En d'autres termes,

$$x_{i0} = x_i^0, \quad x_{i1} = x_i, \quad x_{i2} = x_i^2, \ldots \quad x_{ik-1} = x_i^{k-1}$$
$$= 1$$

Dans ce cas, vous pouvez construire **H** comme indiqué ci-après :

$$H = \begin{Bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{k-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{k-1} \\ . & & & & \\ . & & & & \\ . & & & & \\ . & & & & \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{k-1} \end{Bmatrix}$$

Au lieu d'utiliser $x_{ij} = x_j^i$, vous pouvez également choisir une autre formule de fonction pour ajuster les ensembles de données $\{x_i, y_i\}$. En général, vous pouvez sélectionner $x_{ij} = f_j(x_i)$. Dans le cas présent, $f_j(x_i)$ est le modèle de fonction que vous choisissez pour ajuster vos données d'observation. Dans un ajustement polynomial, $f_j(x_i) = x_i^j$.

En général, vous pouvez construire **H** comme indiqué ci-après :

$$H = \begin{Bmatrix} f_0(x_0) & f_1(x_0) & f_2(x_0) & \dots & f_{k-1}(x_0) \\ f_0(x_1) & f_1(x_1) & f_2(x_1) & \dots & f_{k-1}(x_1) \\ . & & & & \\ . & & & & \\ . & & & & \\ . & & & & \\ f_0(x_{n-1}) & f_1(x_{n-1}) & f_2(x_{n-1}) & \dots & f_{k-1}(x_{n-1}) \end{Bmatrix}$$

Votre modèle d'ajustement est :

$$y_i = b_0 f_0(x) + b_1 f_1(x) + \dots + b_{k-1} f_{k-1}(x)$$

Comment utiliser le VI “Ajustement linéaire général (moindre carré)” (General LS Linear Fit.vi)

Le VI “Ajustement linéaire” (Linear Fit.vi) calcule les coefficients a_0 et a_1 qui correspondent le mieux aux données expérimentales (x[i] et y[i]) sur un modèle de ligne droite donné par

$$y[i] = a_0 + a_1*x[i]$$

Dans ce cas, y[i] est une combinaison linéaire des coefficients a_0 et a_1. Vous pouvez étendre plus avant ce concept pour que le multiplicateur de a_1 soit une fonction quelconque de *x*. Par exemple :

$$y[i] = a_0 + a_1*\sin(\omega x[i])$$

ou

$$y[i] = a_0 + a_1*x[i]^2$$

ou

$$y[i] = a_0 + a_1*\cos(\omega x[i]^2)$$

où ω est la pulsation. Dans chaque cas, y[i] est une combinaison linéaire des coefficients a_0 et a_1. C’est l’idée fondamentale sous-jacente au VI “Ajustement linéaire général (moindre carré)” (General LS Linear Fit.vi), où y[i] peut représenter des combinaisons linéaires de plusieurs coefficients, chacun d’eux pouvant être multiplié par une fonction quelconque de x[i]. Vous pouvez donc utiliser ce VI pour calculer les coefficients de modèles fonctionnels pouvant être représentés comme des combinaisons linéaires des coefficients, tels que

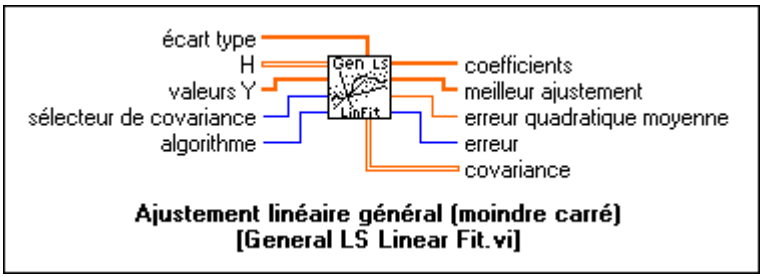
$$y = a_0 + a_1*\sin(\omega x)$$

ou

$$y = a_0 + a_1*x^2 + a2*\cos(\omega x^2)$$
$$y = a_0 + a_1*(3\sin(\omega x)) + a_2*x^3 + a_3 / x + ...$$

Dans chaque cas, remarquez que *y* est une fonction *linéaire* des coefficients (mais *y* peut être une fonction non linéaire de *x*).

Vous allez maintenant apprendre à utiliser le **VI “Ajustement linéaire général (moindre carré)”** (General LS Linear Fit.vi) pour déterminer le meilleur ajustement linéaire d’un ensemble de points. Les entrées et sorties du VI “Ajustement linéaire général (moindre carré)” (General LS Linear Fit.vi) sont représentées ci-dessous.



Les données que vous recueillez (x[i] et y[i]) sont à fournir aux entrées **H** et **Valeurs Y**. La sortie **Covariance** représente la matrice des covariances entre les coefficients a_k, où c_{ij} est la covariance entre a_i et a_j, et où c_{kk} est la variance de a_k. A ce stade, vous ne devez pas vous soucier des entrées **écart-type**, **sélecteur de covariance** et **algorithme**. Pour l’instant, vous n’utiliserez que leur valeur par défaut. Vous pouvez consulter la *Référence en ligne d’analyse* pour plus de détails sur ces entrées.

La matrice **H** est appelée la *Matrice d’observation* et sera expliquée ultérieurement en de plus amples détails. **Valeurs Y** est l’ensemble de points y[i] observés. Par exemple, supposons que vous ayez recueilli des échantillons (**Valeurs Y**) d’un transducteur et que vous souhaitiez déterminer les coefficients du modèle :

$$y = a_o + a_1 \sin(\omega x) + a_2 \cos(\omega x) + a_3 x^2$$

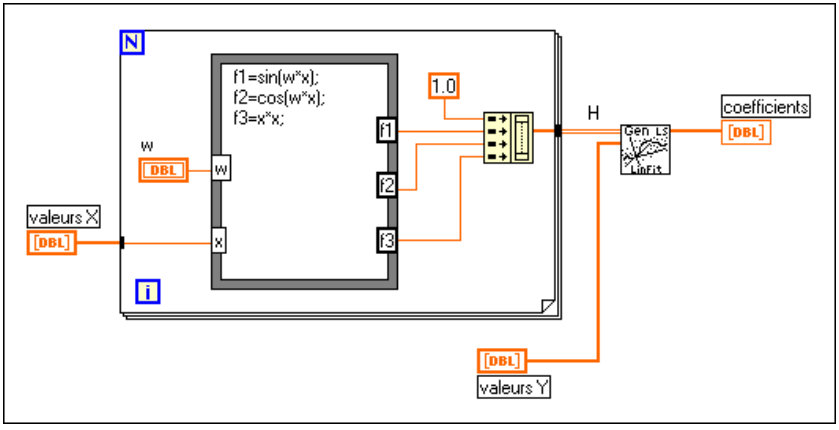
Vous voyez que le multiplicateur de chaque a_j (0 ð j ð3) est une fonction différente. Par exemple, a_0 est multiplié par 1, a_1 est multiplié par sin(ωx), a_2 est multiplié par cos(ωx), et ainsi de suite. Pour construire **H**, définissez chaque colonne de **H** comme des fonctions indépendantes évaluées pour

chaque valeur x[i] de *x*. En supposant qu'il existe 100 valeurs "*x*", **H** devient :

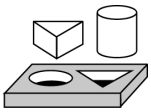
$$H = \begin{bmatrix} 1 & \sin(\omega x_0) & \cos(\omega x_0) & x_0^2 \\ 1 & \sin(\omega x_1) & \cos(\omega x_1) & x_1^2 \\ 1 & \sin(\omega x_2) & \cos(\omega x_2) & x_2^2 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \sin(\omega x_{99}) & \cos(\omega x_{99}) & x_{99}^2 \end{bmatrix}$$

S'il y a *N* points et *k* coefficients (a_0, a_1, ...a_{k-1}) à déterminer, **H** est une matrice *N-fois-k* avec *N* rangées et *k* colonnes. Le nombre de rangées de **H** est donc égal au nombre d'éléments de **Valeurs Y**, tandis que le nombre de colonnes de **H** est égal au nombre de coefficients à déterminer.

En pratique, la matrice **H** n'existe pas et doit être construite. Dans l'hypothèse où vous avez les *N* **Valeurs X** indépendantes et les **Valeurs Y** observé**es**, le diagramme suivant indique comment construire **H** et utiliser le VI "Ajustement linéaire général (moindre carré)" (General LS Linear Fit.vi).



Exercice 17-2. Utiliser le VI “Ajustement linéaire général (moindre carré)” (General LS Linear Fit.vi)



Dans cet exercice, votre objectif est d’apprendre à définir les paramètres d’entrée et à utiliser le VI “Ajustement linéaire général (moindre carré)” (General LS Linear Fit.vi).

Cet exercice permet d’apprendre comment utiliser le VI “Ajustement linéaire général (moindre carré)” (General LS Linear Fit.vi) afin d’obtenir l’ensemble des coefficients de moindres carrés *a* et les valeurs ajustées. Vous apprendrez également comment définir les paramètres d’entrée du VI.

Le but de l’exercice est de rechercher l’ensemble des coefficients de moindres carrés **a** qui représentent le mieux l’ensemble des points (x[i], y[i]). Par exemple, soit un processus physique qui génère des données en suivant la relation

$$y = 2h_0(x) + 3h_1(x) + 4h_2(x) + \text{bruit} \qquad (17\text{-}8)$$

où

$$h_0(x) = \sin(x^2)$$
$$h_1(x) = \cos(x)$$
$$h_2(x) = \frac{1}{x+1}$$

et où le *bruit* représente des valeurs aléatoires. En outre, supposons que vous ayez une idée de la forme générale de la relation entre *x* et *y*, mais que vous ne connaissiez pas la valeur des coefficients. Vous pouvez penser que la relation entre *x* et *y* est de la forme

$$y = a_0f_0(x) + a_1f_1(x) + a_2f_2(x) + a_3f_3(x) + a_4f_4(x) \qquad (17\text{-}9)$$

où

$$f_o(x) = 1,0$$

$$f_1(x) = \sin(x^2)$$

$$f_2(x) = 3\cos(x)$$

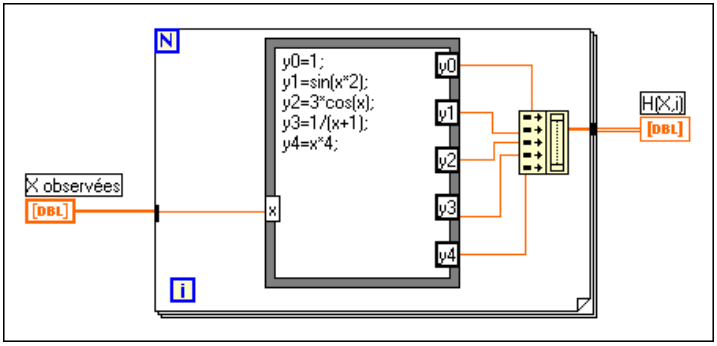
$$f_3(x) = \frac{1}{x+1}$$

$$f_4(x) = x^4$$

Les équations 17-8 et 17-9 correspondent respectivement au processus physique réel et à votre estimation du processus. Les coefficients choisis dans votre estimation peuvent s'avérer proches des valeurs réelles comme ils peuvent en être très éloignés. Votre objectif ici est de déterminer les coefficients *a* avec exactitude.

Construction de la matrice d'observation

Pour obtenir les coefficients *a*, vous devez fournir l'ensemble des points (x[i], y[i]) des tableaux **H** et **Valeurs Y** (où la matrice **H** est un tableau 2D) au VI "Ajustement linéaire général (moindre carré)" (General LS Linear Fit.vi). Les points x[i] et y[i] sont les valeurs observées dans votre exercice. Pour construire aisément la matrice **H**, vous pouvez utiliser la boîte de calcul comme indiqué dans le diagramme suivant.

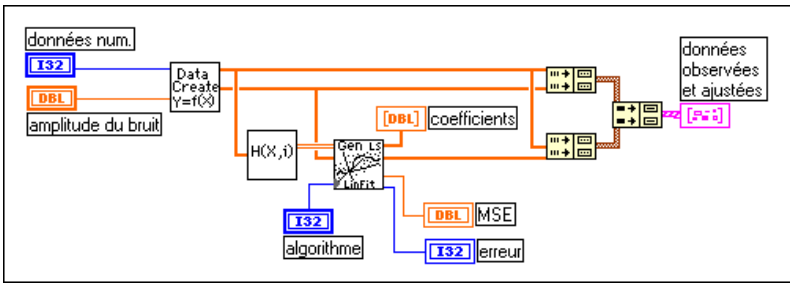


Vous pouvez éditer la boîte de calcul pour modifier, ajouter ou supprimer des fonctions. A ce stade, vous possédez toutes les entrées nécessaires pour utiliser le VI "Ajustement linéaire général (moindre carré)" (General LS Linear Fit.vi) afin de déterminer *a*. Pour obtenir l'équation (1) à partir de l'équation (2), vous devez multiplier $f_0(x)$ par 0, $f_1(x)$ par 2, $f_2(x)$ par 1, $f_3(x)$

par 4 et $f_4(x)$ par 0. En examinant les équations (1) et (2), vous pouvez remarquer que l'ensemble des coefficients prévu est

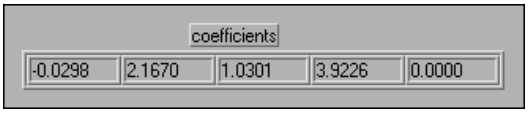
$$a = \{0{,}0 ; 2{,}0 ; 1{,}0 ; 4{,}0 ; 0{,}0\}$$

Le diagramme ci-dessous indique comment configurer le VI "Ajustement linéaire général (moindre carré)" (General LS Linear Fit.vi) pour obtenir les coefficients et un nouvel ensemble de valeurs *y*.



Le sous-VI "Créer les données" (Data Create.vi) génère les tableaux **X** et **Y**. Vous pouvez remplacer cette icône par une autre recueillant réellement les données de vos essais. L'icône libellée **H(X,i)** génère la matrice 2D **H**.

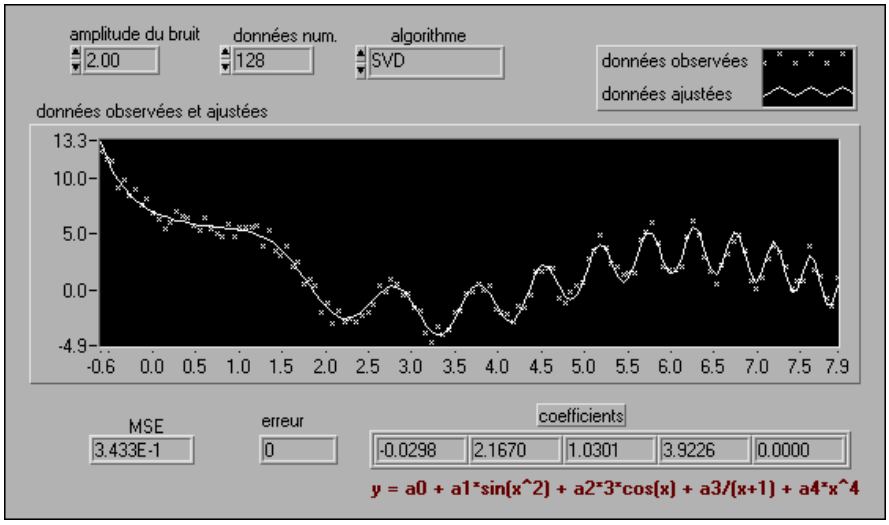
La dernière portion du diagramme se superpose sur l'original et sur les points estimés. Elle crée ainsi un enregistrement visuel de l'ajustement linéaire général (moindre carré). Le VI "Ajustement linéaire général (moindre carré)" (General LS Linear Fit.vi) exécuté avec les valeurs de **X**, **Y**, et **H** retourne l'ensemble des coefficients suivants.



L'équation qui en résulte est donc

$$y = 0{,}0298(1) + 2{,}1670\sin(x^2) + 1{,}0301(3\cos(x)) + 3{,}9226/(x+1) + 0{,}00(x^4)$$
$$= 0{,}0298 + 2{,}1670\sin(x^2) + 1{,}0301(3\cos(x)) + 3{,}9226/(x+1)$$

Le graphe suivant présente les résultats.



Vous pouvez désormais voir le VI dans lequel cet exemple particulier a été mis en application.

1. Ouvrez le VI “Exemple d’ajustement général (moindre carré)” (General LS Fit Example.vi) dans la bibliothèque `examples\analysis\regressn.llb`.
2. Examinez le diagramme et assurez-vous de comprendre tous ses éléments.
3. Observez la face-avant.

   amplitude du bruit : permet de modifier l’amplitude du bruit ajoutée aux points. Plus cette valeur est importante, plus les points sont dispersés.

   données numériques : le nombre de points que vous souhaitez générer.

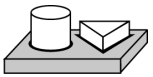
   algorithme : offre une sélection de six algorithmes différents pour obtenir l’ensemble des coefficients et les valeurs ajustées. Dans cet exemple particulier, il n’y a pas d’écart important entre les différents algorithmes. Vous pouvez sélectionner différents algorithmes à partir de la face-avant pour voir les résultats. Dans certains cas, différents algorithmes peuvent présenter des écarts importants, selon votre ensemble de données observées.

   MSE : fournit l’erreur quadratique moyenne. Plus cette erreur est petite, meilleur est l’ajustement.

erreur : donne le code d'erreur en cas d'erreur. Si le code d'erreur = 0, cela signifie qu'il n'y a pas d'erreur. Pour obtenir une liste des codes d'erreur, consultez l'Annexe A, *Codes d'erreur*, dans le *Manuel de référence des VIs et des fonctions de LabVIEW.*

coefficients : les valeurs calculées des coefficients (a_0, a_1, a_2, a_3 et a_4) du modèle.

4. Exécutez le VI en donnant des valeurs de plus en plus grandes à l'**amplitude du bruit**. Que deviennent les données observées tracées sur le graphe ? Qu'en est-il de l'erreur quadratique moyenne ?
5. Pour des valeurs fixes de l'**amplitude du bruit**, exécutez le VI en choisissant différents algorithmes à partir de la commande **algorithme**. Concluez-vous qu'un algorithme est meilleur que les autres ? Lequel vous donne la plus petite erreur quadratique moyenne ?
6. Lorsque vous avez terminé, fermez le VI. N'enregistrez pas les changements.



Fin de l'exercice 17-2.

Théorie de l'ajustement de Lev-Mar non linéaire

Ce VI détermine l'ensemble des coefficients qui minimise la quantité des chi carrés :

$$\chi^2 = \sum_{i=0}^{N-1} \left(\frac{y_i - f(x_i; a_1 \ldots a_M)}{\sigma_i} \right)^2 \qquad (17\text{-}10)$$

Dans cette équation, (x_i, y_i) sont les points d'entrée et $f(x_i;a_1...a_M) = f(X, A)$ représente la fonction non linéaire où $a_1...a_M$ sont les coefficients. Si les erreurs de mesure sont indépendantes et distribuées normalement avec un écart-type constant $\sigma_i = \sigma$, il s'agit également de l'estimation des moindres carrés.

Vous devez préciser la fonction non linéaire $f = f(X, A)$ dans la boîte de calcul située dans le diagramme du VI "Fonct. cible & dérivée non lin." (Target Fnc & Deriv NonLin.vi), qui est un sous-VI du VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi). Vous pouvez accéder au VI "Fonct. cible & dérivée non lin." (Target Fnc & Deriv NonLin.vi) en le

sélectionnant dans le menu qui apparaît lorsque vous sélectionnez **Projet» Sous-VIs appelés par ce VI**.

Ce VI permet de calculer de deux manières différentes le Jacobien (dérivées partielles des coefficients) nécessaire dans l'algorithme. Ces deux méthodes sont les suivantes :

- Calcul numérique : utilise une approximation numérique pour calculer le Jacobien.
- Calcul avec formule : utilise une formule pour calculer le Jacobien. Vous devez préciser la fonction du Jacobien $\partial f / \partial A$ dans la boîte de calcul sur le diagramme du VI "Fonct. cible & dérivée non lin." (Target Fnc & Deriv NonLin.vi), ainsi que la fonction non linéaire $f = f(X, A)$. C'est un calcul plus efficace que le calcul numérique, car le Jacobien ne subit pas d'approximation numérique dans ce cas-là.

Les tableaux d'entrées **X** et **Y** définissent l'ensemble des points d'entrée. Le VI est basé sur le fait que vous avez déjà connaissance de la relation non linéaire qui existe entre les coordonnées *x* et *y*. C'est-à-dire que $f = f(X, A)$, où l'ensemble de coefficients, *A*, est déterminé par l'algorithme de Levenberg-Marquardt.

L'utilisation efficace de cette fonction dépend parfois du degré de similitude entre vos coefficients estimés initiaux et la solution. Il est donc toujours judicieux de prendre le temps et la peine d'obtenir de bons coefficients estimés initiaux en s'aidant de toutes les ressources disponibles avant d'utiliser la fonction.

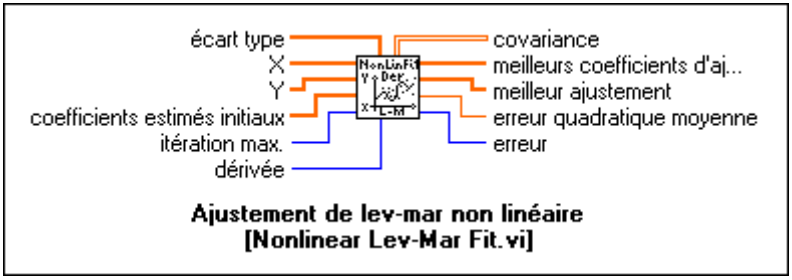
Utilisation du VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi)

Jusqu'à présent, vous avez vu les VIs qui sont utilisés lorsqu'il existe une relation linéaire entre *y* et les coefficients a_0, a_1, a_2, ... Toutefois, lorsque la relation n'est pas linéaire, vous pouvez utiliser le VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi) pour déterminer les coefficients. Ce VI utilise la méthode de Levenberg-Marquardt, très puissante, pour trouver les coefficients $\mathbf{A} = \{a_0, a_1, a_2, ..., a_k\}$ de la relation non linéaire entre **A** et y[i]. Le VI suppose que vous avez déjà connaissance de la relation non linéaire qui existe entre les coordonnées *x* et *y*.

Premièrement, vous devez préciser la fonction non linéaire dans la boîte de calcul située dans le diagramme de l'un des sous-VIs du VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi). Ce sous-VI particulier est le VI "Fonct. cible & dérivée non lin." (Target Fnc & Deriv NonLin.vi). Vous pouvez accéder à ce VI en le sélectionnant dans le menu qui apparaît lorsque vous sélectionnez **Projet»Sous-VIs appelés par ce VI**.

Remarque ***Lors de l'utilisation du VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi), vous devez également préciser la fonction non linéaire dans la boîte de calcul se trouvant sur le diagramme du VI "Fonct. cible & dérivée non lin." (Target Fnc & Deriv NonLin.vi).***

Les connexions du VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi) sont représentées ci-dessous :



X et **Y** sont les points d'entrée x[i] et y[i].

Les **Coefficients estimés initiaux** représentent votre estimation initiale de la valeur des coefficients. Les coefficients sont ceux utilisés dans la formule que vous avez entrée dans la boîte de calcul du VI **"Fonct. cible & dérivée non lin." (Target Fnc & Deriv NonLin.vi)**. Le succès de l'utilisation du VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi) dépend parfois du degré de similarité entre vos coefficients estimés initiaux et la solution réelle. Il est donc toujours judicieux de prendre le temps et la peine d'obtenir de bons coefficients estimés initiaux en s'aidant de toutes les ressources disponibles.

Vous pouvez pour l'instant laisser les autres entrées sur leurs valeurs par défaut. Pour plus d'informations sur ces entrées, consultez la *Référence d'analyse en ligne*.

Meilleurs coefficients d'ajustement : la valeur des coefficients (a_0, a_1, ...) qui s'ajustent le mieux au modèle des données expérimentales.

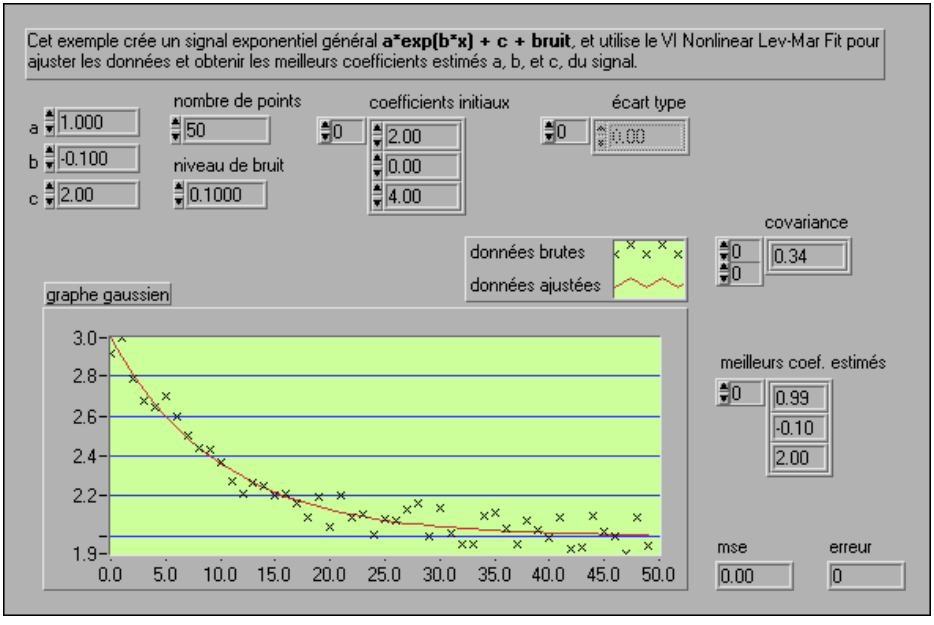
Exercice 17-3. Utiliser le VI “Ajustement de Lev-Mar non linéaire” (Nonlinear Lev-Mar Fit.vi)

Dans cet exercice, votre objectif est de créer un signal exponentiel général de type a*exp(b*x) + c + bruit, puis d’utiliser le VI “Ajustement de Lev-Mar non linéaire” (Nonlinear Lev-Mar Fit.vi) pour ajuster les données et obtenir la meilleure estimation possible des coefficients a, b et c du signal exponentiel général.

Dans cet exercice, vous apprendrez à utiliser le **VI “Ajustement de Lev-Mar non linéaire”** (Nonlinear Lev-Mar Fit.vi) pour déterminer les coefficients *a*, *b* et *c* d’une fonction non linéaire donnée par a*exp(b*x) + c.

Face-avant

1. Ouvrez le VI “Ajustement exponentiel de Lev-Mar non linéaire” (Nonlinear Lev-Mar Exponential Fit.vi) dans la bibliothèque `examples\analysis\regressn.llb`. La face-avant est représentée dans l’illustration suivante.



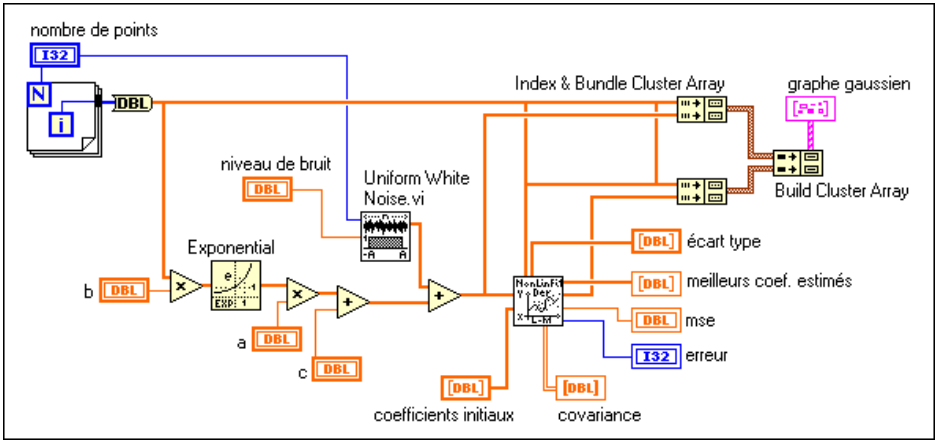
Les commandes **a**, **b** et **c** déterminent les valeurs réelles des coefficients *a*, *b* et *c*. La commande **Coefficients initiaux (Initial Coefficients)** représente votre meilleure estimation des valeurs réelles de *a*, *b* et *c*. L'indicateur **Meilleurs coefficients estimés** (**Best Guess Coef**) vous donne les valeurs de *a*, *b* et *c* calculées par le VI "Ajustement de Lev-Mar non linéaire" (Nonlinear Lev-Mar Fit.vi). Pour simuler un exemple plus pratique, nous ajoutons également du bruit à cette équation, qui prend ainsi la forme :

$$a*\exp(b*x) + c + \text{bruit}$$

La commande **Niveau de bruit (noise level)** ajuste le niveau de bruit. Remarquez que les valeurs réelles de *a*, *b* et *c* choisies sont +1,0 ; –0,1 ; et 2,0. Dans la commande **Coefficients initiaux**, l'estimation par défaut de ces coefficients est *a* = 2,0 ; *b* = 0 ; et *c* = 4,0.

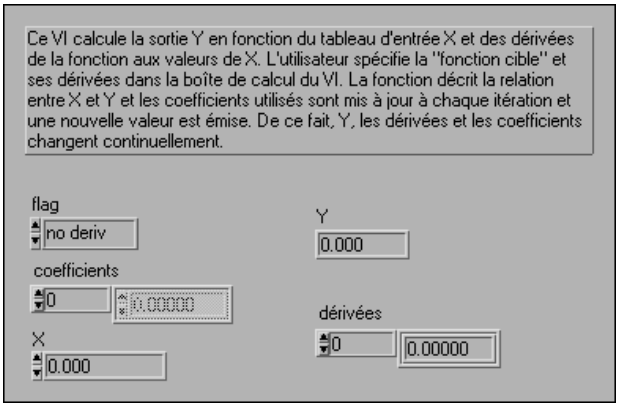
Diagramme

2. Examinez le diagramme.

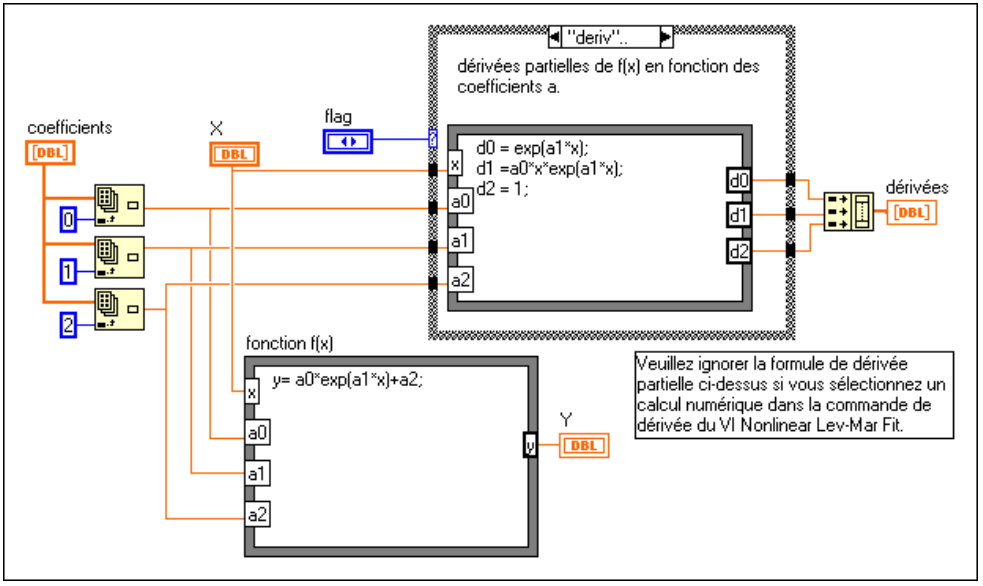


Les échantillons des données de la fonction exponentielle sont simulés en utilisant le VI "Exponentiel" (Exponential.vi) (sous-palette **Numérique»Logarithmique**) et un bruit blanc uniforme est ajouté aux échantillons à l'aide du VI "Bruit blanc uniforme" (Uniform White Noise.vi) (sous-palette **Analyse»Génération de signal**).

3. Dans le menu **Projet**, sélectionnez **Sous-VIs non ouverts»VI “Fonct. cible & dérivée non lin.” (Target Fnc & Deriv NonLin.vi)**. La face-avant du VI “Fonct. cible & dérivée non lin.” (Target Fnc & Deriv NonLin.vi) s’ouvre, comme indiqué ci-dessous.

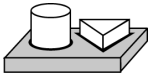


4. Revenez dans le diagramme.



Observez la boîte de calcul en bas du diagramme. Elle possède la forme de la fonction dont vous essayez d’évaluer les paramètres (a0, a1 et a2).

5. Fermez la face-avant et le diagramme du VI “Fonct. cible & dérivée non lin.” (Target Fnc & Deriv NonLin.vi).
6. Exécutez le **VI “Ajustement exponentiel de Lev-Mar non linéaire” (NonLinear Lev-Mar Exponential Fit.vi)**. Remarquez que les valeurs des coefficients retournées dans **Meilleurs coefficients estimés** sont très proches des valeurs réelles entrées dans la commande **Coefficients initiaux**. Notez également la valeur de l’**erreur quadratique moyenne** (standard deviation).
7. Augmentez le **Niveau de bruit** (à l’origine égal à 0,1) jusqu’à 0,5. Qu’advient-il de l’**erreur quadratique moyenne** et des valeurs des coefficients dans **Meilleurs coefficients estimés** ? Pourquoi ?
8. Remettez le **Niveau de bruit** sur 0,1. Faites passer les **Coefficients initiaux sur** 5,0 ; –2,0 ; et 10,0. Exécutez ensuite le VI. Notez les valeurs retournées dans les indicateurs **Meilleurs coefficients estimés** et **erreur quadratique moyenne**.
9. Avec le **Niveau de bruit** toujours à 0,1, faites passer votre estimation des **Coefficients initiaux** sur 5,0 ; 8,0 ; et 10,0. Exécutez ensuite le VI. Cette fois-ci, votre estimation est plus éloignée que celle formulée à l’étape 4. Prenez note de l’erreur. Ceci illustre combien il est important d’avoir une estimation juste et raisonnée des coefficients.
10. Lorsque vous avez terminé, fermez le VI. N’enregistrez pas les changements.



Fin de l’exercice 17-3.

18

Algèbre linéaire

Ce chapitre explique comment utiliser les VIs d'algèbre linéaire pour effectuer des analyses et des calculs matriciels. Pour des exemples d'utilisation de VIs d'algèbre linéaire, consultez la bibliothèque `examples\analysis\linxmpl.llb`.

Systèmes linéaires et analyse de matrice

Il existe des systèmes d'équations algébriques linéaires dans de nombreuses applications qui impliquent des calculs scientifiques, notamment le traitement de signal et la dynamique des fluides computationnelle. De tels systèmes peuvent exister naturellement ou être le résultat d'approximation d'équations différentielles par des équations algébriques.

Types de matrices

Quelle que soit l'application, il est toujours nécessaire de trouver la solution exacte d'un système d'équations de la façon la plus efficace. Dans la notation matrice-vecteur, un tel système d'équations algébriques linéaires est de la forme

$$Ax = b$$

où A est une matrice $n \times n$, b est un vecteur donné comprenant n éléments, et x est le vecteur solution à déterminer. Une matrice est représentée par un tableau 2D composé d'éléments divers. Ces éléments peuvent être des nombres réels, des nombres complexes, des fonctions ou des opérateurs. La matrice A représentée ci-dessous est un tableau de m rangées et n colonnes contenant $m \times n$ éléments.

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m-1,0} & a_{m-1,1} & \cdots & a_{m-1,n-1} \end{bmatrix}$$

Dans ce cas, $a_{i,j}$ représente le $(i,j)^{ème}$ élément situé dans la $i^{ème}$ rangée et la $j^{ème}$ colonne. En général, une telle matrice est appelée une *matrice rectangulaire*. Lorsque $m = n$, c'est-à-dire lorsque le nombre de rangées est égal au nombre de colonnes, il s'agit d'une *matrice carrée*. Une matrice $m \times 1$ (m rangées et une colonne) est appelée un *vecteur-colonne*. Un *vecteur-rangée* est une matrice $1 \times n$ (1 rangée et n colonnes). Si tous les éléments autres que les éléments diagonaux sont nuls (c'est-à-dire que si $a_{i,j} = 0,\ i \neq j$), la matrice est appelée une *matrice diagonale*. Par exemple,

$$A = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

est une matrice diagonale. Une matrice diagonale dont tous les éléments diagonaux sont égaux à la valeur 1 est appelée une *matrice d'identité*, ou *matrice d'unité*. Si tous les éléments au-dessous de la diagonale principale sont nuls, la matrice est appelée une *matrice triangulaire supérieure*. A l'opposé, si tous les éléments au-dessus de la diagonale principale sont nuls, la matrice est une *matrice triangulaire inférieure*. Lorsque tous les éléments sont des nombres réels, la matrice est une *matrice réelle*. Lorsque au moins un des éléments de la matrice est un nombre complexe, la matrice est une *matrice complexe*. Pour plus de clarté, vous travaillerez principalement dans cette leçon avec des matrices réelles. Toutefois, pour les plus aventureux, nous avons également ajouté des exercices impliquant des matrices complexes.

Déterminant d'une matrice

L'un des attributs les plus importants d'une matrice est son **déterminant**. Dans le cas le plus simple, le déterminant d'une matrice 2 x 2

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

est calculé par la formule $ad - bc$. Vous pouvez obtenir le déterminant d'une matrice carrée en prenant le déterminant de ses éléments. Par exemple, si

$$A = \begin{bmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{bmatrix}$$

alors le déterminant de **A**, représenté par $|A|$, est

$$|A| = \left|\begin{bmatrix} 2 & 5 & 3 \\ 6 & 1 & 7 \\ 1 & 6 & 9 \end{bmatrix}\right| = \left(2\left|\begin{bmatrix} 1 & 7 \\ 6 & 9 \end{bmatrix}\right| - 5\left|\begin{bmatrix} 6 & 7 \\ 1 & 9 \end{bmatrix}\right| + 3\left|\begin{bmatrix} 6 & 1 \\ 1 & 6 \end{bmatrix}\right|\right) =$$

$$2(-33) - 5(47) + 3(35) = -196$$

Le déterminant reflète des propriétés importantes de la matrice. Par exemple, si le déterminant de la matrice est zéro, la matrice est *singulière*. En d'autres termes, la matrice ci-dessus (avec un déterminant non nul) est *non singulière*. Vous reverrez le concept de singularité ultérieurement, dans la section *Matrice inverse et résolution de systèmes d'équations linéaires* concernant la solution des équations linéaires et des matrices inverses.

Transposée d'une matrice

La **transposée** d'une matrice réelle est obtenue en permutant ses rangées et ses colonnes. Si la matrice **B** représente la transposée de **A**, indiquée par $\mathbf{A}^T$, alors $b_{j,i}=a_{i,j}$. Pour la matrice **A** définie ci-dessus,

$$B = A^T = \begin{bmatrix} 2 & 6 & 1 \\ 5 & 1 & 6 \\ 3 & 7 & 9 \end{bmatrix}$$

Dans le cas de matrices complexes, on définit une transposition conjuguée complexe. Si la matrice **D** représente la *transposée complexe conjuguée* (si $a = x + iy$, alors le complexe conjugué est $a^* = x - iy$) d'une matrice complexe **C**, alors

$$D = C^H \Rightarrow d_{i,j} = c^*_{j,i}$$

La matrice *D* est donc obtenue en remplaçant chaque élément de **C** par son complexe conjugué et en permutant les rangées et les colonnes de la matrice obtenue.

Une matrice réelle est appelée une *matrice symétrique* si la transposée de la matrice est égale à la matrice elle-même. La matrice *A* de notre exemple n'est pas une matrice symétrique. Si une matrice complexe **C** satisfait à relation $C = C^H$, alors *C* est appelée une *matrice hermitienne*.

Peut-on obtenir un vecteur égal à une combinaison linéaire d'autres vecteurs ? (indépendance linéaire)

Un ensemble de vecteurs x_1, x_2, ..., x_n est *linéairement dépendant* si et seulement s'il existe des scalaires α_1, α_2, ..., α_n, non tous nuls, tels que

$$\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n = 0$$

Plus simplement, si l'un des vecteurs peut être écrit sous la forme d'une combinaison linéaire des autres vecteurs, alors les vecteurs sont linéairement dépendants.

Si le seul ensemble de α_i, pour lequel l'équation ci-dessus est vérifiée, est $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$, alors l'ensemble de vecteurs x_1, x_2, ..., x_n est qualifié de *linéairement indépendant*. Dans ce cas, aucun vecteur ne peut être écrit sous la forme d'une combinaison linéaire des autres vecteurs. Soit un ensemble de vecteurs quelconque, l'équation ci-dessus est toujours résolue pour $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$. Pour prouver l'indépendance linéaire de l'ensemble, vous devez donc démontrer que $\alpha_1 = 0$, $\alpha_2 = 0$, ..., $\alpha_n = 0$ est le seul ensemble de α_i pour lequel l'équation ci-dessus est vérifiée.

Par exemple, considérons d'abord les vecteurs

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Remarquez que $\alpha_1 = 0$ et $\alpha_2 = 0$ sont les seules valeurs pour lesquelles la relation $\alpha_1 x + \alpha_2 y = 0$ est vérifiée. Ces deux vecteurs sont donc linéairement indépendants l'un de l'autre. Considérons maintenant les vecteurs

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

Remarquez que si $\alpha_1 = -2$ et $\alpha_2 = 1$, alors $\alpha_1 x + \alpha_2 y = 0$. Ces deux vecteurs sont donc linéairement dépendants l'un de l'autre. Vous devez bien comprendre la notion d'indépendance linéaire de vecteurs pour apprécier entièrement le concept de *rang* de matrice, traité dans la section suivante.

Comment déterminer l'indépendance linéaire ? (rang d'une matrice)

Le *rang* d'une matrice *A*, représenté par ρ(*A*), est le nombre maximum de colonnes linéairement indépendantes de *A*. Si vous examinez la matrice *A de l'*exemple *précédent*, vous remarquez que toutes les colonnes de *A* sont linéairement indépendantes les unes des autres. En d'autres termes, aucune colonne ne peut être obtenue grâce à une combinaison linéaire des autres colonnes. Le rang de la matrice est donc 3. Considérons une autre matrice *B*, où

$$B = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 0 & 2 \end{bmatrix}$$

Cette matrice ne possède que deux colonnes linéairement indépendantes, car la troisième colonne de *B* est linéairement dépendante des deux premières colonnes. Le rang de cette matrice est donc 2. Il peut être démontré que le nombre de colonnes linéairement indépendantes d'une matrice est égal au nombre de rangées indépendantes. Le rang ne peut donc jamais être supérieur à la plus petite dimension de la matrice. Par conséquent, si *A* est une matrice $n \times m$, alors

$$\rho(A) \leq min(n, m)$$

où *min* représente le minimum des deux nombres. Selon la théorie des matrices, le rang d'une matrice carrée dépend de la matrice non singulière du plus haut ordre formé à partir de celle-ci. Souvenez-vous (d'après l'une des sections précédentes) qu'une matrice est singulière si son déterminant est nul. Le rang dépend de la matrice d'ordre le plus élevé que vous pouvez obtenir et dont le déterminant est non nul. Par exemple, considérons une matrice 4 x 4

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 2 \end{bmatrix}$$

Pour cette matrice, $det(B) = 0$, mais

$$\left|\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}\right| = -1$$

Le rang de *B* est donc 3. Une matrice carrée est de rang complet si et seulement si son déterminant est différent de zéro. La matrice *B* n'est pas une matrice de rang complet.

"Grandeur" (normes) de matrices

Vous devez développer la notion de "grandeur" des vecteurs et des matrices pour mesurer les erreurs et la sensibilité en jeu dans la résolution d'un système d'équations linéaire. Ces systèmes linéaires peuvent être notamment obtenus à partir d'applications de systèmes de contrôle et en dynamique des fluides computationnelle. En deux dimensions, par exemple, vous ne pouvez pas comparer deux vecteurs $x = \begin{bmatrix} x1 & x2 \end{bmatrix}$ et $y = \begin{bmatrix} y1 & y2 \end{bmatrix}$, car vous pouvez avoir $x1 > y1$ mais $x2 < y2$. La norme des vecteurs permet d'affecter une quantité scalaire à ces vecteurs afin qu'ils puissent être comparés entre eux. Ceci est similaire au concept d'amplitude, de module ou de valeur absolue rencontré avec les nombres scalaires.

Il y a plusieurs façons de calculer la norme d'une matrice, notamment la *norme 2* (norme euclidienne), la *norme 1*, la *norme de Frobenius* (norme de type F) et la *norme infinie* (norme à l'infini). Chaque norme possède sa propre interprétation physique. Considérons un cercle unité centré à l'origine. La norme euclidienne d'un vecteur est simplement le facteur d'agrandissement ou de diminution du cercle pour qu'il puisse renfermer exactement le vecteur donné. Ce concept est représenté dans les figures ci-dessous :

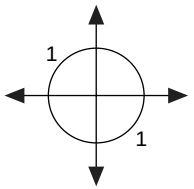


Figure 1a

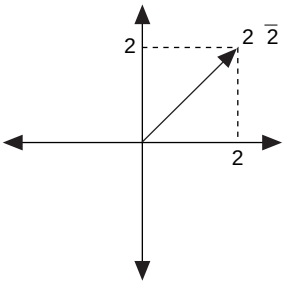


Figure 1b

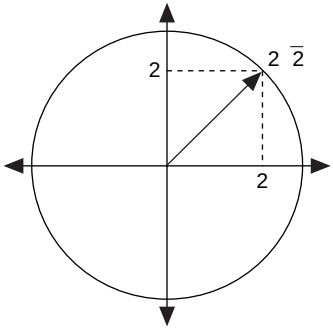


Figure 1c

La figure 1a représente un cercle unité de rayon égal à 1 unité. La figure 1b représente un vecteur de longueur $\sqrt{2^2 + 2^2} = \sqrt{8} = 2\sqrt{2}$. Comme indiqué dans la figure 1c, le cercle unité doit être agrandi par un facteur de $2\sqrt{2}$ pour pouvoir contenir exactement le vecteur donné. La norme euclidienne du vecteur est donc $2\sqrt{2}$.

La norme d'une matrice est définie comme une norme de vecteur sous-jacente. Il s'agit de l'étirement relatif maximum effectué par la matrice sur un vecteur quelconque. Avec la *norme 2* de vecteur, le cercle unité est agrandi par un facteur égal à la norme. En revanche, avec la *norme 2* de matrice, le cercle unité peut devenir une ellipsoïde (ellipse en 3D), avec certains axes plus longs que d'autres. L'axe le plus long détermine la norme de la matrice.

Certaines normes de matrice sont bien plus faciles à calculer que d'autres. La *norme 1* est obtenue en effectuant la somme des valeurs absolues de tous les éléments de chaque colonne de la matrice. La somme la plus élevée est appelée la norme 1. En termes mathématiques, la norme 1 est simplement la somme maximum des valeurs absolues des colonnes de la matrice.

$$\|\boldsymbol{A}\|_1 = max_j \sum_{i=0}^{n-1} |a_{i,j}|$$

Par exemple,

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

alors

$$\|A\|_1 = max(3, 7) = 7$$

La *norme infinie* d'une matrice est la somme maximum des valeurs absolues des rangées de la matrice

$$\|\boldsymbol{A}\|_\infty = max_i \sum_{j=0}^{n-1} |a_{i,j}|$$

Dans ce cas, vous ajoutez les "grandeurs" de tous les éléments de chaque rangée de la matrice. La valeur maximum obtenue est appelée la norme infinie. Pour la matrice de l'exemple ci-dessus,

$$\|\boldsymbol{A}\|_\infty = max(4, 6) = 6$$

La *norme 2* est la plus difficile à calculer car elle est donnée par la valeur singulière la plus élevée de la matrice. Les valeurs singulières sont traitées dans la section *Factorisation de matrice*.

Détermination de la singularité (nombre conditionnel)

Tandis que la norme d'une matrice permet de mesurer sa "grandeur", le *nombre conditionnel* d'une matrice mesure le niveau de singularité de la matrice. Le nombre conditionnel d'une matrice carrée non singulière est défini par

$$cond(\boldsymbol{A}) = \|\boldsymbol{A}\|_p \cdot \left\|A^{-1}\right\|_p$$

où *p* représente l'un des quatre types de normes mentionnés précédemment. Par exemple, pour déterminer le nombre conditionnel d'une matrice *A*, vous pouvez rechercher la norme 2 de *A*, la norme 2 de l'inverse de la matrice *A*, représentée par A^{-1}, puis les multiplier ensemble (l'inverse d'une matrice carrée A est une matrice carrée B telle que $AB=I$, où I est la matrice d'identité). Comme mentionné précédemment, la norme 2 est difficile à calculer manuellement. Vous pouvez utiliser le VI "Norme de la matrice" (Matrix Norm.vi) de la bibliothèque d'analyse LabVIEW pour calculer la norme 2. Par exemple,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, A^{-1} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}, \|A\|_2 = 5.4650, \left\|A^{-1}\right\|_2$$

$$= 2.7325, cond(A) = 14.9331$$

Le nombre conditionnel peut varier entre 1 et l'infini. Une matrice avec un nombre conditionnel élevé est proche de la singularité, alors qu'une matrice avec un nombre conditionnel proche de 1 en est très éloignée. La matrice *A* ci-dessus est non singulière. Toutefois, considérons la matrice

$$B = \begin{bmatrix} 1 & 0.99 \\ 1.99 & 2 \end{bmatrix}$$

Le nombre conditionnel de cette matrice est 47168, et la matrice est donc proche de la singularité. Comme vous vous en souvenez peut-être, une matrice est singulière si son déterminant est égal à zéro. Le déterminant n'est toutefois pas un bon indicateur du degré de singularité d'une matrice. Pour la matrice *B* ci-dessus, le déterminant (0,0299) est non nul ; toutefois, un nombre conditionnel avec une valeur élevée indique que la matrice est presque singulière. Souvenez-vous également que le nombre conditionnel d'une matrice est toujours supérieur ou égal à 1. Ceci est vrai pour les matrices d'identité et de permutation (une *matrice de permutation* est une matrice d'identité avec certaines rangées et colonnes permutées). Le nombre conditionnel est une quantité très utile pour évaluer l'exactitude des solutions dans des systèmes linéaires.

Dans cette section, vous vous êtes familiarisé(e) avec les notations de base et les concepts fondamentaux des matrices, tels que le déterminant et le rang d'une matrice. L'exercice suivant vous permettra de mieux comprendre ces termes, qui seront utilisés fréquemment dans le reste de la leçon.

Opérations matricielles de base et problèmes de valeurs propres-vecteurs propres

Dans cette section, nous traiterons des opérations matricielles de base. Deux matrices, *A* et *B*, sont dites égales si elles ont le même nombre de rangées et de colonnes et si leurs éléments correspondants sont tous égaux. La multiplication d'une matrice *A* par un scalaire α est égale à la multiplication de tous ses éléments par le scalaire. C'est-à-dire que

$$\boldsymbol{C} = \alpha \boldsymbol{A} \Rightarrow c_{i,j} = \alpha a_{i,j}$$

Par exemple,

$$2\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

Deux matrices (ou davantage) peuvent être ajoutées ou soustraites si et seulement si elles possèdent le même nombre de rangées et de colonnes. Si les deux matrices *A* et *B* ont *m* rangées et *n* colonnes, alors leur somme *C* est une matrice *m*-fois-*n* définie par $C = A \pm B$, où $c_{i,j} = a_{i,j} \pm b_{i,j}$. Par exemple,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 4 \\ 5 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 8 & 5 \end{bmatrix}$$

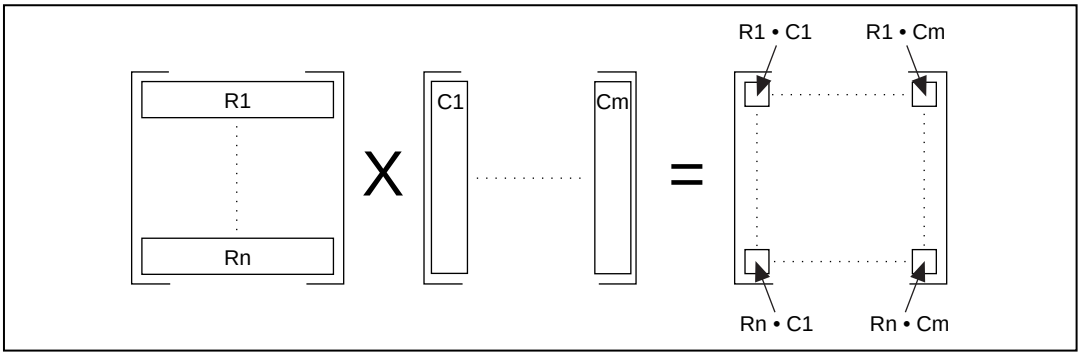
Pour la multiplication de deux matrices, le nombre de colonnes de la première matrice doit être égal au nombre de rangées de la seconde matrice. Si la matrice *A* possède *m* rangées et *n* colonnes et si la matrice *B* comprend *n* rangées et *p* colonnes, alors leur produit *C* est une matrice *m*-fois-*p* définie par $C = AB$, où

$$c_{i,j} = \sum_{k=0}^{n-1} a_{i,k} b_{k,j}$$

Par exemple,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 4 \\ 5 & 1 \end{bmatrix} = \begin{bmatrix} 12 & 6 \\ 26 & 16 \end{bmatrix}$$

Pour obtenir les éléments de la première rangée et de la première colonne de *C*, vous devez donc multiplier les éléments de la première rangée de *A* par les éléments correspondants de la première colonne de *B* et ajouter tous les résultats. De façon similaire, pour calculer l'élément de la $i^{ème}$ rangée et la $j^{ème}$ colonne de *C*, multipliez les éléments de la $i^{ème}$ rangée de *A* par les éléments correspondants de la $j^{ème}$ colonne de *C*, puis ajoutez-les tous ensemble. Ceci est représenté graphiquement ci-dessous :



La multiplication de matrices n'est en général pas commutative, c'est-à-dire que $AB \neq BA$. De plus, souvenez-vous que la multiplication d'une matrice par une matrice d'identité est égale à la matrice d'origine.

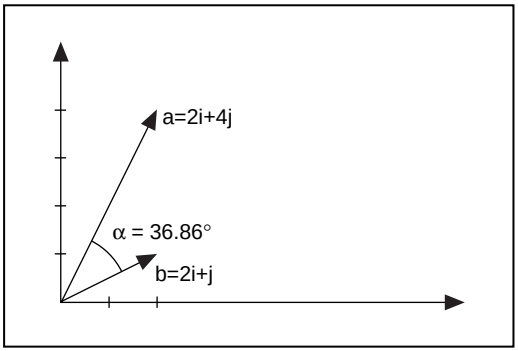
Produit scalaire et produit externe

Si *X* représente un vecteur et si *Y* en représente un autre, on obtient le *produit scalaire* de ces deux vecteurs en multipliant les éléments correspondants de chaque vecteur et en ajoutant les résultats. Ceci est représenté par

$$X \bullet Y = \sum_{i=0}^{n-1} x_i y_i$$

où *n* est le nombre d'éléments de *X* et *Y*. Remarquez que les deux vecteurs doivent posséder le même nombre d'éléments. Le produit scalaire est une quantité scalaire qui intervient dans de nombreuses applications pratiques.

Par exemple, considérons les vecteurs $a = 2i + 4j$ et $b = 2i + j$ dans un système de coordonnées rectangulaire à deux dimensions.



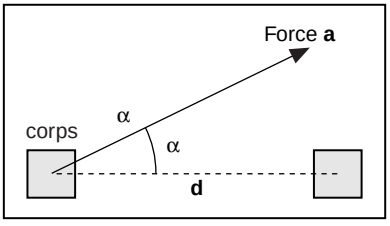
Le produit scalaire de ces deux vecteurs est donné par

$$d = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \bullet \begin{bmatrix} 2 \\ 1 \end{bmatrix} = (2 \times 2) + (4 \times 1) = 8$$

L'angle α entre ces deux vecteurs est donné par

$$\alpha = inv\cos\left(\frac{a \bullet b}{|a||b|}\right) = inv\cos\left(\frac{8}{10}\right) = 36.86^{o}$$

où |**a**| représente la "grandeur" (valeur absolue) de *a*.



Comme seconde application, considérons un objet subissant une force constante *a*. Le travail W effectué par *a* lors du déplacement de l'objet est défini comme le produit de |*d*| et de la composante de *a* dans la direction de déplacement *d*. C'est-à-dire :

$$W = |a||d|\cos\alpha = a \bullet d$$

Par ailleurs, le *produit externe* de ces deux vecteurs est une matrice. Le *(i,j)*^{ème} élément de cette matrice est obtenu grâce à la formule

$$a_{i,j} = x_i \times y_j$$

Par exemple,

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix}$$

Valeurs propres et vecteurs propres

Pour comprendre les valeurs propres et les vecteurs propres, commençons par leur définition classique. Soit une matrice *A* de type $n \times n$, déterminons un scalaire λ et un vecteur *x* non nul tel que

$$Ax = \lambda x$$

Un tel scalaire λ est appelé une *valeur propre*, et *x* est le *vecteur propre* correspondant.

Les calculs des valeurs propres et des vecteurs propres sont des principes fondamentaux d'algèbre linéaire. Ils vous permettent de résoudre de nombreux problèmes, tels que des systèmes d'équations différentielles, lorsque vous comprenez ce qu'ils représentent. Considérez un vecteur propre *x* d'une matrice *A* comme un vecteur non nul qui ne change pas de direction lorsque *x* est multiplié par *A* (excepté peut-être pour pointer dans la direction exactement opposée). Le vecteur propre *x* peut varier en longueur ou avoir une direction opposée, mais ne peut pas pivoter sur les côtés. En d'autres termes, il existe une constante scalaire λ pour laquelle l'équation ci-dessus est vraie. La valeur λ est une *valeur propre* de *A*.

Considérons l'exemple suivant. L'un des vecteurs propres de la matrice *A*, où

$$A = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}$$

est le vecteur

$$x = \begin{bmatrix} 0.62 \\ 1.00 \end{bmatrix}$$

La multiplication de la matrice *A* par le vecteur *x* force simplement le vecteur *x* à s'agrandir avec un facteur de 6,85. La valeur 6,85 est donc l'une des valeurs propres du vecteur *x*. Pour toute constante α, le vecteur αx est également un vecteur propre avec la valeur propre λ, car

$$A(\alpha x) = \alpha Ax = \lambda \alpha x$$

En d'autres termes, un vecteur propre d'une matrice détermine une direction dans laquelle la matrice étend ou diminue par un multiple scalaire un vecteur quelconque se trouvant dans cette direction. Le facteur d'expansion ou de contraction est donné par la valeur propre correspondante. Un problème *généralisé* de valeur propre consiste à déterminer un scalaire λ et un vecteur non nul *x* tel que

$$Ax = \lambda Bx$$

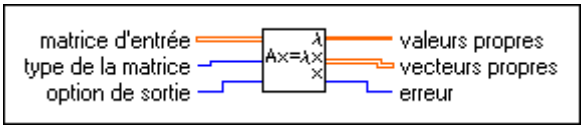
où *B* est une autre matrice $n \times n$.

Certaines propriétés importantes des valeurs propres et des vecteurs propres sont mentionnées ci-après :

- Les valeurs propres d'une matrice ne sont pas nécessairement toutes distinctes. En d'autres termes, une matrice peut posséder plusieurs valeurs propres.
- Il n'est pas nécessaire que toutes les valeurs propres d'une matrice réelle soient réelles. Cependant, les valeurs propres complexes d'une matrice réelle doivent être des paires de complexes conjugués.
- Les valeurs propres d'une matrice diagonale sont ses entrées diagonales, et les vecteurs propres sont les colonnes correspondantes d'une matrice d'identité de la même dimension.
- Une matrice réelle symétrique possède toujours des valeurs propres et des vecteurs propres.
- Comme mentionné précédemment, les vecteurs propres peuvent être mis à l'échelle de façon arbitraire.

Il existe de nombreuses applications pratiques dans le domaine scientifique et de l'ingénierie du problème des valeurs propres. Par exemple, la stabilité d'une structure et ses modes et fréquences naturels de vibration sont déterminés par les valeurs propres et les vecteurs propres d'une matrice appropriée. Les valeurs propres sont également très utiles pour analyser des méthodes numériques, notamment l'analyse de convergence des méthodes itératives pour résoudre des systèmes d'équations algébriques, et l'analyse de stabilité des méthodes pour résoudre des systèmes d'équations différentielles.

Le VI “Valeurs et vecteurs propres” (EigenValues and Vectors.vi) est représenté ci-dessous. La **matrice d’entrée** est une matrice carrée réelle N-fois-N. Le **type de la matrice** détermine le type de la matrice d’entrée. Le **type de la matrice** peut être *0*, indiquant une *matrice générale*, ou 1, indiquant une *matrice symétrique*. Une matrice symétrique possède toujours des valeurs propres réelles et des vecteurs propres. Une matrice générale n’a pas de propriété spéciale comme une structure symétrique ou triangulaire.



La commande **option de sortie** détermine ce qui doit être calculé. Si l’option de sortie = 0, ceci indique que seules les valeurs propres doivent être calculées. Si l’option de sortie = 1, les valeurs propres et les vecteurs propres doivent être calculés. Le calcul des valeurs propres et des vecteurs propres requiert beaucoup de temps et d’effort. Il est donc important que vous utilisiez avec précaution la commande **option de sortie** du VI “Valeurs et vecteurs propres” (EigenValues and Vectors.vi). Selon votre application particulière, vous voudrez peut-être calculer uniquement les valeurs propres, ou les valeurs propres et les vecteurs propres. De plus, sachez qu’une matrice symétrique nécessite moins de calculs qu’une matrice asymétrique. Choisissez donc avec soin la commande **type de la matrice**.

Dans cette section, vous avez appris certaines opérations matricielles de base ainsi que la notion de valeurs propres et de vecteurs propres. L’exemple suivant introduit certains VIs de la bibliothèque d’analyse qui réalisent ces opérations.

Matrice inverse et résolution de systèmes d’équations linéaires

L’*inverse*, représenté par A^{-1}, d’une matrice carrée A est une matrice carrée telle que

$$A^{-1}A = AA^{-1} = I$$

où I est la matrice d’identité. L’inverse d’une matrice existe si et seulement si le déterminant de la matrice est non nul (c’est-à-dire si et seulement si la matrice est non singulière). En général, vous pouvez seulement déterminer

l'inverse d'une matrice carrée. Vous pouvez toutefois calculer la *pseudo-inverse* d'une matrice rectangulaire, comme mentionné ultérieurement dans la section *Factorisation de matrice*.

Solutions de systèmes d'équations linéaires

En notation matrice-vecteur, un système d'équations linéaires est de la forme $Ax = b$, où A est une matrice $n \times n$ et b est un vecteur de n éléments donnés. Le but est de déterminer x, le vecteur solution de n éléments inconnus. Vous devez vous poser deux importantes questions concernant l'existence d'une telle solution. Une telle solution existe-t-elle et si oui, est-elle unique ? La réponse à ces deux questions repose dans la détermination de la singularité (ou de la non-singularité) de la matrice A.

Comme mentionné précédemment, on dit qu'une matrice est singulière si elle possède l'une des propriétés équivalentes suivantes :

- L'inverse de la matrice n'existe pas.
- Le déterminant de la matrice est égal à zéro.
- Les rangées (ou colonnes) de A sont linéairement dépendantes.
- $Az = 0$ pour un vecteur quelconque $z \neq 0$.

Sinon, la matrice est non singulière. Si la matrice est non singulière, son inverse A^{-1} existe, et le système $Ax = b$ possède une solution unique : $x = A^{-1}b$, quelle que soit la valeur de b. Par contre, si la matrice est singulière, alors le nombre de solutions est déterminé par le vecteur b. Si la matrice A est singulière et si $Ax = b$, alors $A(x + \Upsilon z) = b$ pour tout scalaire Υ, où le vecteur z est tel que dans la dernière définition ci-dessus. Si un système singulier possède une solution, la solution ne peut donc pas être unique.

Il n'est pas judicieux de calculer explicitement l'inverse d'une matrice, car un tel calcul est sujet à des imprécisions numériques. Il n'est donc pas conseillé de résoudre un système d'équations linéaires en multipliant l'inverse de la matrice A par le vecteur connu à droite du signe égal (=). La stratégie générale pour résoudre un tel système d'équations est la suivante : transformez le système d'origine en un système dont la solution est identique au système d'origine, mais plus simple à calculer. Vous pouvez notamment utiliser la technique d'élimination de Gauss. Consultez l'Annexe A, *Références d'analyse*, pour plus d'informations sur les calculs matriciels. Les trois étapes de base impliquées dans la technique

d'élimination de Gauss sont les suivantes. Exprimons d'abord la matrice *A* comme un produit

$$A = LU$$

où *L* est une matrice unité triangulaire inférieure et *U* est une matrice triangulaire supérieure. Une telle factorisation est appelée factorisation LU. Ceci fait, le système linéaire $Ax = b$ peut être exprimé comme $LUx = b$. Un tel système peut alors être résolu en déterminant d'abord *y* dans le système triangulaire inférieur $Ly = b$ par *substitution avant*. Ceci est la deuxième étape de la technique d'élimination de Gauss. Par exemple, si

$$l = \begin{bmatrix} a & 0 \\ b & c \end{bmatrix} \qquad y = \begin{bmatrix} p \\ q \end{bmatrix} \qquad b = \begin{bmatrix} r \\ s \end{bmatrix}$$

alors

$$p = \frac{r}{a}, q = \frac{(s - bp)}{c}$$

Le premier élément de *y* peut être aisément déterminé grâce à la nature triangulaire inférieure de la matrice *L*. Vous pouvez alors utiliser cette valeur pour calculer séquentiellement les éléments restants du vecteur inconnu (d'où le nom de "substitution avant"). L'étape finale implique la résolution du système triangulaire supérieur $Ux = y$ par *substitution arrière*. Par exemple, si

$$U = \begin{bmatrix} a & b \\ 0 & c \end{bmatrix} \qquad x = \begin{bmatrix} m \\ n \end{bmatrix} \qquad y = \begin{bmatrix} p \\ q \end{bmatrix}$$

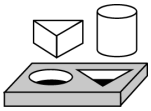
alors

$$n = \frac{q}{c}, m = \frac{(p - bn)}{a}$$

Dans ce cas, ce dernier élément de **x** peut être aisément déterminé puis utilisé pour déterminer séquentiellement les autres éléments (d'où le nom de "substitution arrière"). Jusqu'à présent, ce chapitre a traité le cas des matrices carrées. Comme une matrice non carrée est nécessairement singulière, le système d'équations doit avoir zéro solution ou une solution

non unique. Dans une telle situation, vous trouvez généralement une solution unique *x* qui résout le système linéaire de façon approximative.

La bibliothèque d'analyse fournit des VIs pour calculer l'inverse d'une matrice, calculer la décomposition LU d'une matrice et résoudre un système d'équations linéaires. Il est important d'identifier correctement la matrice d'entrée pour éviter des calculs superflus, vous permettant de réduire les imprécisions numériques. Les quatre types possibles de matrices sont les matrices générales, les matrices définies positives, et les matrices triangulaires inférieures et supérieures. Une matrice réelle est définie positive si et seulement si elle est symétrique et si la forme quadratique de tous les vecteurs non nuls est X. Si la matrice d'entrée est carrée, mais si elle n'a pas un rang complet (*matrice à rang déficient*), le VI détermine la solution *x* des *moindres carrés*. La solution des moindres carrés est celle qui minimise la norme de $Ax - b$. Ceci est également vrai pour les matrices non carrées.



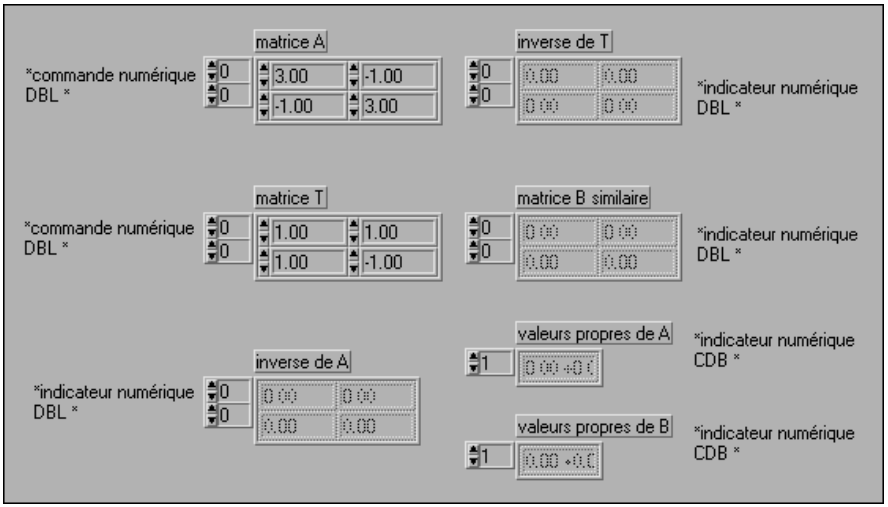
Exercice 18-1. Calculer l'inverse d'une matrice

Votre objectif est de calculer l'inverse d'une matrice.

Vous allez construire un VI qui calculera l'inverse d'une matrice *A*. Vous allez ensuite calculer une matrice *B similaire* à la matrice *A*. Une matrice *B* est similaire à une matrice *A* s'il existe une matrice non singulière *T* telle que $B = T^{-1}AT$, de sorte que *A* et *B* aient les mêmes valeurs propres. Vous vérifierez cette définition des matrices similaires.

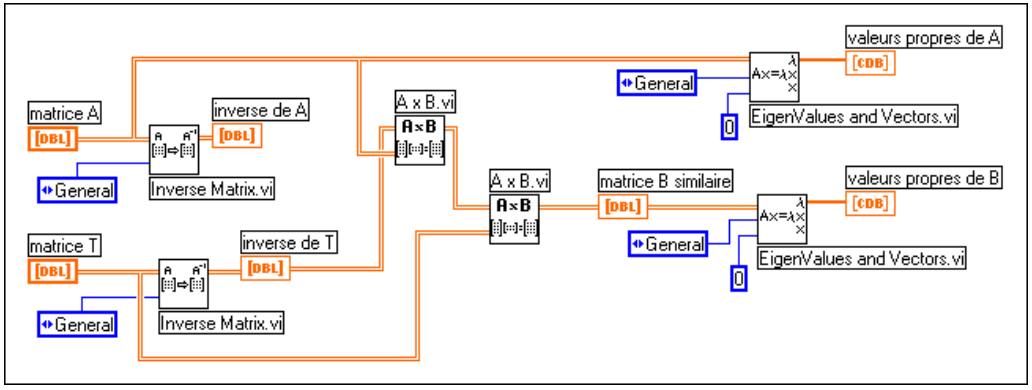
Face-avant

1. Construisez la face-avant comme indiqué ci-dessous. La matrice *A* est une matrice 2×2 réelle. La matrice *T* est une matrice 2×2 non singulière qui sera utilisée pour construire la matrice similaire *B*.



Diagramme

2. Ouvrez le diagramme et modifiez-le comme indiqué dans l'illustration suivante.

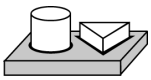


VI "Matrice inverse" (Inverse Matrix.vi) (sous-palette **Analyse»Algèbre linéaire**). Dans cet exercice, cette fonction calcule l'inverse de la matrice d'entrée *A*.

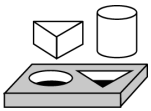
VI "AxB" (AxB.vi) (sous-palette **Analyse»Algèbre linéaire**). Dans cet exercice, cette fonction multiplie deux matrices d'entrée à deux dimensions.

VI “Valeurs et vecteurs propres” (Eigen Values and Vectors.vi) (sous-palette **Analyse»Algèbre linéaire**). Dans cet exercice, le VI calcule les valeurs propres et les vecteurs propres de la matrice d’entrée.

3. Enregistrez le VI sous le nom `Inverse de matrice.vi` dans le répertoire `LabVIEW\Activity`.
4. Revenez sur la face-avant et exécutez le VI. Vérifiez si les valeurs propres de *A* et de la matrice similaire *B* sont les mêmes.



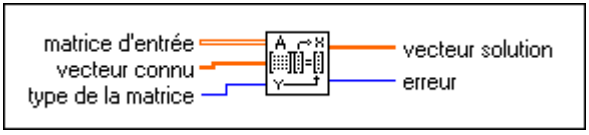
Fin de l’exercice 18-1.



Exercice 18-2. Résoudre un système d’équations linéaires

Votre objectif est de résoudre un système d’équations linéaires.

Vous avez besoin de résoudre un système d’équations linéaires dans de nombreuses applications pratiques. La défense militaire est l’un des domaines d’application très importants. Ceci inclut l’analyse de la diffusion électromagnétique et de la radiation de cibles volumineuses, l’analyse de performance de larges radomes, et la conception de véhicules aérospatiaux à faible surface équivalente radar (technologie Stealth). Un second domaine d’application porte sur la conception et la modélisation des systèmes de communication sans fil, tels que les téléphones cellulaires. Cette liste d’applications va s’élargissant et il est donc très important que vous compreniez bien comment utiliser les VIs de la bibliothèque d’analyse pour résoudre un système d’équations linéaires.



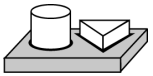
1. Utilisez le VI “Résoudre des équations linéaires” (`Solve Linear Equations.vi`) de la sous-palette **Analyse»Algèbre linéaire** pour

résoudre le système d'équations $Ax = b$ où la Matrice d'entrée *A* et le Vecteur connu *b* sont

$$A = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -1 & 7 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix}$$

Choisissez un type général pour la matrice.

2. Utilisez le VI "A x Vecteur" (A x Vector.vi) pour multiplier la matrice *A* par le vecteur *x* (sortie de l'opération ci-dessus) et vérifiez si le résultat est égal au vecteur *b* ci-dessus.
3. Enregistrez le VI sous le nom `Linear System.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l'exercice 18-2.

Factorisation de matrice

La section précédente traitait de la façon dont un système d'équations linéaires peut être transformé en un système dont la solution est plus simple à calculer, l'idée de base étant de factoriser la matrice d'entrée pour obtenir une multiplication de plusieurs matrices simplifiées. Vous avez découvert une telle technique, la technique de *décomposition LU*, avec laquelle vous avez factorisé la matrice d'entrée sous forme d'un produit de matrices triangulaires inférieures et supérieures. D'autres méthodes de factorisation couramment utilisées sont les méthodes de *Cholesky*, *QR* et la *décomposition en valeurs singulières (Singular Value Decomposition - SVD)*. Vous pouvez utiliser ces méthodes de factorisation pour résoudre de nombreux problèmes de matrice, comme la résolution d'un système d'équations linéaires, l'inversion d'une matrice et la recherche du déterminant d'une matrice.

Si la matrice d'entrée *A* est symétrique et définie positive, une factorisation LU peut être effectuée telle que $A = U^T U$, où *U* est une matrice triangulaire supérieure. Ceci s'appelle la *factorisation de Cholesky*. Cette méthode ne requiert que la moitié du travail et du stockage comparé à la factorisation LU d'une matrice générale par élimination gaussienne. Il est facile de déterminer si une matrice est définie positive en utilisant le VI "Matrice définie positive-test" (Test Positive Definite.vi) de la bibliothèque d'analyse.

Une matrice *Q* est *orthogonale* si ses colonnes sont *orthonormales*, c'est-à-dire si $Q^TQ = I$, la matrice d'identité. La technique de *factorisation QR* factorise une matrice sous forme de produit d'une matrice orthogonale *Q* et d'une matrice triangulaire supérieure *R*. En d'autres termes, $A = QR$. La factorisation QR est utile pour les matrices carrées et rectangulaires. Plusieurs algorithmes sont possibles pour la factorisation QR, comme la *transformation Householder*, la *transformation de Givens* et la *transformation de Givens rapide*.

La méthode de dé*composition* en valeurs *singulières* (SVD) décompose une matrice en produit de trois matrices : $A = USV^T$. *U* et *V* sont des matrices orthogonales. *S* est une matrice diagonale dont les valeurs diagonales sont appelées les *valeurs singulières* de *A*. Les valeurs singulières de *A* sont les racines carrées non négatives des valeurs propres de A^TA ; les colonnes de *U* et *V*, qui sont appelées les vecteurs singuliers gauche et droit, sont respectivement des vecteurs propres orthonormaux de AA^T et A^TA. La méthode SVD est utile pour résoudre des problèmes d'analyse comme le calcul du rang, de la norme, du nombre conditionnel et de la pseudo-inverse des matrices. La section suivante traite de cette dernière application.

Pseudo-inverse

La pseudo-inverse d'un scalaire σ est définie par $1/\sigma$ si $\sigma \neq 0$, et zéro dans le cas contraire. Dans le cas de scalaires, la pseudo-inverse est identique à l'inverse. Vous pouvez désormais définir la pseudo-inverse d'une matrice diagonale en transposant la matrice et en prenant ensuite la pseudo-inverse scalaire de chaque entrée. La pseudo-inverse d'une matrice réelle générale *A de type* $m \times n$, représentée par $A^\dagger$, est donnée par

$$A^\dagger = VS^\dagger U^T$$

Remarquez que la pseudo-inverse existe, que la matrice soit carrée ou rectangulaire. Si *A* est une matrice carrée non singulière, la pseudo-inverse est similaire à l'inverse d'une matrice classique. La bibliothèque d'analyse contient un VI pour calculer la pseudo-inverse de matrices réelles et complexes.

En résumé

- Une matrice peut être considérée comme un tableau à deux dimensions contenant *m* rangées et *n* colonnes. Le déterminant, le rang et le nombre conditionnel sont des attributs importants d'une matrice.
- Le nombre conditionnel d'une matrice affecte l'exactitude de la solution finale.
- Le déterminant d'une matrice diagonale, d'une matrice triangulaire supérieure ou d'une matrice triangulaire inférieure est le produit de ses éléments diagonaux.
- Deux matrices ne peuvent être multipliées entre elles que si le nombre de colonnes de la première matrice est égal au nombre de rangées de la seconde matrice.
- Un vecteur propre d'une matrice est un vecteur non nul qui ne change pas de direction lorsque la matrice lui est appliquée. Des matrices similaires ont les mêmes valeurs propres.
- L'existence d'une solution unique dans un système d'équations dépend de la singularité ou de la non-singularité d'une matrice.

19

Probabilités et statistiques

Ce chapitre explique certains concepts fondamentaux sur les probabilités et les statistiques, et vous indique comment utiliser ces concepts pour résoudre des problèmes du monde réel. Pour des exemples d'utilisation des VIs de probabilités et de statistiques, consultez la bibliothèque `examples\analysis\statxmpl.llb`.

Probabilités et statistiques

Nous vivons dans un âge de l'information où les faits et nombres occupent une place importante de la vie quotidienne. Des commentaires tels que "Le prix des magnétoscopes a baissé de 10 %", "Pierre est classé parmi les cinq premiers élèves de sa classe", "Michael Jordan a une moyenne de 30 points par match cette saison" sont courants. Ces déclarations fournissent beaucoup d'informations, mais nous pensons rarement à la façon dont ces informations ont été obtenues. Y avait-il beaucoup de données impliquées pour obtenir ces informations ? Si c'est le cas, comment les a-t-on condensées en des nombres uniques tels qu'une *baisse de 10 %* et une *moyenne de 30 points*, ou en des expressions comme *classé parmi les cinq premiers*. La réponse à toutes ces questions mène au domaine très intéressant des statistiques.

Considérez d'abord la façon dont les informations (données) sont générées. Considérez par exemple la saison de basket-ball de 1997. Michael Jordan des Chicago Bulls a joué 51 matches, marquant un total de 1568 points. Ceci inclut les 45 points qu'il a marqués dans une victoire de 103 à 100 sur l'équipe des Charlotte Hornets, y compris le panier à trois points décisif pour la victoire dans les dernières secondes du match ; ses 36 points dans une victoire de 88 à 84 sur l'équipe des Portland Trail Blazers ; une saison riche de 51 points dans une victoire de 88 à 87 sur l'équipe des New York Nicks ; 45 points, 7 rebonds, 5 assistances et 3 passes décisives dans une victoire de 102 à 97 sur l'équipe de Cleveland Cavaliers ; et ses 40 points, 6 rebonds et 6 assistances dans une victoire de 107 à 104 sur les Milwaukee Bucks. L'idée ici n'est pas que Jordan est un excellent joueur, mais qu'un simple joueur puisse générer autant de données dans une seule saison. Comment condenser toutes ces données pour qu'elles mettent en valeur les

informations essentielles et qu'il soit facile de s'en rappeler ? C'est là où le terme *statistiques* entre en jeu.

Pour condenser toutes les données, les nombres individuels doivent avoir davantage de signification pour vous aider à en tirer des conclusions. Par exemple, considérez le nombre de points que Jordan a marqués lors de différents matches. Il est difficile de se souvenir du nombre de points qu'il a marqués dans chaque match. Cependant, si vous divisez le nombre total de points que Jordan a marqués (1568) par le nombre de matches qu'il a joués (51), vous obtenez un seul nombre de 30,7 que vous pouvez appeler *moyenne* de points par match.

Supposons que vous souhaitiez classer les lancers francs de Jordan. Ceci peut s'avérer difficile en observant ses performances dans chaque match. Toutefois, vous pouvez diviser le nombre de lancers francs qu'il a marqués dans tous les matches par le nombre total de lancers francs qu'il a obtenus. Ceci donne un *pourcentage* de lancers francs de 84,4 %. Vous pouvez obtenir ce nombre pour tous les joueurs de la NBA, puis les classer. Vous pouvez ainsi condenser les informations pour tous les joueurs en des nombres uniques représentant le pourcentage de lancers francs, les points marqués par match et la moyenne des paniers à trois points. D'après ces informations, vous pouvez classer les joueurs dans différentes catégories. Vous pouvez ensuite pondérer ces différents nombres et obtenir un nombre unique pour chaque joueur. Ces nombres uniques peuvent alors nous aider à déterminer le "joueur le plus performant" de la saison. Ainsi, dans un sens large, le terme "statistiques" invoque différentes façons de récapituler des données pour en déduire des informations utiles et importantes.

Que sont au juste les probabilités ? Vous avez recherché des façons de récapituler de nombreuses données en des nombres uniques. Ces nombres aident ensuite à tirer des conclusions pour le présent. Par exemple, les statistiques sur Jordan au cours de la saison 1996 ont permis de l'élire le "joueur le plus performant" de la saison. Mais pouvons-nous tirer des conclusions pour le futur ? Pouvons-nous mesurer le degré d'exactitude de l'inférence et l'utiliser pour prendre des décisions futures ? La réponse repose dans la théorie des probabilités. Tandis que, selon les termes du non initié, il est *probable* que Jordan restera le meilleur dans les années à venir, vous pouvez utiliser différents concepts de probabilité, comme traité ultérieurement dans ce chapitre, pour émettre davantage de déclarations quantitatives.

Dans un scénario complètement différent, il y a des expériences dont les résultats ne peuvent pas être déterminés à l'avance, mais certains résultats peuvent être plus probables que d'autres. Ceci nous conduit de nouveau à

la notion de probabilités. Par exemple, si vous lancez une pièce de monnaie quelconque en l'air, quelles sont les chances pour qu'elle tombe en montrant son côté face ? La chance ou probabilité est de 50 %. Ceci signifie que si vous lancez la pièce en l'air de façon répétée, elle tombera en montrant son côté face la moitié du temps. Cela signifie-t-il que pour 10 lancers, il y aura exactement cinq résultats du côté face ? 100 lancers mèneront-ils exactement à 50 côtés face ? Probablement pas. Toutefois, pour un grand nombre d'essais, la probabilité se situera aux alentours de 0,5.

En résumé, les statistiques vous permettent de récapituler des données et de tirer des conclusions utiles pour le présent, tandis que la probabilité vous permet de mesurer le degré d'exactitude de ces conclusions et de les utiliser dans le futur.

Statistiques

Dans cette section, vous étudierez différents concepts et termes couramment utilisés en statistiques et vous apprendrez comment utiliser les VIs d'analyse en G dans différentes applications.

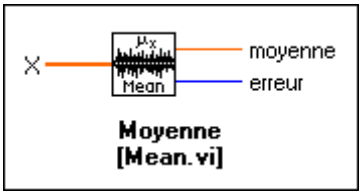
Moyenne

Considérons un ensemble de données *X* comprenant *n* échantillons x_0, x_1, x_2, x_3, … , x_{n-1}. La valeur moyenne, $\bar{x}$, est définie par la formule

$$\bar{x} = \frac{1}{n}(x_0 + x_1 + x_2 + x_3 + \ldots + x_{n-1})$$

En d'autres termes, $\bar{x}$ est la somme de toutes les valeurs d'échantillon divisée par le nombre d'échantillons. Dans l'exemple de Michael Jordan, l'ensemble de données comprend 51 échantillons. Chaque échantillon est égal au nombre de points que Jordan a marqués dans chaque match. Le total de tous ces points est 1568, qui divisé par le nombre d'échantillons (51), donne une valeur moyenne de 30,7.

Les connexions d'entrée/sortie du VI "Moyenne" (Mean.vi) sont présentées ci-dessous.



Médiane

Soit $S = \{s_0, s_1, s_2, \ldots, s_{n-1}\}$ représentant la séquence classée de l'ensemble de données X. La séquence peut être classée dans l'ordre ascendant ou descendant. La médiane de la séquence est représentée par $x_{\text{médiane}}$ et est obtenue par la formule

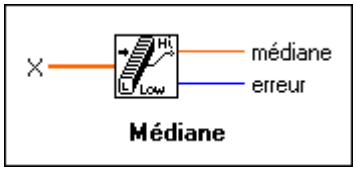
$$x_{\text{médiane}} = \begin{cases} s_i & \text{n impair} \\ 0.5(s_{k-1} + s_k) & \text{n pair} \end{cases}$$

où

$$i = \frac{n-1}{2} \text{ et } k = \frac{n}{2}$$

En d'autres termes, la médiane d'une séquence de données est la valeur *centrale* de la version classée de cette séquence. Par exemple, considérons la séquence {5, 4, 3, 2, 1} contenant cinq (nombre impair) échantillons. Cette séquence est déjà classée dans l'ordre descendant. Dans ce cas, la médiane est la valeur du milieu, 3. Considérons une séquence différente $\{1, 2, 3, 4\}$ comprenant quatre (nombre pair) échantillons. Cette séquence est déjà classée dans l'ordre ascendant. Dans ce cas, il y a deux valeurs du milieu, 2 et 3. Comme pour la formule ci-dessus, la médiane est égale à $0,5 \times (2 + 3) = 2,5$. Si un étudiant X a obtenu 4,5 points dans un test et si un autre étudiant Y a obtenu 1 dans le même test, la médiane est une quantité très utile pour émettre des déclarations qualitatives telles que "X se situe dans la moitié supérieure de la classe" ou "Y se situe dans la moitié inférieure de la classe".

Les connexions d'entrée/sortie du VI "Médiane" (Median.vi) sont présentées ci-dessous.



Variance d'échantillon

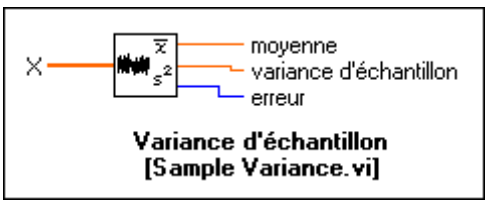
La variance d'échantillon de l'ensemble de données *X* comprenant *n* échantillons est représentée par s^2 et est définie par la formule

$$s^2 = \frac{1}{n-1}[(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2]$$

où $\bar{x}$ représente la moyenne de l'ensemble des données. La variance d'échantillon est donc égale à la somme des carrés des écarts des valeurs d'échantillon par rapport à la moyenne, divisée par *n–1*.

Remarque ***La formule précédente n'est pas définie pour n=1. Il n'y a en effet aucun intérêt à calculer la variance d'échantillon s'il n'y a qu'un échantillon dans l'ensemble des données.***

Les connexions d'entrée/sortie du VI "Variance d'échantillon" (Sample Variance.vi) sont présentées ci-dessous.

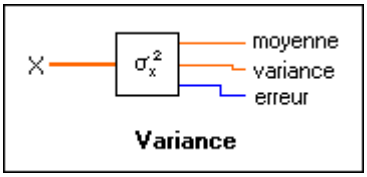


En d'autres termes, la variance d'échantillon mesure l'étendue ou la dispersion des valeurs d'échantillon. Si l'ensemble des données représente les points marqués par un joueur lors de différents matches, la variance d'échantillon peut être utilisée pour mesurer la régularité du joueur. Elle est toujours positive, sauf lorsque toutes les valeurs d'échantillon sont égales, et sont donc égales à la moyenne.

Il existe un autre type de variance appelé "variance de population". La formule pour calculer la variance de population est similaire à la formule

de calcul d'une variance d'échantillon, sauf pour le (*n–1*) dans le dénominateur, qui est remplacé par *n*.

Les connexions d'entrée/sortie du VI "Variance" (Variance.vi) sont présentées ci-dessous.



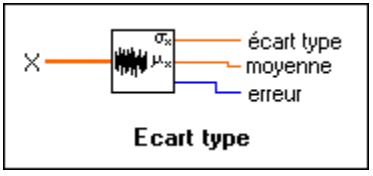
Le VI "Variance d'échantillon" (Sample Variance.vi) calcule la variance d'échantillon, tandis que le VI "Variance" (Variance.vi) calcule la variance de population. Les statisticiens et les mathématiciens préfèrent utiliser la variance de population, mais les ingénieurs optent pour la variance d'échantillon. Cela n'a vraiment pas d'importance pour des valeurs élevées de *n*, telles que $n \geq 30$.

Remarque ***Utilisez le type de VI qui convient à votre application.***

Ecart-type

La racine carrée positive de la variance d'échantillon s^2 est représentée par s et est appelée l'écart-type de l'échantillon.

Les connexions d'entrée/sortie du VI "Ecart-type" (Standard Deviation.vi) sont présentées ci-dessous.



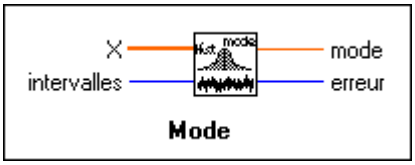
Mode

Le mode d'un échantillon est la valeur d'échantillon qui apparaît le plus souvent dans l'échantillon. Par exemple, si la séquence d'entrée *X* est

$$X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 7\}$$

alors le mode de *X* est 4, car c'est la valeur qui apparaît le plus souvent dans *X*.

Les connexions d'entrée/sortie du VI "Mode" (Mode.vi) sont présentées ci-dessous.



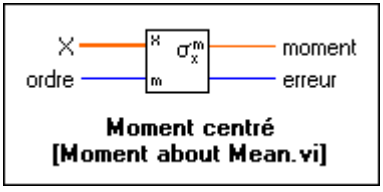
Moment centré

Si X représente la séquence d'entrée contenant n nombre d'éléments, et si $\bar{x}$ est la moyenne de cette séquence, alors le moment du $m^{\text{ème}}$ ordre peut être calculé grâce à la formule

$$\sigma_x^{\,m} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^m$$

En d'autres termes, le moment centré est une mesure de l'écart des éléments de la séquence par rapport à la moyenne. Remarquez que pour $m = 2$, le moment centré est égal à la variance de la population.

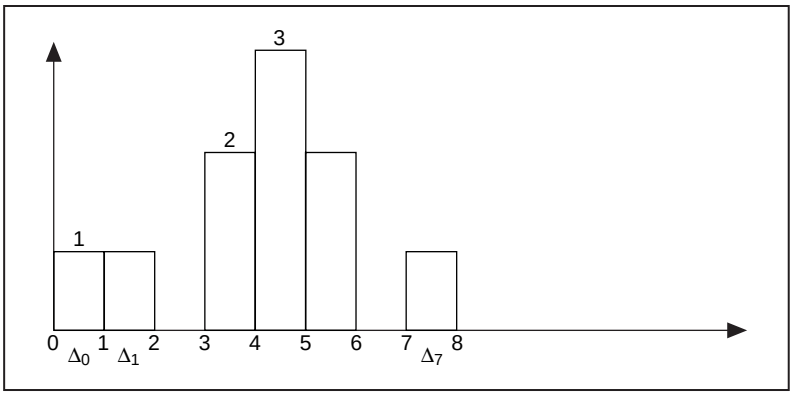
Les connexions d'entrée/sortie du VI "Moment centré" (Moment about Mean.vi) sont présentées ci-dessous.



Histogramme

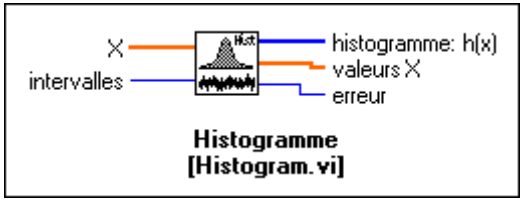
Ce chapitre a présenté jusque-là différentes façons d'extraire les caractéristiques essentielles d'un ensemble de données. Les données sont généralement enregistrées sous forme de tableaux, souvent jugés difficiles à interpréter. Un affichage graphique des données permet de mieux les comprendre. L'histogramme est ainsi une méthode graphique destinée à afficher des données et à récapituler des informations clés. Considérons une séquence de données $X = \{0, 1, 3, 3, 4, 4, 4, 5, 5, 8\}$. Divisons l'intervalle total de valeurs en huit intervalles. Ces intervalles sont 0 – 1, 1 – 2, 2 – 3, ..., 7 – 8. L'histogramme de la séquence X présente alors le

nombre des échantillons de données se trouvant dans cet intervalle, sans inclure la limite supérieure.



La figure ci-dessus indique qu'il y a un échantillon de données dans l'intervalle 0 – 1 et dans l'intervalle 1 – 2. Il n'y a aucun échantillon dans l'intervalle 2 – 3. Il y a deux échantillons dans l'intervalle 3 – 4, et trois échantillons dans l'intervalle 4 – 5. Examinez la séquence de données *X* ci-dessus et assurez-vous que vous comprenez ce concept.

Il y a différentes façons de calculer les données d'un histogramme. Vous verrez comment le VI "Histogramme" (Histogram.vi) fonctionne.



Comme indiqué ci-dessus, les entrées de ce VI sont la **séquence d'entrée** *X* et le nombre d'intervalles *m*. Le VI obtient un histogramme comme une fonction de x, défini par **Histogramme:h(x)** indiqué ci-après. Le VI balaie *X* pour déterminer sa gamme de valeurs, puis établit la largeur d'intervalle, Δx , d'après la valeur spécifiée de *m*

$$\Delta x = \frac{max - min}{m}$$

où *max* est la valeur maximum trouvée dans *X*, où *min* est la valeur minimum trouvée dans *X*, et où *m* est le nombre d'intervalles précisé.

Si

$$m = 8$$

alors

$$\Delta x = \frac{8-0}{8} = 1$$

Soit χ représentant la séquence de sortie Valeurs *X*. L'histogramme est une fonction de *X*. Ce VI évalue les éléments de χ en utilisant

$$\chi_i = min + 0,5\Delta x + i\Delta x \qquad pour \qquad i = 0, 1, 2, \ldots, m-1$$

Pour cet exemple,

$$\chi_0 = 0.5, \chi_1 = 1.5, \ldots, \chi_7 = 7,5$$

Le VI définit alors le $i^{ème}$ intervalle de la gamme des valeurs, de $\chi_i - (0,5\Delta x)$ à $\chi_i + 0,5\Delta x$ (non inclus),

$$\Delta_i = [(\chi_i - (0,5\Delta x)), (\chi_i + 0,5\Delta x)], pour \qquad i = 0, 1, 2, \ldots, m-1$$

et définit la fonction $y_i(x) = 1$ pour *x* appartenant à Δ_i et zéro pour tous les autres cas. La fonction est égale à 1 si la valeur de *x* s'inscrit dans l'intervalle précisé, sans inclure la limite. Sinon, la fonction est égale à zéro. Remarquez que l'intervalle est centré sur χ_i et que sa largeur est Δ_x. Si une valeur est égale au maximum (max), elle est considérée comme appartenant au dernier intervalle.

Pour notre exemple,

$$\Delta_0 = [0, 1], \Delta_1 = [1, 2], \ldots, \Delta_7 = [7, 8]$$

avec le cas particulier

$$y_0(0) = 1$$

et

$$y_0(1) = y_0(3) = y_0(4) = y_0(5) = y_0(8) = 0$$

En dernier lieu, le VI évalue la séquence de l'histogramme *H* en utilisant

$$h_i = \sum_{j=0}^{n-1} y_i(x_j) \qquad pour \qquad i = 0, 1, 2, \ldots, m-1$$

où h_i représente les éléments de la séquence de sortie *Histogramme: h(X)* et *n* est le nombre d'éléments de la séquence d'entrée *X*. Pour cet exemple, $h_0 = 1, h_4 = 3, \ldots, h_7 = 1$.

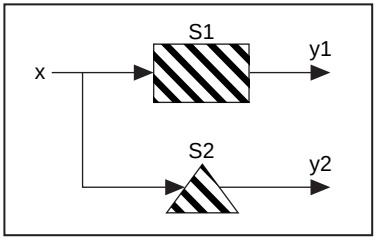
La bibliothèque d'analyse en G possède également un VI "Histogramme général" (General Histogram.vi), plus avancé que le VI "Histogramme" (Histogram.vi). Pour plus de détails, consultez la *Référence d'analyse en ligne*.

Erreur quadratique moyenne

Si X et Y représentent deux séquences d'entrée, alors l'erreur quadratique moyenne est la moyenne de la somme du carré de la différence entre les éléments correspondants des deux séquences d'entrée. La formule suivante est utilisée pour déterminer l'erreur quadratique moyenne (mse).

$$mse = \frac{1}{n}\sum_{i=0}^{n-1} (x_i - y_i)^2$$

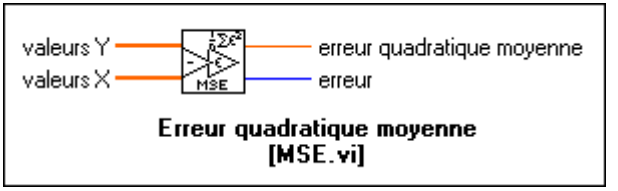
où *n* est le nombre de points.



Considérons un signal numérique x fourni à un système, S1. La sortie de ce système est y1. Soit un nouveau système, S2, qui doit théoriquement générer le même résultat que S1, mais avec un temps de réponse deux fois plus rapide. Avant de remplacer l'ancien système, vous souhaitez être absolument certain que la réponse de sortie des deux systèmes est la même. Si les séquences y1 et y2 sont très importantes, il est difficile de comparer

chaque élément dans les séquences. Dans un tel scénario, vous pouvez utiliser le VI “Erreur quadratique moyenne” (MSE.vi) pour calculer l’erreur quadratique moyenne des deux séquences y1 et y2. Si l’erreur quadratique moyenne est inférieure aux valeurs de tolérance permises, alors le système S1 peut être remplacé sans problème par le nouveau système S2.

Les connexions d’entrée/sortie du VI “Erreur quadratique moyenne” (MSE.vi) sont indiquées ci-dessous.



Moyenne quadratique

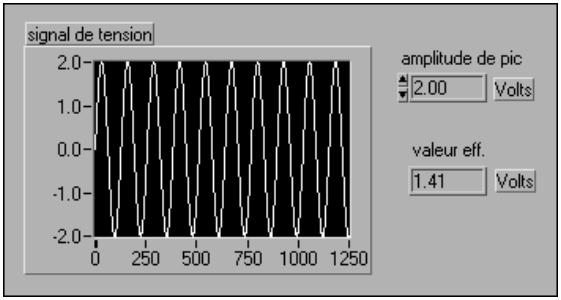
La moyenne quadratique Ψ_x d’une séquence X est la racine carrée positive de la moyenne du carré de la séquence d’entrée. En d’autres termes, vous pouvez élever au carré la séquence d’entrée, calculer la moyenne de la séquence mise au carré, puis prendre la racine carrée de cette quantité. La formule utilisée pour calculer la moyenne quadratique est

$$\Psi_x = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2}$$

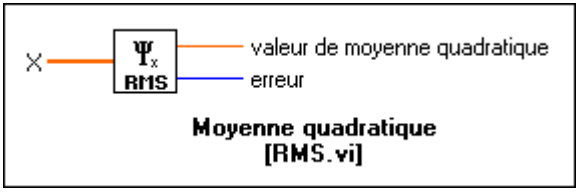
où n représente le nombre d’éléments de X.

La moyenne quadratique est une quantité largement utilisée dans le cas de signaux analogiques. Pour un signal de tension sinusoïdal, si V_p est l’amplitude de crête du signal, alors la tension de moyenne quadratique V_{eff} est donnée par $\frac{V_p}{\sqrt{2}}$.

La figure suivante représente un signal de tension d'amplitude maximum égale à 2 V, avec une valeur de moyenne quadratique de $\sqrt{2} \approx (1,41)\text{V}$ calculée en utilisant la bibliothèque d'analyse.



Les connexions d'entrée/sortie du VI "Moyenne quadratique" (RMS.vi) sont indiquées ci-dessous.



Probabilités

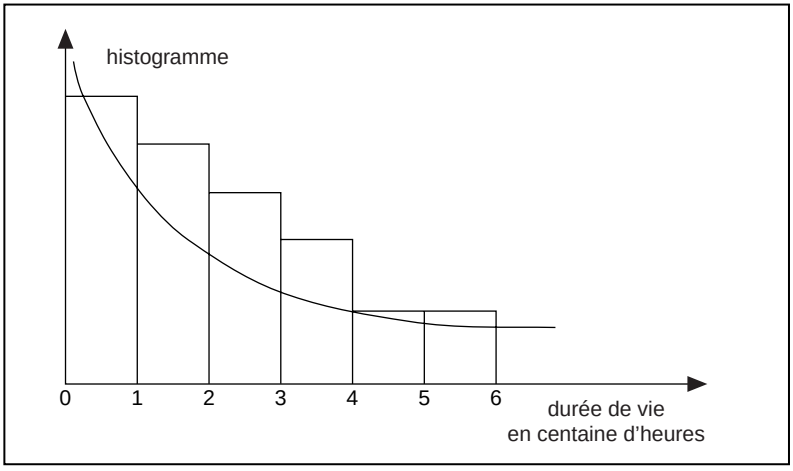
Dans toute expérience aléatoire, il y a toujours la possibilité qu'un événement particulier se produise ou non. Un nombre entre 0 et 1 est affecté pour mesurer la chance, ou probabilité, qu'un événement particulier se produise. Si vous êtes absolument certain que l'événement va se produire, sa probabilité est de 100 % ou de 1, mais si vous êtes certain que l'événement ne va pas se produire, sa probabilité est égale à 0.

Considérons un exemple simple. Si vous faites rouler un seul dé (non faussé), six événements peuvent se produire : vous pouvez obtenir un 1, un 2, un 3, un 4, un 5, ou un 6. Quelle est la probabilité d'obtenir un 2 ? Cette probabilité est égale à 1 sur 6, c'est-à-dire 0,16666. Vous pouvez définir la probabilité en ces termes simples : la probabilité qu'un événement A se produise est le rapport du nombre de résultats favorables à A sur le nombre total des résultats également vraisemblables.

Variables aléatoires

De nombreuses expériences génèrent des résultats que vous pouvez interpréter grâce à des nombres réels. De tels exemples peuvent être le nombre de voitures s'arrêtant à un panneau de stop chaque jour, le nombre de votants préférant le candidat A, et le nombre d'accidents survenant à une intersection particulière. Les valeurs des résultats numériques peuvent changer d'une expérience à l'autre et sont appelées "variables aléatoires". Les variables aléatoires peuvent être discrètes (si elles ne peuvent prendre qu'un nombre fini de valeurs possibles) ou continues. Comme exemple de ces dernières, le poids des patients d'une clinique peut se situer n'importe où entre 35 et 200 kg. De telles variables aléatoires peuvent prendre n'importe quelle valeur dans un intervalle de nombres réels. Etant donné une telle situation, supposons que vous souhaitiez trouver la probabilité de rencontrer un patient pesant exactement 65,50 kilos. Vous apprendrez comment calculer cette probabilité dans la section suivante, à l'aide d'un exemple.

Considérons une expérience destinée à mesurer la durée de vie *x* de 50 batteries d'un certain type. Ces batteries sont sélectionnées parmi une population importante de batteries identiques. L'histogramme des données observées est présenté ci-dessous.



Cette figure indique que la plupart des durées de vie se situent entre zéro et 100 heures, et que les valeurs de l'histogramme diminuent régulièrement à mesure que les durées de vie sont plus longues.

Vous pouvez estimer approximativement l'histogramme indiqué ci-dessus grâce à une courbe exponentielle décroissante. Vous pouvez prendre cette

fonction comme un modèle mathématique représentant le comportement de l'échantillon des données. Si vous souhaitez connaître la probabilité qu'une batterie sélectionnée de façon aléatoire dure plus longtemps que quatre cents heures, cette valeur peut être estimée par la surface de la zone se trouvant sous la courbe, à droite de la valeur 4. Une telle fonction qui modélise l'histogramme de la variable aléatoire est appelée la *densité de probabilité*.

Pour récapituler toutes les informations ci-dessus dans une définition, une variable aléatoire *X* est dite *continue* si elle peut prendre un nombre infini de valeurs possibles associées aux intervalles de nombres réels, et il existe une fonction *f(x)*, appelée *densité de probabilité*, telle que

1. $(x) \geq 0$ pour tout x

2. $\int_{-\infty}^{\infty} f(x)dx = 1$

3. $P(a \leq X \leq b) = \int_{a}^{b} f(x)dx$

Remarquez dans l'équation (3) ci-dessus, que pour une valeur spécifique de la variable aléatoire, *a* :

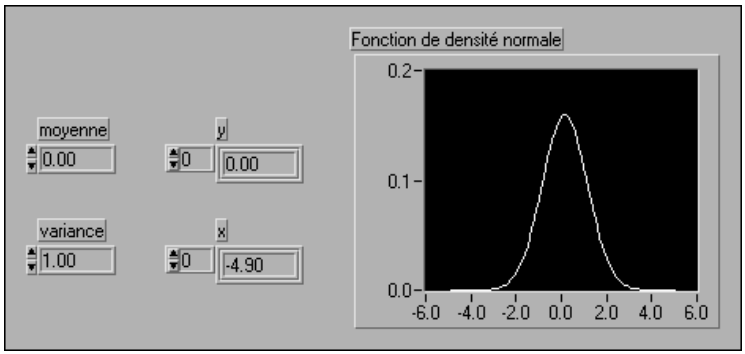
$$\text{X=a}, P(X = a) = \int_{a}^{a} f(x)dx = 0$$

Il n'est pas surprenant d'affecter une probabilité de zéro à toute valeur spécifique, car la variable aléatoire peut prendre un nombre infini de valeurs possibles. La chance pour que cette variable prenne une valeur spécifique $X = a$ est donc extrêmement faible.

Dans l'exemple utilisé précédemment, la fonction exponentielle servait de modèle à la densité de probabilité. Il y a de nombreux autres choix de fonction, notamment la fonction de distribution normale, présentée ci-dessous.

Distribution normale

La distribution normale est l'une des distributions de probabilité continue les plus couramment utilisées. Cette fonction de distribution a la forme d'une cloche symétrique, comme indiqué dans l'illustration suivante.



La courbe est centrée autour de la valeur moyenne $\bar{x} = 0$, et son étendue est mesurée par la variance $s^2 = 1$. Ces deux paramètres déterminent entièrement la forme et l'emplacement de la fonction de densité normale, dont la forme fonctionnelle est donnée par

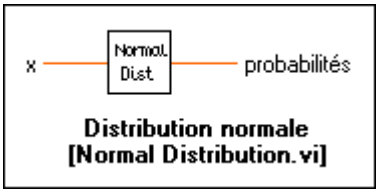
$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{-(x-\bar{x})^2/(2s^2)}$$

Supposons qu'une variable aléatoire Z possède une distribution normale avec une moyenne égale à zéro et une variance égale à 1. Cette variable aléatoire possède une *distribution normale standard.*

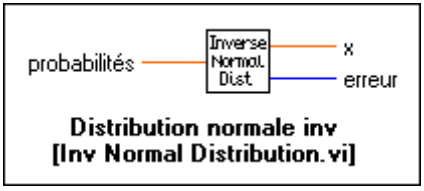
Le VI "Distribution normale" (Normal Distribution.vi) d'analyse en G calcule la probabilité unilatérale, *p*, d'une variable aléatoire distribuée normalement *x*.

$$p = Prob(X \leq x)$$

où *X* est une distribution normale standard avec une valeur moyenne égale à zéro et une variance égale à 1, où *p* est la probabilité et *x*, la valeur.



Supposons que vous meniez une expérience dans laquelle vous mesurez la taille d'hommes adultes. Vous menez cette expérience sur 1000 hommes choisis de façon aléatoire et vous obtenez un ensemble de données E. La distribution de l'histogramme révèle de nombreuses mesures très regroupées autour d'une hauteur moyenne, avec relativement peu d'hommes très petits ou très grands dans la population. L'histogramme peut donc très bien subir une approximation par une distribution normale. Supposons maintenant que, dans un groupe différent de 1000 hommes choisis de façon aléatoire, vous souhaitiez déterminer la probabilité pour laquelle la taille d'un homme est supérieure ou égale à 1,70 m. Vous pouvez utiliser le VI "D**istribution normale**" (Normal Distribution.vi) pour déterminer cette probabilité. Définissez l'entrée $x = 170$. Le choix de la densité de probabilité est donc fondamental pour obtenir une valeur de probabilité correcte.



Le VI "Distribution normale inverse" (Inv Normal Distribution.vi) effectue exactement la fonction opposée. Soit une probabilité *p*, le VI détermine les valeurs *x* qui ont une chance de se trouver dans un échantillon distribué normalement. Par exemple, vous souhaiterez peut-être déterminer les hauteurs qui ont 60 % de chance de se situer dans un ensemble de données choisies de façon aléatoire.

Comme mentionné précédemment, il y a différents choix possibles pour la densité de probabilité. Les choix les plus connus et les plus répandus sont la distribution en chi2, la distribution de type F et la distribution de type T. Pour plus d'informations sur ces distributions, consultez l'Annexe A, *Références d'analyse*. La bibliothèque d'analyse en G contient des VIs qui

calculent la probabilité unilatérale de ces différents types de distributions. En outre, elle contient des VIs qui effectuent l'opération inverse.

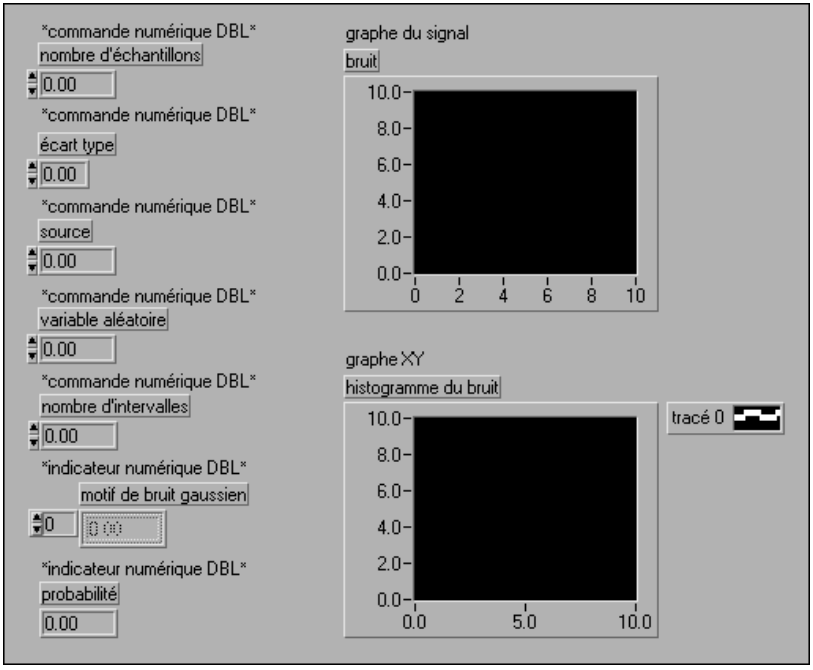
Exercice 19-1. Utiliser le VI "Distribution normale" (Normal Distribution.vi)

Votre objectif est de comprendre les concepts clés des probabilités.

Dans cet exercice, vous allez d'abord générer un échantillon de données avec une distribution normale standard, puis utiliser le VI "D**istribution normale"** (Normal Distribution.vi) pour vérifier la probabilité d'une variable aléatoire *x*.

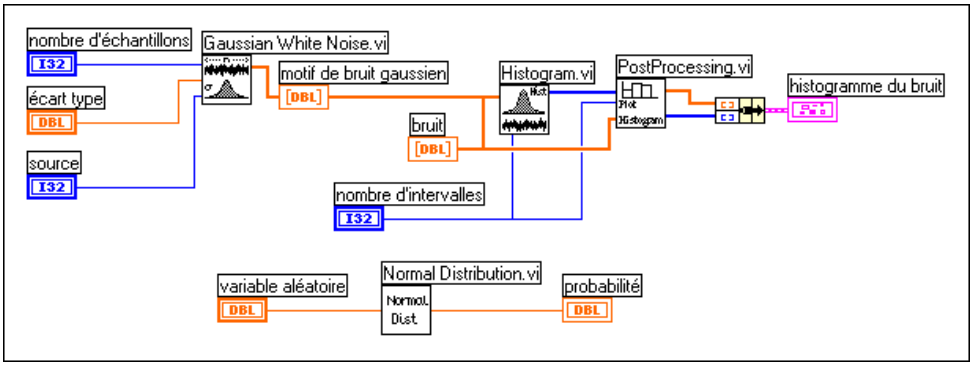
Face-avant

1. Construisez la face-avant comme indiqué dans la figure suivante. Le Tracé du bruit est représenté par un graphe oscilloscopique, tandis que l'Histogramme du bruit est un graphe XY.



Diagramme

2. Construisez le diagramme comme indiqué dans l'illustration suivante. Le Bruit blanc gaussien génère un motif à distribution gaussienne, avec une valeur moyenne égale à 0 et un écart-type défini par l'utilisateur, en utilisant l'écart-type de l'entrée. L'entrée Nombre d'échantillons (Number of samples) représente le nombre d'échantillons du Motif bruit gaussien. L'entrée Source (Seed) correspond à la valeur source utilisée pour générer le bruit aléatoire. Connectez le Motif bruit gaussien au Tracé du bruit du graphe (Noise Plot).



VI "Bruit blanc gaussien" (Gaussian White Noise.vi) (sous-palette **Analyse»Génération de signal**). Dans cet exercice, cette fonction génère un motif Bruit blanc gaussien.

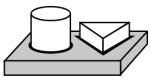
VI "Histogramme" (Histogram.vi) (sous-palette **Analyse»Probabilités et statistiques**). Dans cet exercice, cette fonction calcule l'histogramme du Motif bruit gaussien.

VI "Distribution normale" (Normal Distribution.vi) (sous-palette **Analyse»Probabilités et statistiques**). Dans cet exercice, cette fonction calcule la probabilité unilatérale de la variable aléatoire distribuée normalement **Variable aléatoire (Random Variable)**.

3. Calculez l'histogramme du Motif bruit gaussien en utilisant le VI "Histogramme" (Histogram.vi) utilisé dans l'exercice précédent.
4. Comme mentionné précédemment, effectuez un traitement ultérieur pour tracer l'histogramme d'une façon différente. Sélectionnez le VI "Après-traitement" (PostProcessing.vi) dans le répertoire `LabVIEW\Activity`.
5. Assemblez les sorties de ce VI et connectez le cluster à l'Histogramme du bruit.
6. Sélectionnez le VI "Distribution normale" (Normal Distribution.vi). Connectez la commande de variable aléatoire au terminal d'entrée, puis connectez la sortie à l'indicateur de probabilité.
7. Revenez sur la face-avant. Faites passer le nombre d'échantillons sur la valeur 2048, définissez l'écart-type sur la valeur 1, la source sur la valeur 2 et le nombre d'intervalles sur la valeur 10. Exécutez le VI.
8. Vous pouvez voir le Bruit blanc gaussien sur le graphe de Tracé du bruit. Il est difficile de tirer beaucoup de conclusions de ce tracé. Toutefois, le tracé de l'histogramme du même motif de bruit fournit de nombreuses informations, notamment que la plupart des échantillons sont centrés autour de la valeur moyenne de zéro. A partir de cet histogramme, vous pouvez faire l'approximation de ce motif de bruit par une fonction de distribution normale (distribution gaussienne). Comme la valeur moyenne est zéro et que vous définissez l'écart-type égal à la valeur 1, la densité de probabilité est réellement une distribution normale standard.

Remarque ***Il est très important de choisir avec soin une fonction de distribution adéquate pour l'approximation de vos données. Dans cet exemple, vous avez tracé l'histogramme pour prendre cette décision. Bien souvent, vous pouvez prendre une décision judicieuse simplement d'après une connaissance préalable du comportement et des caractéristiques de l'échantillon des données.***

9. Revenez sur la face-avant et entrez une valeur pour la Variable aléatoire. Ce VI va calculer la probabilité unilatérale de cette variable aléatoire distribuée normalement. Souvenez-vous que vous avez supposé que la variable était distribuée normalement après avoir observé l'histogramme
10. Enregistrez le VI sous le nom `Probability.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l'exercice 19-1..

En résumé

- Différents concepts de statistiques et probabilités permettent de déchiffrer des informations et des données afin de prendre des décisions bien fondées.
- La moyenne, la médiane, la variance d'échantillon et le mode représentent quelques-unes des techniques statistiques pour émettre des inférences à partir de l'échantillon d'une population.
- Les histogrammes sont largement utilisés car il s'agit d'une méthode simple mais informative pour afficher des données.
- En utilisant la théorie des probabilités, vous pouvez émettre des inférences à partir d'un échantillon de population, puis mesurer le degré d'exactitude de ces inférences.

Partie IV

Communication inter-applications et sur réseau

Cette section contient des informations de base sur la communication inter-applications et sur réseau.

La Partie IV, *Communication inter-applications et sur réseau*, contient les chapitres suivants.

- Le chapitre 20, *Introduction à la communication*, présente la façon dont LabVIEW gère la communication inter-applications et sur réseau.
- Le chapitre 21, *TCP et UDP*, décrit le protocole Internet (IP), le protocole de datagramme utilisateur (UPD), le protocole de gestion de transmission (TCP), les adresses Internet et les exemples d'applications client/serveur TCP.
- Le chapitre 22, *Support d'ActiveX*, explique comment utiliser LabVIEW comme un serveur et un client ActiveX. ActiveX est identique à la communication Automation OLE.
- Le chapitre 23, *Utilisation du protocole DDE*, décrit les VIs LabVIEW pour l'échange dynamique de données (DDE) pour Windows 3.1, Windows 95 et Windows NT. Ces VIs exécutent des fonctions DDE pour le partage de données avec d'autres applications qui acceptent les connexions DDE.
- Le chapitre 24, *AppleEvents*, décrit AppleEvents, une forme de communication inter-applications uniquement pour Macintosh (IAC), via laquelle des applications Macintosh peuvent communiquer.
- Le chapitre 25, *Communication programme à programme*, décrit la communication programme à programme (PPC), une forme bas niveau de communication inter-applications Apple (IAC) via laquelle des applications Macintosh envoient et reçoivent des blocs de données.

20

Introduction à la communication

Ce chapitre présente la façon dont LabVIEW gère la mise en réseau et les communications inter-applications.

Présentation de la communication dans LabVIEW

Pour les besoins de cette discussion, le terme *mise en réseau* fait référence à la communication entre plusieurs processus (pouvant s'effectuer sur des ordinateurs séparés). Cette communication s'effectue généralement sur un réseau matériel, tel que Ethernet ou LocalTalk.

L'une des utilisations principales de la mise en réseau dans des applications logicielles consiste, pour une ou plusieurs applications, à utiliser les services d'une autre application. Par exemple, l'application fournissant les services (le serveur) peut être une application de regroupement de données s'exécutant sur un ordinateur dédié ou un programme de base de données fournissant des informations à d'autres applications.

Dans ce chapitre, nous vous présenterons la terminologie relative à la mise en réseau et à la communication, ainsi que les applications de programmation mises en réseau.

Introduction aux protocoles de communication

Pour qu'une communication puisse se produire entre des processus, ceux-ci doivent utiliser un langage de communication commun, appelé *protocole*.

Un protocole de communication vous permet de préciser les données que vous souhaitez envoyer ou recevoir, ainsi que la position de la destination ou de la source. Vous n'avez pas à vous soucier de la façon dont les données sont transmises. Le protocole traduit vos commandes en données acceptées par les drivers de réseau. Ces drivers prennent ensuite en charge le transfert des données dans le réseau, comme il convient.

Plusieurs protocoles de mise en réseau sont devenus des standards pour les communications. En général, un protocole n'est pas compatible avec un

autre. Ainsi, dans les applications de communication, l'une des premières choses que vous devez faire est de décider du protocole à utiliser. Si vous souhaitez communiquer avec une application existante sur le marché, vous devez travailler dans les protocoles supportés par cette application.

Lorsque vous écrivez l'application, vous avez plus de latitude quant au choix du protocole. Les facteurs pour le choix du protocole incluent le type d'ordinateur sur lesquels les processus seront exécutés, le type de réseau matériel disponible et la complexité de la communication nécessaire à votre application.

Plusieurs protocoles sont intégrés à LabVIEW, et certains d'entre eux sont spécifiques à certains types d'ordinateur. LabVIEW utilise les protocoles suivants pour communiquer entre les ordinateurs :

- **TCP** : disponible sur tous les ordinateurs
- **UDP** : disponible sur tous les ordinateurs
- **DDE** : disponible sur PC, pour la communication entre des applications Windows
- **ActiveX** : disponible avec Windows 95 et Windows NT
- **AppleEvents** : disponible sur Macintosh, pour envoyer des messages entre des applications Macintosh
- **PPC** : disponible sur Macintosh, pour envoyer et recevoir des données entre des applications Macintosh

Les protocoles sont tous différents, particulièrement dans leur manière de faire référence à la position sur le réseau d'une application à distance. Ils sont incompatibles entre eux, aussi si vous souhaitez communiquer entre un Macintosh et un PC, vous devez utiliser un protocole compatible avec les deux, tel que le TCP.

D'autres options de communication LabVIEW incluent les éléments suivants :

- **VI "Exéc. système" (System Exec.vi)** : vous permet d'exécuter une commande de niveau système. Il y a en fait deux VIs "Exéc. système", l'un pour toutes les versions Windows, l'autre pour Sun et HP-UX
- **Canaux de communication** : disponible uniquement sur UNIX
- **HiQ** : disponible uniquement sur Macintosh et PC

Partage de fichiers et protocoles de communication

Avant d'entrer dans de plus amples détails sur les protocoles de communication, prenez le temps de voir si une autre approche s'avère plus appropriée pour votre application. Par exemple, considérez une application où un système dédié acquiert des données, celles-ci étant enregistrées sur un ordinateur différent.

Vous pouvez écrire une application qui utilise des protocoles de mise en réseau pour envoyer les données depuis un ordinateur d'acquisition vers l'ordinateur de lecture-écriture, où une application séparée recueille les données et les enregistre sur un disque.

Une méthode plus simple consiste à utiliser les fonctions de partage de fichiers disponibles sur la plupart des ordinateurs mis en réseau. Avec le partage de fichiers, les drivers du système d'exploitation vous permettent de vous connecter à d'autres ordinateurs. Le stockage de disque de l'ordinateur à distance est traité comme une extension de votre propre stockage de disque. Une fois que vous avez connecté deux systèmes, le partage de fichiers rend (d'habitude) cette connexion transparente, de sorte que toute application peut écrire sur le disque à distance comme si elle était connectée localement. Le partage de fichiers constitue souvent la méthode la plus simple pour transférer des données entre des ordinateurs.

Modèle client/serveur

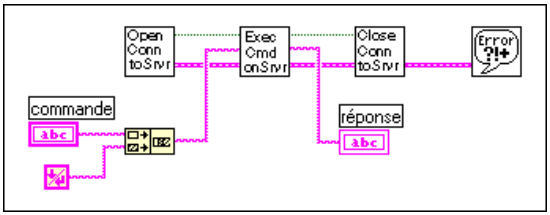
Le modèle client/serveur est un modèle courant pour les applications mises en réseau. Dans le modèle client/serveur, un ensemble de processus (clients) requiert des services d'un autre ensemble de processus (serveurs).

Par exemple, dans votre application, vous pouvez configurer un ordinateur dédié à acquérir des mesures du monde réel. L'ordinateur se comporte comme un serveur lorsqu'il fournit des données sur demande à d'autres ordinateurs. Il se comporte comme un client lorsqu'il demande à une autre application, un programme de base de données par exemple, d'enregistrer les données qu'il obtient.

Dans LabVIEW, vous pouvez utiliser les applications client et serveur avec tous les protocoles, à l'exception d'AppleEvents de Macintosh. Vous pouvez utiliser AppleEvents pour envoyer des commandes à d'autres applications. Vous ne pouvez pas configurer un serveur de commande dans LabVIEW avec AppleEvents. Si vous avez besoin de capacités serveur sur Macintosh, utilisez les protocoles TCP, UDP ou PPC.

Modèle général pour un client

Le diagramme suivant représente le modèle simplifié d'un client dans LabVIEW.



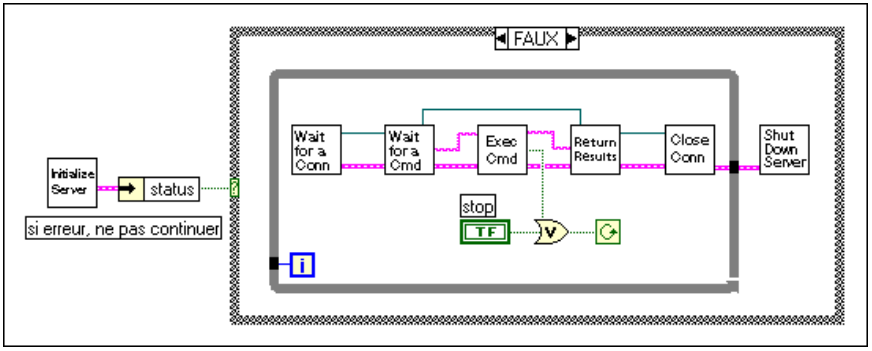
Dans le diagramme précédent, LabVIEW ouvre d'abord une connexion vers un serveur. LabVIEW envoie ensuite une commande au serveur, obtient une réponse à cette commande, puis ferme la connexion vers le serveur. Enfin, il rapporte toutes les erreurs qui se sont produites pendant le processus de communication.

Pour obtenir de meilleures performances, vous pouvez traiter plusieurs commandes après avoir établi la connexion. Après l'exécution de ces commandes, vous pouvez fermer la connexion.

Cette structure de diagramme de base sert de modèle (utilisé ailleurs dans ce manuel) pour implémenter un protocole donné dans LabVIEW.

Modèle général pour un serveur

Le diagramme suivant représente le modèle simplifié d'un serveur dans LabVIEW.

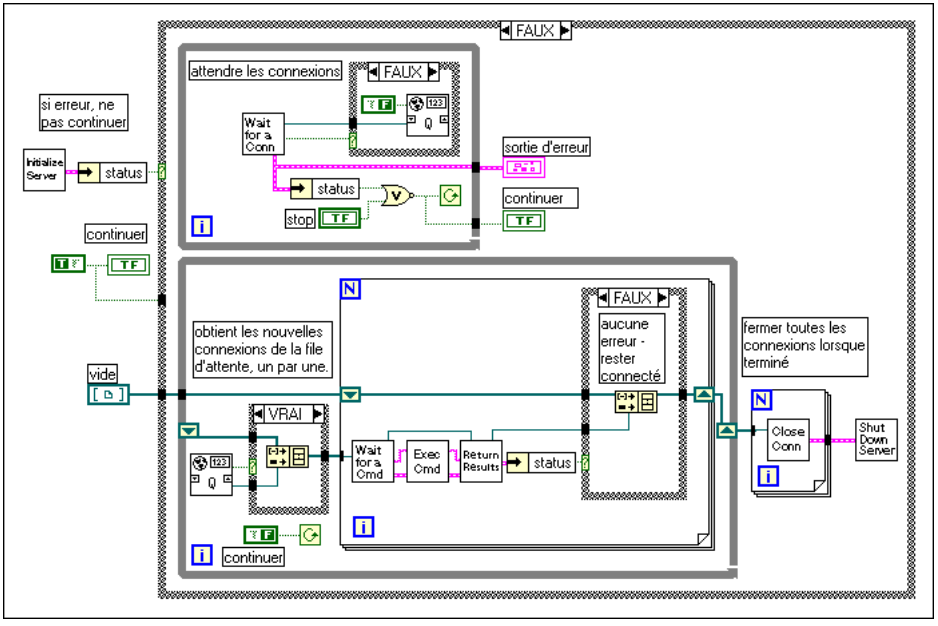


Dans le diagramme précédent, LabVIEW initialise d'abord le serveur. Si l'initialisation est réussie, LabVIEW se met en boucle, où il attend une

connexion. Une fois la connexion établie, LabVIEW attend de recevoir une commande. LabVIEW exécute la commande et retourne les résultats, puis la connexion est fermée. LabVIEW répète entièrement ce processus jusqu'à ce qu'il soit arrêté localement lorsqu'un utilisateur appuie sur un bouton Stop de la face-avant ou lorsqu'il reçoit à distance la commande d'arrêter le VI.

Ce VI ne rapporte pas les erreurs. Il peut renvoyer une réponse indiquant qu'une commande est invalide, mais il n'affiche pas une boîte de dialogue lorsqu'une erreur se produit. Un serveur pouvant être laissé sans surveillance, considérez avec soin la façon dont le serveur doit gérer les erreurs. Vous ne souhaiterez probablement pas voir s'afficher une boîte de dialogue, car ceci implique une interaction de l'utilisateur sur le serveur (quelqu'un devra appuyer sur le bouton OK). Cependant, vous souhaiterez peut-être que LabVIEW génère un rapport de transferts et d'erreurs dans un fichier ou dans une chaîne de caractères.

Vous pouvez améliorer les performances en laissant la connexion ouverte afin de pouvoir recevoir plusieurs commandes, mais cette action empêche d'autres clients de se connecter tant que le client actuel ne s'est pas déconnecté. Si le protocole supporte plusieurs connexions simultanées, vous pouvez restructurer LabVIEW afin de gérer plusieurs clients en même temps, comme indiqué dans le diagramme suivant.



Le diagramme précédent tire profit des capacités multi-tâches de LabVIEW pour exécuter deux boucles simultanément. Une boucle attend continuellement une connexion. Lorsqu'une connexion est reçue, elle est ajoutée à une file d'attente. L'autre boucle vérifie chaque connexion ouverte et exécute toutes les commandes reçues. Si une erreur se produit sur l'une des connexions, la connexion est déconnectée. Lorsque l'utilisateur abandonne le serveur, toutes les connexions ouvertes sont fermées. Cette structure de diagramme de base sert de modèle (utilisé ailleurs dans ce manuel) pour implémenter un protocole donné dans LabVIEW.

21

TCP et UDP

Ce chapitre traite du protocole Internet (Internet Protocol [IP]), du protocole de datagramme utilisateur (User Datagram Protocol [UPD]), du protocole de gestion de transmission (Transmission Control Protocol [TCP]) et des adresses Internet. Vous y trouverez également des exemples d'applications clients/serveurs TCP.

Présentation générale

TCP/IP est une suite de protocoles de communication, développés à l'origine pour l'Agence de la défense pour les projets de recherche avancés (Defense Advanced Research Projects Agency [DARPA]). Depuis son développement, la suite TCP/IP a été largement acceptée et est disponible sur de nombreux systèmes informatiques.

Le terme "TCP/IP" vient du nom des deux protocoles les plus connus de la suite, TCP (Transmission Control Protocol) et IP (Internet Protocol). Les protocoles TCP, IP et UDP (User Datagram Protocol) représentent les outils de base de la communication sur réseau.

TCP/IP permet la communication sur des réseaux simples ou sur plusieurs réseaux interconnectés (Internet). Les réseaux individuels peuvent être séparés par de grandes distances géographiques. TCP/IP transmet les données d'un réseau ou d'un ordinateur Internet à un autre. Comme TCP/IP est disponible sur la plupart des ordinateurs, il peut transférer des informations entre différents systèmes.

IP (Internet Protocol) transmet les données dans le réseau. Ce protocole de bas niveau prend des données d'une taille limitée et les envoie dans le réseau sous la forme d'un *datagramme*. Comme il n'est pas garanti que les données arriveront à l'autre extrémité, IP est rarement utilisé directement par des applications. De plus, lorsque vous envoyez plusieurs datagrammes, ils arrivent parfois dans le désordre ou sont délivrés plus d'une fois, selon la façon dont le transfert réseau se produit. Le protocole UDP, construit sur IP, présente les mêmes problèmes.

TCP est un protocole de niveau supérieur qui utilise IP pour transférer les données. TCP décompose les données en composantes pouvant être gérées par IP. TCP permet également de détecter les erreurs et d'assurer l'arrivée en ordre des données, sans qu'elles ne soient dupliquées. Ce sont les raisons pour lesquelles TCP constitue généralement le meilleur choix pour les applications de réseau.

LabVIEW et TCP/IP

Vous pouvez utiliser la suite de protocoles TCP/IP avec LabVIEW sur toutes les plates-formes. LabVIEW possède un ensemble de VIs TCP et UDP que vous pouvez utiliser pour créer des VIs clients ou serveurs.

Adresses Internet

Chaque hôte d'un réseau IP possède une adresse Internet unique de 32 bits. Cette adresse identifie le réseau auquel l'hôte est relié sur Internet, ainsi que l'ordinateur spécifique sur ce réseau. Utilisez cette adresse pour identifier l'expéditeur ou le destinataire des données. Le protocole IP place l'adresse dans les en-têtes des datagrammes, afin d'assurer l'acheminement correct de chaque datagramme.

La notation décimale à points de IP représente une façon de décrire cette adresse 32 bits, en divisant l'adresse 32 bits en quatre nombres de 8 bits. L'adresse est écrite sous la forme de quatre entiers séparés par des points. Par exemple, l'adresse 32 bits suivante est écrite en notation décimale à points sous la forme `132.13.2.30`.

```
10000100     00001101     00000010     00011110
```

Vous pouvez également utiliser l'adresse 32 bits avec les noms mappés à l'adresse IP. Les drivers de réseau réalisent généralement cette représentation en consultant un fichier local *central* qui contient un nom pour les représentations d'adresses, ou en consultant une base de données plus importante avec le Système de noms de domaine (Domain Name System) pour rechercher d'autres systèmes informatiques comme adresse pour un nom donné. Votre configuration de réseau régit le mécanisme exact de ce processus, connu sous le nom de *résolution d'adresse Internet (hostname resolution).*

Protocole Internet (IP)

Le protocole Internet (Internet Protocol [IP]) réalise le transfert des données de bas niveau entre des ordinateurs. IP regroupe les données dans des composantes appelées *datagrammes*. Un datagramme contient notamment des données et un en-tête indiquant les adresses de source et de destination. IP détermine le chemin correct pour que le datagramme traverse le réseau ou Internet et envoie les données à la destination précisée.

L'hôte d'origine ne connaît peut-être pas le chemin complet qu'emprunteront les données. Grâce à l'en-tête, tout hôte du réseau peut acheminer les données à leur destination, soit directement, soit en les faisant suivre sur un autre hôte. Comme certains systèmes ont des capacités de transfert différentes, IP peut fragmenter les datagrammes en de plus petits segments lorsque c'est nécessaire ; lorsque les données arrivent à destination, IP rassemble automatiquement les données dans leur format d'origine.

Le transfert des données n'est pas garanti avec IP. De plus, comme IP achemine séparément les datagrammes, ceux-ci peuvent arriver dans le désordre. En fait, IP peut envoyer plusieurs fois un même paquet si celui-ci est dupliqué durant la transmission. IP ne détermine pas l'ordre des paquets. Ce sont les protocoles de niveau supérieur à IP qui ordonnent les paquets et qui assurent la fiabilité du transfert. IP est rarement utilisé directement car les programmes utilisent plutôt le protocole TCP ou UDP.

Protocole de datagramme utilisateur (User Datagram Protocol [UDP])

UDP permet d'établir une communication simple de bas niveau entre des processus sur des ordinateurs. Les processus communiquent en envoyant des datagrammes vers une machine ou un port de destination. IP gère le transfert entre machines. Une fois que UDP est installé sur une machine, il transmet le datagramme sur son port de destination. Si le port de destination n'a pas un processus de réception attaché, le datagramme est rejeté. Tous les problèmes de transfert de IP sont également présents dans UDP.

Typiquement, UDP est utilisé dans des applications où la fiabilité n'est pas un critère crucial. Par exemple, une application peut transmettre suffisamment souvent des données d'information sur une même destination pour que quelques segments de données perdus ne représentent pas un problème.

Utilisation de UDP

Contrairement à TCP, UDP n'est pas un protocole basé sur des connexions. Ceci signifie que vous n'avez pas besoin d'établir une connexion avec une destination avant l'envoi ou la réception de données. A la place, la destination des données est précisée lorsque chaque datagramme est envoyé. Le système ne rapporte pas les erreurs de transmission.

Vous pouvez utiliser le VI UDP Open pour ouvrir un port. Un port est l'endroit où les données sont envoyées. Le nombre de ports UDP ouverts simultanément dépend du système. Le VI UDP Open retourne un refnum de connexion réseau, un symbole opaque utilisé dans toutes les opérations ultérieures appartenant à ce port.

Vous pouvez utiliser le VI UDP Write pour envoyer des données à une destination et le VI UDP Read pour les lire. Chaque écriture requiert une adresse et un port de destination. Chaque lecture contient l'adresse et le port de la source. Les limites du paquet sont préservées, à savoir qu'une lecture ne contient jamais des données envoyées lors de deux opérations d'écriture séparées.

En théorie, vous pouvez envoyer des paquets de données de toute taille. S'il le faut, un paquet est désassemblé et réorganisé en de plus petites sections, puis envoyé de cette manière. A leur destination, les sections sont rassemblées et le paquet est présenté au processus demandeur. En pratique, les systèmes n'allouent qu'une certaine quantité de mémoire aux paquets rassemblés. Un paquet qui ne peut pas être rassemblé est abandonné. La taille de paquet la plus importante pouvant être envoyée sans désassemblage dépend du matériel de réseau.

Lorsque LabVIEW termine toutes les communications, l'appel au VI UDP Close libère les ressources du système.

Protocole de gestion de transmission (Transmission Control Protocol [TCP])

Le protocole de gestion de transmission (Transmission Control Protocol [TCP]) assure une fiabilité de transmission dans les réseaux, en envoyant les données dans l'ordre et sans erreur, perte ou duplication. Lorsque vous transférez les données via TCP, celui-ci joint des informations supplémentaires et transfère les données via IP, qui regroupe les données dans des datagrammes et les transmet de cette façon. Ce processus s'inverse au niveau de la réception, où TCP vérifie les données pour tenter d'y déceler des erreurs, ordonne correctement les données, et accuse

réception des transmissions réussies. Si TCP (protocole expéditeur) ne reçoit pas d'accusé de réception, il retransmet le segment de données.

Utilisation de TCP

TCP est un protocole basé sur des connexions, ce qui signifie que les sites doivent établir une connexion avant de transférer des données. TCP permet plusieurs connexions simultanées.

Vous pouvez initier une connexion en attendant une connexion en amont ou en recherchant activement une connexion avec une adresse spécifiée. En établissant les connexions TCP, vous devez préciser à la fois l'adresse et un port à cette adresse. Un port est représenté par un nombre entre 0 et 65535. Sur les systèmes UNIX, les numéros de port inférieurs à 1024 sont réservés pour des applications privilégiées. Différents ports à une adresse donnée identifient différents services à cette adresse et facilitent la gestion de plusieurs connexions simultanées.

Vous pouvez établir activement une connexion avec une adresse et un port spécifiques grâce à la fonction TCP Open Connection. Cette fonction vous permet de préciser l'adresse et le port avec lequel vous souhaitez communiquer. Si la connexion est réussie, la fonction retourne une identification (ID) de connexion qui identifie cette connexion de façon unique. Utilisez cette ID de connexion pour vous référer à la connexion lors d'appels de fonctions ultérieurs.

Il existe deux méthodes pour attendre une connexion en amont :

- Première méthode : utilisez le VI TCP Listen pour créer un auditeur et attendez une connexion TCP acceptée à un port spécifié. Si la connexion est réussie, le VI retourne une ID de connexion, ainsi que l'adresse et le port TCP à distance.
- Seconde méthode : utilisez la fonction TCP Create Listener pour créer un auditeur, puis utilisez la fonction TCP Wait On Listener pour déceler et accepter de nouvelles connexions. La fonction TCP Wait On Listener retourne la même ID d'auditeur que celle transmise à la fonction, ainsi que l'ID de connexion. Lorsque vous n'attendez plus de nouvelles connexions, vous pouvez utiliser la fonction TCP Close pour fermer un auditeur. Vous ne pouvez ni lire ni écrire dans un auditeur.

La seconde méthode est avantageuse, car elle permet d'annuler une opération d'écoute en appelant le VI TCP Close. Ceci est utile dans le cas où vous souhaitez écouter une connexion sans utiliser de timeout, mais où vous souhaitez annuler l'écoute lorsque une certaine autre condition devient vraie (par exemple, lorsque l'utilisateur appuie sur un bouton).

Lorsqu'une connexion est établie, vous pouvez lire et écrire des données dans l'application à distance grâce aux fonctions TCP Read et TCP Write.

Enfin, utilisez la fonction TCP Close Connection pour fermer la connexion de l'application à distance. S'il y a des données non lues et si la connexion se ferme, ces données risquent d'être perdues. Les parties connectées doivent utiliser un protocole de niveau supérieur pour déterminer le moment où fermer la connexion. Une fois qu'une connexion est fermée, vous ne pouvez ni relire ni récrire dessus.

TCP et UDP

Si vous écrivez à la fois sur le client et le serveur et si votre système peut utiliser TCP/IP, le protocole TCP est probablement le meilleur protocole à utiliser car il est fiable et basé sur des connexions. Le protocole UDP n'est pas basé sur des connexions et présente des performances supérieures, mais il n'assure pas une transmission fiable des données.

Exemple de client TCP

Vous trouverez ci-après une description généralisée de l'utilisation des composantes du modèle de diagramme client avec TCP.

Utilisez la fonction TCP Open Connection pour ouvrir une connexion sur un serveur. Vous devez préciser l'adresse Internet du serveur, ainsi que le *port* du serveur. L'adresse identifie un ordinateur sur le réseau. Le port est un numéro supplémentaire qui identifie un canal de communication sur l'ordinateur utilisé par le serveur pour écouter des requêtes de communication. Lorsque vous créez un TCP serveur, vous devez préciser le port que le serveur utilisera pour la communication.

Pour exécuter une commande sur le serveur, utilisez la fonction TCP Write pour envoyer la commande au serveur. Utilisez ensuite la fonction TCP Read pour relire les résultats à partir du serveur. Avec la fonction TCP Read, vous devez préciser le nombre de caractères que vous souhaitez lire. Comme la longueur de la réponse peut varier, ceci peut s'avérer peu commode. Le serveur peut présenter le même problème avec la commande, car la longueur d'une commande peut aussi varier.

Vous pouvez utiliser les méthodes suivantes pour adresser des commandes de tailles différentes :

- Faites précéder la commande et le résultat d'un paramètre de taille fixe qui spécifie la taille de la commande ou du résultat. Dans ce cas, lisez le paramètre de taille, puis lisez le nombre de caractères précisé par la taille. Cette option constitue une méthode efficace et souple.
- Donnez une taille fixe à chaque commande et résultat. Lorsque la taille d'une commande est inférieure à la taille précisée, vous pouvez l'augmenter jusqu'à obtenir la taille spécifiée.
- Faites suivre chaque commande et résultat d'un caractère de terminaison spécifique. Vous devez alors lire les données par petites sections jusqu'à ce que vous obteniez le caractère de terminaison.

Utilisez la fonction TCP Close Connection pour fermer la connexion au serveur.

Timeouts et erreurs

La section précédente traitait du protocole de communication du serveur. Lorsque vous concevez une application réseau, considérez avec attention ce qui se passera en cas de problème. Par exemple, si le serveur tombe en panne, comment ce problème sera-t-il géré par chaque VI client ?

Une solution est de s'assurer que chaque VI possède un timeout (délai d'attente). Si un résultat attendu ne se produit pas, le client va continuer l'exécution pendant un certain temps. En continuant, le client peut essayer de rétablir l'exécution ou de rapporter l'erreur. Si nécessaire, il peut arrêter l'application client.

Exemple de serveur TCP

La section suivante explique comment vous pouvez utiliser TCP pour exécuter chaque composante du modèle de serveur général.

Comme aucune initialisation n'est nécessaire avec TCP, vous pouvez ignorer cette étape.

Utilisez le VI TCP Listen pour attendre une connexion. Vous devez préciser le port qui sera utilisé pour la communication. Ce port doit être le même que celui auquel le client essaie de se connecter. Pour plus d'informations, consultez la section *Exemple de client TCP* dans ce chapitre.

Si une connexion est établie, effectuez une lecture à partir de ce port pour récupérer une commande. Comme indiqué dans l'exemple du client TCP, vous devez décider du format des commandes. Si les commandes sont précédées d'un champ de longueur, lisez d'abord ce champ puis lisez la quantité de données indiquée par le champ.

L'exécution d'une commande doit être indépendante du protocole car elle est réalisée sur l'ordinateur local. Lorsque l'exécution est terminée, transmettez les résultats à la prochaine étape, où ils sont transmis au client.

Utilisez la fonction TCP Write pour retourner des résultats. Comme indiqué dans la section *Exemple de client TCP*, les données doivent se trouver dans un format accepté par le client.

Utilisez la fonction TCP Close pour fermer la connexion.

Vous pouvez ignorer cette étape avec TCP, car tout est terminé une fois que vous avez fermé la connexion.

Serveur TCP avec plusieurs connexions

Le protocole TCP gère aisément plusieurs connexions. Vous pouvez utiliser les méthodes décrites dans la section précédente pour implémenter les composantes d'un serveur avec plusieurs connexions.

Configuration

Avant de pouvoir utiliser TCP/IP, vous devez vous assurer que vous avez une configuration correcte, qui varie selon l'ordinateur que vous utilisez.

UNIX

Le support TCP/IP est intégré. En supposant que votre réseau est configuré correctement, aucune configuration supplémentaire n'est nécessaire pour LabVIEW.

Macintosh

TCP/IP est intégré au système d'exploitation Macintosh version 7.5 et supérieure. Pour utiliser TCP/IP avec un système plus ancien, vous devez installer le logiciel driver MacTCP, disponible auprès de l'Association APDA (Apple Programmer Developer Association). Vous pouvez

contacter l'APDA au (800) 282-2732 (appel depuis les Etats-Unis et le Canada) pour obtenir des informations sur la licence du driver MacTCP. LabVIEW fonctionne également avec Open Transport.

Windows 3.x

Pour utiliser TCP/IP, vous devez installer une carte Ethernet avec son driver de bas niveau. Vous devez également acheter et installer le logiciel TCP/IP qui comprend une DLL Windows Sockets (WinSock) conforme au standard 1.1. WinSock est une interface standard qui active la communication d'application avec différents drivers de réseau. Plusieurs distributeurs fournissent un logiciel de réseau qui inclut la DLL WinSock. Installez la carte Ethernet, les drivers de la carte et la DLL WinSock conformément aux instructions du distributeur de logiciel.

Plusieurs distributeurs fournissent des drivers WinSock qui fonctionnent avec de nombreuses cartes. Contactez le distributeur de votre carte pour savoir s'il offre une DLL WinSock que vous pouvez utiliser avec la carte. Installez la DLL WinSock conformément aux instructions du distributeur.

National Instruments vous recommande d'utiliser la DLL WinSock fournie par Microsoft pour Windows pour Workgroups. Avant de lancer cette DLL WinSock sur le marché, National Instruments a testé de nombreuses DLL WinSock. Les tests ayant démontré que de nombreuses DLL ne sont pas totalement conformes au standard, utilisez une version de démonstration de DLL avant d'acheter la version véritable. Vous pouvez généralement obtenir une version de démonstration auprès du fabricant. La plupart des versions de démonstration sont pleinement fonctionnelles mais expirent au bout d'un certain temps.

Windows 95 et Windows NT

Le support TCP est intégré à Windows 95 et à Windows NT. Vous n'avez pas besoin d'utiliser une DLL de tierce partie pour communiquer si vous utilisez TCP.

22

Support d'ActiveX

Ce chapitre explique la façon d'utiliser LabVIEW comme un serveur et un client ActiveX.

Avec l'Automation ActiveX, vous pouvez accéder aux propriétés et aux méthodes (généralement regroupées dans des objets) d'une application Windows et les utiliser dans une autre application Windows. LabVIEW pour Windows 3.*x* ne supporte pas l'Automation ActiveX. Celle-ci n'est supportée que sous Windows 95 et Windows NT.

Une application supporte l'automation, comme serveur ou comme client. Les applications qui exposent des objets et qui fournissent des méthodes pour utiliser ces objets sont des serveurs d'Automation ActiveX. Quant aux applications qui utilisent les méthodes exposées par une autre application, ce sont des clients d'Automation ActiveX.

LabVIEW peut fonctionner à la fois comme un serveur ActiveX et comme un client ActiveX. LabVIEW peut également afficher un objet ActiveX sur la face-avant avec le container ActiveX. Pour plus d'informations sur l'Automation ActiveX, reportez-vous au chapitre 16, *Commandes ActiveX*, dans le *Manuel de référence de programmation en G*.

☞ **Remarque** ***Tout au long de ce document, le terme*** ActiveX ***fait référence à la technologie ActiveX de Microsoft Corporation et à la technologie OLE.***

Pour des informations générales concernant ActiveX, reportez-vous à la documentation de Microsoft Developers Network et au livre intitulé *Inside OLE****, deuxième édition, de Kraig Brockschmidt.***

Fonctionnalité de serveur d'Automation ActiveX

Puisque vous pouvez utiliser LabVIEW comme un serveur d'Automation ActiveX, d'autres applications activées par ActiveX (comme Microsoft Excel) peuvent nécessiter des propriétés et des méthodes de LabVIEW et des VIs individuels.

Pour activer LabVIEW comme un serveur ActiveX, sélectionnez **Edition»Préférences»Serveur: Configuration**. La boîte de dialogue suivante apparaît :

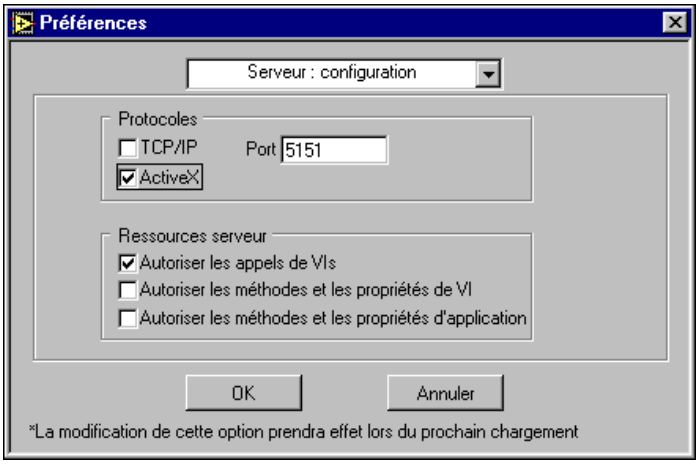


Figure 22-1. Boîte de dialogue Préférences (configuration de serveur)

Sélectionnez le protocole **ActiveX**. LabVIEW exporte une classe pouvant être créée (Application) et une classe de répartition (VirtualInstrument) dans ActiveX. Le progId `LabVIEW.Application` ou `LabVIEW.Application.5` crée un objet d'application. La méthode d'application GetVIReference crée et retourne un pointeur sur l'objet Instrument virtuel.

Propriétés et méthodes de serveur ActiveX

Reportez-vous à la *Référence en ligne* de LabVIEW pour obtenir la description détaillée des propriétés et des méthodes des classes exportées.

Vous trouverez un exemple qui illustre la façon dont vous pouvez utiliser les propriétés et les méthodes dans le répertoire `examples\comm\`

`freqresp.xls`. Cet exemple utilise une macro en script Visual Basic pour exécuter un VI et pour classer des résultats.

Fonctionnalité de client d'Automation ActiveX

LabVIEW peut se comporter comme un client d'Automation ActiveX contrôlant d'autres serveurs d'Automation ActiveX. LabVIEW peut définir et obtenir des propriétés, ainsi qu'exécuter des méthodes rendues disponibles par des serveurs ActiveX. Un serveur exporte des informations concernant ses objets, ses méthodes et ses propriétés grâce à un fichier de bibliothèque de types. Une bibliothèque de types est normalement créée par l'environnement dans lequel les serveurs ont été construits. Pour plus d'informations, reportez-vous à la documentation de chaque application de serveur.

Le tableau 22-1 répertorie et décrit les fonctions que vous pouvez utiliser pour un client d'Automation ActiveX.

Tableau 22-1. Fonctions pour support de client d'Automation ActiveX

Fonction	Description
Automation Open	Sélectionne une classe d'automation à ouvrir
Nœud de méthode	Exécute les fonctions d'une classe
Nœud de propriété	Définit ou obtient les propriétés d'une classe
Automation Close	Ferme un refnum Automation

Vous pouvez créer une application client en C en suivant les étapes suivantes :

1. Obtenez l'interface IDispatch de l'objet dont vous voulez accéder aux méthodes.
2. Obtenez le DispatchID de la méthode de cet objet.
3. Appelez la méthode en utilisant les fonctions Invoquer de l'interface IDispatch, en regroupant tous les paramètres dans la liste des paramètres.

Pour créer une application client dans LabVIEW, suivez les étapes suivantes :

1. Utilisez la fonction Automation Open pour obtenir un refnum Automation définissant de façon unique l'interface IDispatch.
2. Utilisez la fonction "Nœud de méthode" pour exécuter une méthode appartenant à cet objet. LabVIEW convertit ses types de données en variable ActiveX si l'application serveur requiert des données dans ce format.

Les exemples de la section suivante, *Exemples de client ActiveX*, illustrent l'utilisation de ces nœuds.

Exemples de client ActiveX

Les exemples suivants illustrent la façon d'utiliser les nouvelles fonctions ActiveX répertoriées ci-dessus.

Conversion des données de variantes ActiveX en données G

Le premier exemple, indiqué à la figure 22-2, illustre la façon de convertir les données de variantes ActiveX en données G. Dans chaque application ActiveX, vous devez ouvrir un refnum Automation ActiveX au début et fermer le refnum Automation à la fin. Dans cet exemple, vous ouvrez l'objet d'application Microsoft Excel et affichez la propriété "DefaultSaveFormat" en utilisant le nœud de propriété. La propriété "DefaultSaveFormat" se trouve sous le format de variante ActiveX. La fonction "En données G" doit convertir les informations de propriété en un format supporté par LabVIEW.

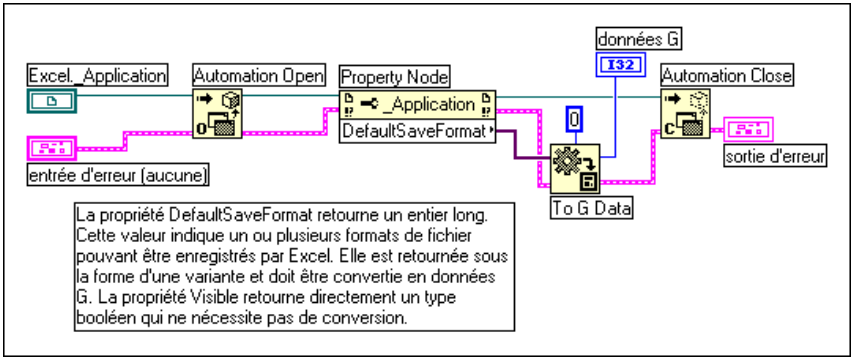


Figure 22-2. Diagramme affichant des données de variantes ActiveX transformées en données G

Pour en apprendre davantage sur les informations de propriété des autres applications, reportez-vous à l'aide en ligne de l'application concernée. La *Référence en ligne de* LabVIEW ne contient pas d'informations de propriété pour d'autres applications activées par ActiveX.

Ajouter un classeur dans Microsoft Excel à partir de LabVIEW

Le second exemple, présenté à la figure 22-3, permet d'ajouter un classeur dans Microsoft Excel à partir de LabVIEW. Comme mentionné précédemment dans cette section, vous devez ouvrir chaque refnum d'application ActiveX avec la fonction "Automation Open" et fermer l'application ActiveX avec la fonction "Automation Close". Pour ajouter un autre classeur, vous devez définir un refnum de classeur. Dans cet exemple, vous ouvrez le refnum Excel avec la fonction "Automation Open" et vous accédez au refnum de classeur avec le nœud de propriété. Une fois que vous avez ajouté un classeur dans Excel, un refnum faisant référence à ce classeur est retourné dans LabVIEW. Lorsque Excel n'a plus besoin d'être ouvert, fermez-le ainsi que le refnum de classeur.

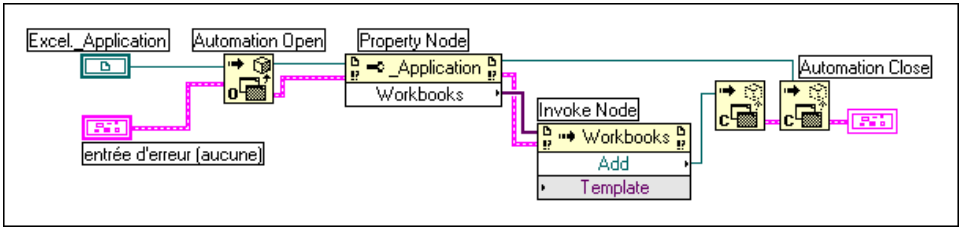


Figure 22-3. Ajouter un classeur dans Microsoft Excel

23

Utilisation du protocole DDE

Ce chapitre décrit les VIs LabVIEW utilisés pour l'échange dynamique de données (Dynamic Data Exchange [DDE]) avec Windows 3.1, Windows 95 et Windows NT. Ces VIs exécutent des fonctions DDE pour le partage de données avec d'autres applications qui acceptent les connexions DDE.

Aperçu du protocole DDE

L'échange dynamique de données (DDE) est un protocole permettant l'échange de données entre des applications Windows.

Dans les communications TCP/IP, les applications ouvrent une ligne de communication puis transfèrent les données brutes. Le protocole DDE fonctionne à un niveau supérieur, où les applications s'envoient des messages pour échanger des informations. Un message simple est par exemple l'envoi d'une commande à une autre application. La plupart des autres messages concernent le transfert des données, les données étant référencées par leur nom.

Pour que la communication DDE puisse commencer, les deux applications doivent être en cours d'exécution et donner à Windows leur adresse de fonction de callback. La fonction callback accepte tous les messages DDE que Windows envoie à l'application.

Un client DDE initie une conversation avec une autre application (un serveur DDE) en envoyant un message de connexion. Après avoir établi une connexion, le client peut envoyer des commandes au serveur et demander ou modifier la valeur des données gérées par le serveur.

Un client peut demander des données depuis un serveur par requête (request) ou avis (advise). Le client utilise une requête pour demander la valeur actuelle des données. Si un client souhaite surveiller une valeur pendant un certain temps, il doit demander d'être informé des changements (advise). En demandant d'être informé de la valeur des données, le client établit un lien entre le serveur et lui-même, grâce auquel le serveur notifie le client lorsque les données changent. Le client peut arrêter la surveillance des données en demandant au serveur d'interrompre le lien d'avis.

Lorsque la communication DDE d'une conversation est terminée, le client envoie un message de fin de conversation au serveur.

Le protocole DDE est surtout adapté à la communication avec des applications grand public classiques telles que Microsoft Excel.

Avec LabVIEW, vous pouvez créer des VIs qui fonctionnent comme des clients envers d'autres applications (ils demandent ou envoient des données à d'autres applications). Vous pouvez également créer des VIs qui fonctionnent comme des serveurs fournissant des informations désignées permettant d'accéder à d'autres applications. En tant que serveur, LabVIEW n'utilise pas une communication basée *connexion*. A la place, vous fournissez des informations désignées à d'autres applications qui peuvent alors les lire ou définir les valeurs de ces informations par leur nom.

Services, sujets et données

Avec TCP/IP, vous identifiez le processus avec lequel vous souhaitez communiquer grâce à son adresse informatique et un numéro de port. Avec DDE, vous identifiez l'application avec laquelle vous souhaitez communiquer en référençant le nom d'un service et d'un sujet. Le serveur décide des noms du service et du sujet de manière arbitraire. En général, un serveur donné utilise le nom de son application pour le service, mais pas nécessairement. Ce serveur peut proposer plusieurs sujets qu'il souhaite communiquer. Avec Excel, par exemple, le sujet peut représenter le nom d'un tableur.

Pour communiquer avec un serveur, identifiez d'abord les noms du service et du sujet dont vous voulez discuter. Ouvrez ensuite une conversation en utilisant ces deux noms pour identifier le serveur.

Sauf si vous êtes sur le point d'envoyer une commande au serveur, vous travaillez généralement avec des données que le serveur souhaite partager. Vous pouvez traiter ces données comme une liste de *variables* que le serveur vous laisse manipuler. Vous pouvez changer ces variables par nom, en fournissant une nouvelle valeur pour la variable. Vous pouvez aussi demander les valeurs des variables par nom.

Exemples de communication client avec Excel

Chaque application qui supporte le protocole DDE possède un ensemble varié de services, de sujets et de données qu'elle peut faire partager. Par exemple, deux programmes tableurs peuvent avoir des approches très différentes pour la spécification de leurs cellules. Pour déterminer ce qui est

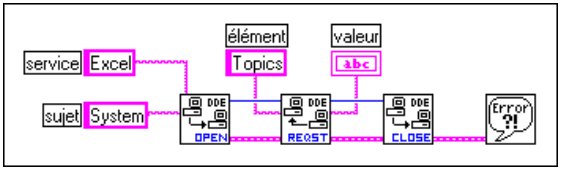
supporté par une application donnée, consultez la documentation fournie avec cette application.

Microsoft Excel, populaire tableur pour Windows, possède le support DDE. Vous pouvez utiliser le protocole DDE pour envoyer des commandes à Excel. Vous pouvez également lire et manipuler des données de tableur en utilisant leur nom. Pour plus d'informations concernant l'utilisation du protocole DDE avec Excel, reportez-vous au *Guide d'utilisation de Microsoft Excel*.

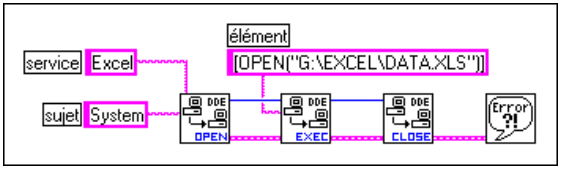
Avec Excel, le nom de service est `Excel`. Pour le sujet, utilisez le nom d'un document ouvert, tel qu'un document tableur, ou bien le mot `System`.

Si vous utilisez le nom `System`, vous pouvez demander des informations sur l'état d'Excel ou envoyer des commandes générales à Excel (commandes qui ne sont pas adressées à un tableur spécifique). Par exemple, pour le sujet `System`, Excel communique des éléments tels que State, qui a la valeur de `Busy` si Excel est occupé, ou `Ready` si Excel est prêt à exécuter les commandes. `Topics` représente une autre donnée que vous pouvez utiliser lorsque le sujet est `System`. Cet élément retourne une liste de sujets sur lesquels Excel peut communiquer, y compris tous les documents tableur ouverts et le sujet `System`.

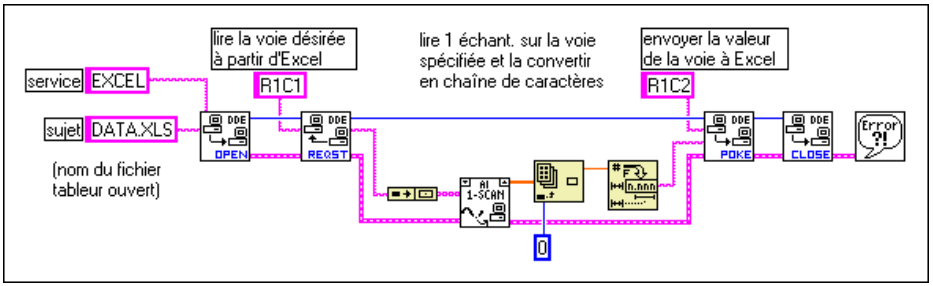
Le VI suivant montre comment vous pouvez utiliser la commande `Topics` dans LabVIEW. La valeur retournée est une chaîne de caractères contenant les noms des tableurs ouverts et le mot `System`.



Une autre façon d'utiliser le sujet `System` avec Excel est de demander à Excel d'ouvrir un document spécifique. Utilisez le VI `DDE Execute` pour envoyer une macro Excel forçant Excel à ouvrir le document, comme présenté dans le diagramme LabVIEW suivant.



Une fois que vous avez ouvert un fichier tableur, vous pouvez envoyer des commandes au tableur pour lire les valeurs de cellules. Dans ce cas, votre sujet est le nom du document du tableur. L'élément est le nom d'une cellule, d'une rangée de cellules, ou de la section désignée d'un tableur. Par exemple, dans le diagramme suivant, LabVIEW peut extraire la valeur de la cellule de la première ligne de la première colonne. Il acquiert ensuite un échantillon à partir du canal spécifié, puis renvoie l'échantillon final à Excel.



Cet exemple utilise les chaînes d'entrée R1C1 et R1C2, qui font respectivement référence à la cellule de la ligne 1, colonne 1 de la feuille de calcul Excel et la cellule de la ligne 1, colonne 2. Si vous utilisez une version d'Excel autre que la version anglaise, utilisez le style de référence de cellule approprié pour votre version d'Excel. Par exemple, si vous utilisez la version française d'Excel, utilisez les chaînes d'entrée L1C1 et L1C2, qui font respectivement référence à la cellule de la ligne 1, colonne 1 et à la cellule de la ligne 1, colonne 2.

VIs LabVIEW comme serveurs DDE

Vous pouvez créer des VIs LabVIEW qui fonctionnent comme des serveurs de données. Le concept général est qu'un VI LabVIEW indique s'il souhaite fournir des informations concernant un service et un sujet particuliers. LabVIEW peut utiliser n'importe quel nom pour le service et le sujet. Il peut spécifier que le nom du service soit le nom de l'application (`LabVIEW`) et que le nom du sujet soit le nom du VI serveur, ou proposer une classification générale des données qu'il fournit, comme `Lab data (données Lab)`.

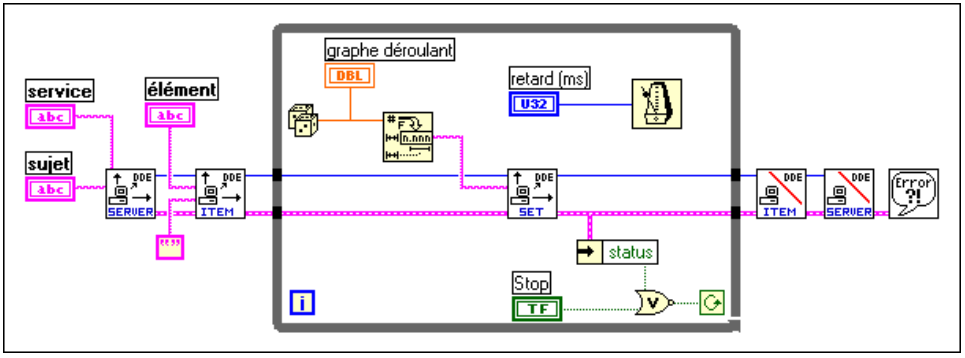
Le VI serveur enregistre alors les données pour un service particulier dont il va parler. LabVIEW mémorise les noms des données et leurs valeurs, et gère la communication avec d'autres applications. Lorsque le VI serveur change la valeur des données enregistrées pour la communication DDE, LabVIEW informe toutes les applications clientes qui ont demandé une

notification concernant ces données. De la même façon, si une autre application envoie un message Poke pour changer la valeur des éléments de données, LabVIEW modifie cette valeur.

Vous ne pouvez pas utiliser la commande DDE Execute avec un VI LabVIEW fonctionnant comme un serveur. Si vous souhaitez envoyer une commande à un VI, vous devez envoyer la commande en utilisant des éléments de données.

Remarquez également que LabVIEW ne possède actuellement rien qui ressemble au sujet Système fourni par Excel. L'application LabVIEW n'est pas, de manière inhérente, un serveur DDE auquel vous pouvez envoyer des commandes ou demander des informations d'état. Vous pouvez néanmoins utiliser les VIs LabVIEW pour créer un serveur DDE.

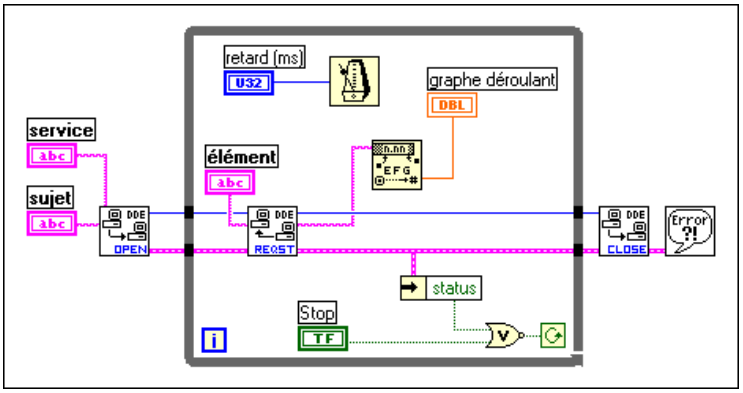
L'exemple suivant indique comment créer un VI DDE Server qui fournit des données à d'autres applications clientes. Dans ce cas, les données représentent des nombres aléatoires. Vous pouvez remplacer facilement les nombres aléatoires par des données réelles, à partir de cartes d'acquisition de données ou de périphériques connectés à l'ordinateur par des connexions GPIB, VXI ou série.



Le VI du diagramme précédent déclare un serveur avec LabVIEW. Le VI déclare un élément qu'il veut fournir aux clients. Dans la boucle, le VI définit périodiquement la valeur de l'élément. Comme mentionné précédemment, LabVIEW signale à d'autres applications que les données sont disponibles. Lorsque la boucle est achevée, le VI se termine en annulant la déclaration de l'élément et celui du serveur.

Les clients de ce VI peuvent être n'importe quelle application qui comprend le protocole DDE, y compris d'autres VIs LabVIEW. Le diagramme suivant illustre un client du VI du diagramme précédent. Il est

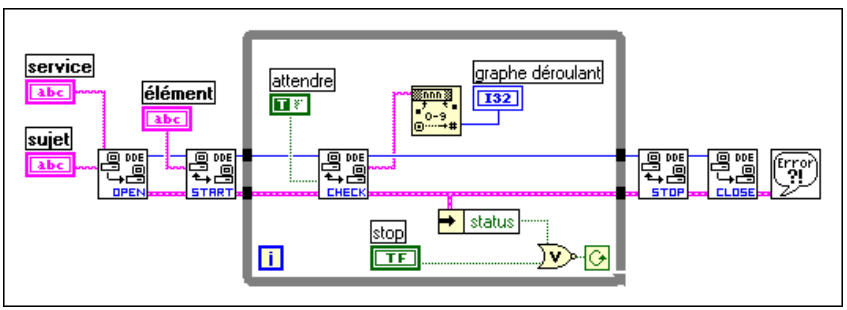
important que les noms du service, du sujet et de l'élément soient les mêmes que ceux utilisés par le serveur.



Requête de données (request) contre avis de données (advise)

L'exemple client précédent utilise le VI DDE Request dans une boucle pour extraire des données. Avec le VI DDE Request, les données sont extraites immédiatement, que vous ayez vu les données précédemment ou non. Si le serveur et le client n'ont pas exactement la même vitesse de boucle, vous pouvez dupliquer ou manquer des données.

Pour éviter la duplication des données, vous pouvez notamment utiliser le VI DDE Advise pour demander la notification des changements concernant la valeur d'une donnée. Le diagramme suivant vous indique comment implémenter ce système.



Dans le diagramme précédent, LabVIEW ouvre une conversation. Il utilise alors le VI DDE Advise Start pour demander la notification des changements concernant la valeur d'une donnée. A chaque itération de la boucle, LabVIEW appelle le VI DDE Advise Check, qui attend qu'une

donnée change de valeur. Lorsque la boucle est achevée, LabVIEW termine la boucle d'information en appelant le VI DDE Advise Stop et en fermant la conversation.

Synchronisation des données

Les exemples de serveur client de la section précédente fonctionnent pour la surveillance de données. En revanche, il n'est pas certain dans ces exemples que le client reçoive toutes les données envoyées par le serveur. Même avec la boucle DDE Advise, si le client ne contrôle pas la modification des données suffisamment souvent, il peut manquer une valeur fournie par le serveur.

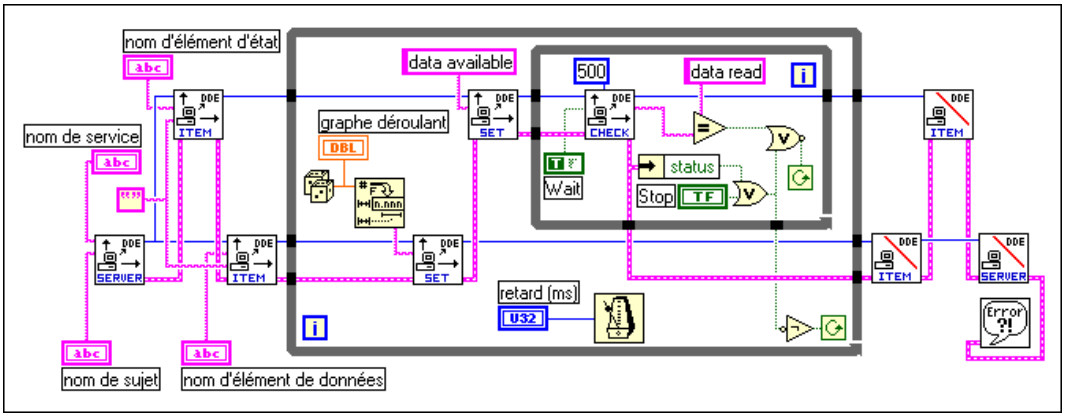
Dans certaines applications, manquer des données n'est pas un problème. Par exemple, si vous surveillez un système d'acquisition de données, les données manquées peuvent ne pas causer de problèmes lorsque vous observez des tendances générales. Dans d'autres applications, il se peut en revanche que vous souhaitiez vous assurer qu'aucune donnée ne manque.

Une différence importante entre les protocoles TCP et DDE est que TCP met les données en file d'attente de manière à ce que vous ne les manquiez pas et que vous les receviez dans le bon ordre. DDE ne fournit pas ce service.

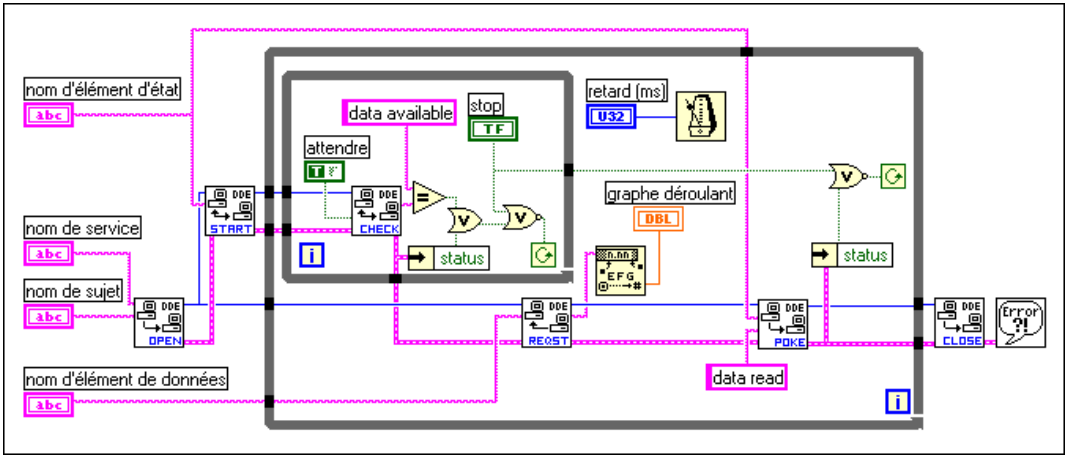
Dans le protocole DDE, vous pouvez définir un élément séparé, que le client utilise pour accuser réception des données les plus récentes. Mettez ensuite à jour les données acquises pour contenir uniquement un nouveau point lorsque le client confirme la réception des données précédentes.

Par exemple, vous pouvez modifier l'exemple de serveur présenté dans la section *Requête de données (request) contre avis de données (advise)*, de ce chapitre, pour définir un élément d'*état* à une valeur spécifique après la mise à jour des données acquises. Le serveur surveille alors l'élément

d'*état* jusqu'à ce que le client accuse réception des données. Cette modification est présentée dans le diagramme suivant.



Un client de ce serveur, comme présenté dans le diagramme suivant, surveille l'élément d'état jusqu'à ce qu'il devienne une *donnée disponible (data avalaible)*. A ce moment-là, le client lit les données à partir des données acquises fournies par le serveur, puis met à jour l'élément d'état pour qu'il prenne la valeur *données lues (data read)*.



Cette technique permet de synchroniser le transfert de données entre un serveur et un client unique. Il y a cependant des points faibles. D'abord, vous ne pouvez avoir qu'un client. Plusieurs clients peuvent se retrouver en conflit les uns avec les autres. Par exemple, un client peut recevoir des données et en accuser réception avant qu'un autre client ne remarque que de nouvelles données sont disponibles. Vous pouvez construire des

diagrammes DDE plus complexes pour résoudre ce problème, mais ils deviennent rapidement difficiles à gérer.

Le contrôle de la vitesse de votre acquisition par la vitesse de votre transfert de données est un autre problème lié à cette technique de synchronisation de la communication. Vous pouvez régler ce problème en séparant l'acquisition et la transmission en deux boucles séparées. L'acquisition peut mettre les données en file d'attente pour qu'elles soient ensuite envoyées par la boucle de transmission. Ceci est semblable à l'exemple de serveur TCP dans lequel le serveur gère plusieurs connexions.

Si votre application nécessite une synchronisation fiable du transfert de données, vous pouvez préférer TCP/IP à la place, parce qu'il fournit la mise en file d'attente, l'accusé de réception du transfert de données et le support de plusieurs connexions au niveau du driver.

DDE en réseau

Vous pouvez utiliser le protocole DDE pour communiquer avec des applications du même ordinateur ou pour communiquer en réseau avec des applications de différents ordinateurs. Pour utiliser DDE en réseau, vous devez exécuter Windows pour Workgroups 3.1 ou une version supérieure, Windows 95 ou Windows NT. La version classique de Windows 3.1 ne supporte pas DDE en réseau.

Chaque ordinateur sous Windows pour Workgroups possède un nom d'ordinateur de réseau. Configurez ce nom en utilisant le panneau de configuration Réseau.

Lorsque vous communiquez dans un réseau, la signification des chaînes de service et de sujet change. Le nom du service change pour indiquer que vous souhaitez utiliser DDE en réseau et inclut le nom de l'ordinateur avec lequel vous désirez communiquer. Le nom du service prend la forme suivante :

`\\`*`nom_ordinateur`*`\ndde$`

Vous pouvez indiquer un nom arbitraire pour le sujet. Modifiez ensuite le fichier `SYSTEM.INI` pour associer ce nom de sujet aux service et sujet réels qui seront utilisés sur l'ordinateur à distance. Cette configuration comporte également des paramètres qui configurent la connexion au réseau. Cette section pourrait ressembler à ce qui est présenté dans l'exemple suivant :

```
[DDE Shares]
nom_sujet= nom_app, sujet_réel, ,31,,0,,0,0,0
```

Nom_sujet représente le nom que votre VI client utilise comme sujet. Nom_app indique le nom de l'application à distance. Avec DDE en réseau, il doit s'agir du nom du service. Sujet_réel est le sujet à utiliser sur l'ordinateur à distance. Les paramètres restants configurent la façon dont DDE fonctionne. Utilisez les paramètres comme indiqué dans l'exemple précédent. La signification de ces paramètres n'est pas expliquée dans la documentation Microsoft.

Par exemple, si vous souhaitez que deux ordinateurs utilisant LabVIEW communiquent en utilisant DDE en réseau, le serveur doit utiliser LabVIEW pour le nom du service, et un nom, tel que donnéeslab, pour le sujet.

Supposons que le nom de l'ordinateur serveur soit Lab ; le client essaie d'ouvrir une conversation en utilisant \\Lab\ndde$ pour le service. Pour le sujet, le client peut utiliser le nom de labdistant.

Pour que cela fonctionne, vous devez modifier le fichier SYSTEM.INI de l'ordinateur serveur afin d'obtenir la ligne suivante dans la section [DDEShares] :

```
labdistant=LabVIEW,donnéeslab,,31,,0,,0,0,0
```

Pour Windows NT, lancez DDEShare.exe, situé dans le répertoire winnt/system 32. Choisissez **Shares»DDE Shares…** puis sélectionnez **Add a Share…** pour enregistrer le nom du service et le nom du sujet sur le serveur.

Utilisation de NetDDE

NetDDE est construit dans Windows pour Workgroups 3.11, Windows 95 et Windows NT. Il est également disponible sur Windows 3.1 avec un package intégré de WonderWare. Si vous utilisez Windows 3.1 avec le package WonderWare, consultez la documentation WonderWare sur l'utilisation de NetDDE.

Si vous utilisez Windows pour Workgroups, Windows 95 ou Windows NT, utilisez les instructions suivantes :

Serveur

Windows pour Workgroups

Ajoutez la ligne suivante à la section `[DDE partages]` du fichier `system.ini` sur le serveur (application recevant des commandes DDE) :

```
lvdemo = nom_service, nom_sujet,,31,,0,,0,0,0
```

où

`lvdemo` peut représenter n'importe quel nom.

`nom_service` est typiquement le nom de l'application, comme `excel`.

`nom_sujet` est typiquement le nom du fichier spécifique, comme `feuille1`.

Entrez d'autres virgules et d'autres nombres comme indiqué.

Windows 95

☞ **Remarque** ***NetDDE ne démarre pas automatiquement avec Windows 95. Vous devez exécuter le programme \WINDOWS\NETDDE.EXE. (Ceci peut être ajouté au dossier de démarrage pour être systématiquement lancé au démarrage.)***

Pour configurer un serveur NetDDE sous Windows 95 :

- Exécutez `\WINDOWS\REGEDIT.EXE`.
- Dans l'arborescence, ouvrez le dossier `My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetDDE\DDE Shares`.
- Créez un nouveau partage DDE en sélectionnant **Edit»New»Key** et appelez-le `vdemo`.
- Avec la clé `lvdemo` sélectionnée, ajoutez les valeurs requises pour le partage comme suit. (Pour référence future, ces clés sont simplement copiées à partir du partage `CHAT$`, mais `REDEGIT` ne vous permet pas de couper, copier ou coller les clés ou les valeurs.) Utilisez **Edit»New** pour ajouter de nouvelles valeurs. Lorsque vous créez la clé, il y a une valeur par défaut appelée (`Default`) et une valeur de (`value not set`). Laissez ces valeurs telles quelles et ajoutez les éléments suivants :

Tableau 23-1. Valeurs à ajouter à la place des valeurs par défaut

Type de valeur	Nom	Valeur
Binaire	Décompte d'élément supplémentaire	`00 00 00 00`
Chaîne de caractères	Application	`nom_service`
Chaîne de caractères	Elément	`nom_service`
Chaîne de caractères	Mot de passe1	`nom_service`
Chaîne de caractères	Mot de passe2	`nom_service`
Binaire	Permissions1	`1f 00 00 00`
Binaire	Permissions2	`00 00 00 00`
Chaîne de caractères	Sujet	`nom_sujet`

- Fermez `REGEDIT`.
- Redémarrez l'ordinateur. (NetDDE doit redémarrer pour que les changements prennent effet.)

Windows NT

Lancez `DDEShare.exe`, qui se trouve dans le répertoire `winnt\system32`. Dans **Shares»DDE Shares»Add a Share...**, choisissez les noms du service et du sujet pour les enregistrer sur le serveur.

Client

Sur l'ordinateur client (application initiant la conversation DDE), aucun changement de configuration n'est nécessaire.

Utilisez les entrées suivantes pour le VI `DDE Open Conversation` :

Service : `\\nom_ordinateur\ndde$`

Sujet : `lvdemo`

où
`nom_ordinateur` précise le nom du serveur.

`lvdemo` correspond au nom précisé dans la section `[DDE Shares]` sur le serveur.

Considérez les exemples de VIs "Graphe déroulant client" (`Chart Client.vi`) et "Graphe déroulant serveur" (`Chart server.vi`) qui se trouvent dans la bibliothèque `examples\network\ddeexamp.llb`. Pour utiliser ces VIs afin d'échanger des informations entre deux ordinateurs grâce à NetDDE, suivez les étapes suivantes :

Serveur :

1. Ne modifiez aucune valeur de la face-avant.
2. Dans le fichier `system.ini` du serveur, ajoutez la ligne suivante dans la section `[DDEShares]` :
   `lvdemo = TestServer,Chart,,31,,0,,0,0,0`

Client :

Sur la face-avant, définissez les commandes de la manière suivante :
Service = `\\nom_ordinateur\ndde$`
Sujet = `lvdemo`
Elément = `Random`

24

AppleEvents

Ce chapitre décrit AppleEvents, une forme de communication inter-applications (IAC) disponible uniquement pour Macintosh, grâce à laquelle des applications Macintosh peuvent communiquer entre elles.

AppleEvents

AppleEvents est un protocole spécifique à Macintosh qui permet aux applications de communiquer entre elles. Comme avec le protocole DDE, les applications utilisent un message pour demander des actions ou retourner des informations provenant d'autres applications. Une application peut envoyer un message à elle-même, à une application du même ordinateur, ou à une application s'exécutant sur un ordinateur dans un autre emplacement du réseau. Vous pouvez utiliser AppleEvents pour envoyer des commandes à d'autres applications, comme ouvrir ou imprimer, ou pour envoyer des requêtes de données, comme des informations concernant un tableur.

LabVIEW contient des VIs permettant d'envoyer certaines commandes communes à la plupart des applications. Les VIs sont faciles à utiliser et ne requièrent pas une connaissance approfondie du fonctionnement d'AppleEvents. La liste suivante répertorie différents types d'utilisation d'AppleEvents dans des applications LabVIEW (liste non exhaustive) :

- Vous pouvez commander à LabVIEW de dire à une autre application (même une application d'un autre ordinateur connecté par réseau) d'effectuer une action. Par exemple, LabVIEW peut demander à un tableur de créer un graphe. Reportez-vous à la section *Envoi d'AppleEvents* dans ce chapitre pour plus de détails.
- Vous pouvez utiliser un programme AppleScript en amont pour demander à LabVIEW d'exécuter des VIs spécifiques.

- En envoyant des instructions pour réaliser des opérations spécifiques, vous pouvez communiquer avec des applications LabVIEW et les contrôler sur d'autres ordinateurs connectés par un réseau. Reportez-vous à la section *Envoi d'AppleEvents* dans ce chapitre pour plus de détails.
- Vous pouvez commander à LabVIEW de s'envoyer des messages, tels que des instructions pour charger, exécuter et décharger des VIs spécifiques. Par exemple, dans des applications de taille importante où la mémoire est limitée, vous pouvez remplacer des appels de sous-VI par un VI utilitaire (le VI AESend Open, Run, Close) et charger, exécuter et décharger les VIs de façon dynamique.

Ces VIs utilisent le VI AESend de bas niveau pour envoyer des AppleEvents. Apple a défini un vaste *vocabulaire* pour les messages afin de standardiser la communication AppleEvent. Vous pouvez combiner des *mots* de ce vocabulaire pour construire des messages complexes. Vous pouvez utiliser ce VI pour envoyer des AppleEvents arbitraires à d'autres applications. Cependant, créer et envoyer des AppleEvents à ce niveau est compliqué et nécessite une connaissance approfondie d'AppleEvents. Reportez-vous pour cela à *Inside Macintosh* et au *AppleEvent Registry.*

Envoi d'AppleEvents

La sous-palette **Communication** de la palette **Fonctions** contient des VIs permettant d'envoyer des AppleEvents. Avec ces VIs, vous pouvez sélectionner une application de destination pour un AppleEvent, créer des AppleEvents et envoyer les AppleEvents à l'application de destination.

La palette **VIs AppleEvent** de la sous-palette **Communication** contient des VIs qui envoient des messages AppleEvent spécifiques. Ces VIs vous permettent d'envoyer plusieurs AppleEvents standard (Open Document, Print Document et Close Application) et tous les AppleEvents personnalisés de LabVIEW. Ces VIs principaux ne demandent pas de connaître la programmation AppleEvent dans les détails. Leurs diagrammes sont de bons exemples de la création et de l'envoi d'AppleEvents.

Vous pouvez utiliser le VI AESend de bas niveau si vous souhaitez envoyer un AppleEvent pour lequel LabVIEW ne fournit pas de VI. La palette **VIs AppleEvent** de la sous-palette **Communication** contient également des VIs qui peuvent vous aider à créer un AppleEvent. Cependant, créer et envoyer un AppleEvent à ce niveau demande une connaissance approfondie d'AppleEvents, comme décrit dans *Inside Macintosh, Volume VI* et dans *AppleEvent Registry.*

Modèle serveur client

Vous ne pouvez pas utiliser les VIs AppleEvent pour créer des diagrammes LabVIEW qui fonctionnent comme des serveurs. Les VIs sont utilisés pour envoyer des messages à d'autres applications. Si vous avez besoin de capacités serveur basées sur diagramme, vous devez utiliser TCP ou PPC.

LabVIEW lui-même se comporte comme un serveur AppleEvent, dans le sens où il comprend et répond à un ensemble d'AppleEvents. De manière spécifique, en utilisant AppleEvents, vous pouvez donner l'ordre à LabVIEW d'ouvrir des VIs, de les imprimer, les exécuter et les fermer. Vous pouvez demander à LabVIEW si un VI particulier s'exécute. Vous pouvez aussi demander à LabVIEW de quitter.

En utilisant ces capacités serveur, vous pouvez donner l'ordre à d'autres applications LabVIEW d'exécuter des VIs et de contrôler LabVIEW à distance. Vous pouvez également commander à LabVIEW de s'envoyer des messages, en demandant le chargement de VIs spécifiques. Par exemple, dans des applications de taille importante où la mémoire est limitée, vous pouvez remplacer des appels de sous-VI par des appels au VI AESend Open, Run, Close pour charger et exécuter les VIs lorsque c'est nécessaire. Remarquez que lorsque vous exécutez un VI de cette manière, sa face-avant s'ouvre, exactement comme si vous aviez sélectionné **Fichier»Ouvrir...**

Exemples de client AppleEvent

Lancement d'autres applications

Pour envoyer un message à une application, cette application doit être en cours d'exécution. Vous pouvez utiliser le VI AESend Finder Open pour lancer une autre application. Ce VI envoie un message au Finder. Le Finder est une application qui comprend un nombre limité d'AppleEvents. L'exemple simple suivant indique comment vous pouvez utiliser AppleEvents pour lancer Teach Text avec un fichier texte spécifique.

Si l'application est sur un ordinateur à distance, vous devez préciser la position de cet ordinateur. Vous pouvez utiliser des entrées au VI AESend Finder Open pour préciser la zone du réseau et le nom du serveur de l'ordinateur avec lequel vous souhaitez communiquer. Si la zone du réseau et le nom du serveur ne sont pas spécifiés, comme dans l'application

précédente, ils sont définis par défaut sur les paramètres de l'ordinateur actuel.

Remarquez que si vous essayez d'envoyer des messages à un autre ordinateur, le système vous demande automatiquement d'ouvrir une session sur cet ordinateur. Il n'y a pas moyen d'éviter ce message, parce qu'il est intégré au système d'exploitation. Ceci peut engendrer des problèmes lorsque vous souhaitez exécuter votre application sur un système informatique non surveillé.

Envoi d'événements à d'autres applications

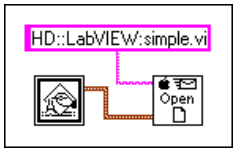
Une fois qu'une application s'exécute, vous pouvez lui envoyer des messages en utilisant d'autres AppleEvents. Les applications ne supportent pas toutes AppleEvents et celles qui le supportent n'acceptent peut-être pas tous les AppleEvents du marché. Pour déterminer les AppleEvents supportés par une application, consultez sa documentation.

Si l'application comprend des AppleEvents, appelez un VI AppleEvent avec l'ID de destination de l'application. Une ID de destination est un cluster qui décrit la position d'une destination sur le réseau (zone, serveur et application support). Vous ne devez pas vous soucier de la structure exacte de ce cluster parce que LabVIEW fournit des VIs que vous pouvez utiliser pour générer une ID de destination.

Il y a deux façons de créer une ID de destination. Vous pouvez utiliser le VI Get Target ID pour créer avec un programme une ID de destination basée sur le nom de l'application et sa position dans le réseau. Ou vous pouvez utiliser le VI PPC Browser, qui affiche une boîte de dialogue listant les applications de réseau supportant AppleEvents. Faites votre choix dans cette liste pour créer une ID de destination.

Vous pouvez aussi utiliser le VI PPC Browser pour déterminer si une autre application utilise des AppleEvents. Si vous utilisez le VI et sélectionnez l'ordinateur qui exécute l'application, la boîte de dialogue liste l'application si elle supporte AppleEvent.

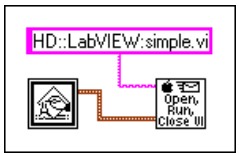
Dans le diagramme suivant, LabVIEW sélectionne interactivement une application qui supporte AppleEvent sur le réseau et lui demande d'ouvrir un document. Dans ce cas, LabVIEW demande à l'application d'ouvrir un VI.



Chargement et exécution dynamique d'un VI

Le VI AESend Open, Run, Close envoie des messages demandant à LabVIEW d'exécuter un VI. Il envoie d'abord le message Open Document, pour que LabVIEW ouvre un VI. Puis le VI "Open Run Close" envoie le message LabVIEW Run VI et LabVIEW exécute le VI spécifié. Le VI "Open, Run, Close" envoie ensuite le message VI Active? et LabVIEW retourne l'état du VI spécifié jusqu'à ce que VI ne s'exécute plus. Finalement, le VI envoie le message Close VI.

En supposant que le programme LabVIEW de destination se trouve sur un autre ordinateur, vous pouvez utiliser le diagramme suivant pour charger et exécuter le VI. Si vous l'envoyez au LabVIEW actuel, vous n'avez pas besoin du VI PPC Browser.



25

Communication programme à programme

Ce chapitre décrit la communication programme à programme (PPC), une forme de communication inter-applications de bas niveau (IAC) d'Apple, grâce à laquelle des applications Macintosh envoient et reçoivent des blocs de données.

Introduction à la communication PPC

La communication programme à programme (PPC) est un protocole Macintosh permettant de transférer des blocs de données entre des applications. Vous pouvez l'utiliser pour créer des VIs qui fonctionneront comme clients ou serveurs. Bien qu'elle soit supportée par tous les Macintosh exécutant System 7.*x*, elle n'est pas souvent utilisée par des applications Macintosh. A la place, la plupart des applications Macintosh utilisent pour communiquer AppleEvents, un protocole de haut niveau permettant d'envoyer des commandes entre applications.

Bien que le protocole PPC ne soit pas aussi souvent supporté qu'AppleEvents, il fournit certains avantages. Parce qu'il est de niveau inférieur, il offre de meilleures performances qu'AppleEvents. Dans LabVIEW, vous pouvez également créer des VIs qui utilisent le protocole PPC pour fonctionner comme des clients ou des serveurs. Vous ne pouvez pas créer des diagrammes qui se comportent comme des serveurs AppleEvent.

Les VIs LabVIEW peuvent utiliser le protocole PPC pour envoyer et recevoir des quantités d'informations importantes entre des applications d'un même ordinateur ou d'ordinateurs différents sur un réseau. Pour que deux applications communiquent avec PPC, elles doivent être en cours d'exécution et être préparées à envoyer ou recevoir des informations.

La structure PPC ressemble à celle du protocole TCP, en termes d'applications serveur et client. La méthode PPC pour spécifier une application à distance est différente de la méthode TCP. A part cela, les deux protocoles offrent les mêmes performances et les mêmes

caractéristiques. Les deux protocoles gèrent la mise en file d'attente et la transmission fiable des données. Vous pouvez utiliser ces deux protocoles avec plusieurs connexions ouvertes.

Lorsque vous choisissez entre les protocoles TCP ou PPC, considérez les plates-formes sur lesquelles vous prévoyez d'exécuter vos VIs et les plates-formes avec lesquelles vous souhaitez communiquer. Si votre application n'est que pour Macintosh, PPC est un bon choix parce qu'il est intégré au système d'exploitation. TCP est intégré au système d'exploitation version 7.5 et supérieure. Pour utiliser TCP avec un système plus ancien, vous devez acheter un driver TCP/IP séparé auprès d'Apple. Si l'achat du driver séparé n'est pas un problème, vous pouvez alors décider d'utiliser le protocole TCP parce que l'interface TCP est plus simple que PPC. PPC utilise en effet des structures de données assez compliquées pour décrire les adresses.

Si votre application doit communiquer avec d'autres plates-formes ou s'exécuter sur d'autres plates-formes, vous devez alors utiliser TCP/IP.

Ports, ID de destination et sessions

Pour communiquer en utilisant PPC, les clients et les serveurs doivent ouvrir des ports à utiliser pour une communication ultérieure. Le VI Open Port ouvre le port en utilisant un cluster qui contient, entre autres choses, le nom que vous voulez utiliser pour le port. Les ports sont utilisés pour faire la distinction entre différents services fournis par une application. Chaque application peut avoir plusieurs ports ouverts en même temps.

Chaque port peut supporter plusieurs sessions ou conversations simultanées. Pour ouvrir une session, un client utilise une ID de destination indiquant la position du serveur. PPC utilise le même type d'ID de destination utilisé par les VIs AppleEvent. Vous pouvez utiliser les VIs PPC Browser ou Get Target ID pour générer l'ID de destination de l'application à distance.

Un serveur attend que les clients essaient d'ouvrir une session en utilisant le VI PPC Inform Session. Le serveur peut accepter ou rejeter la session en utilisant le VI PPC Accept Session. Un client peut essayer d'ouvrir une session avec un serveur en utilisant le VI PPC Start Session.

Une fois que la session est lancée, vous pouvez utiliser les VIs PPC Read et PPC Write pour transférer des données. Vous pouvez fermer une session avec le VI PPC End Session, et vous pouvez fermer un port avec le VI PPC Close Port.

Exemple de client PPC

Les paragraphes suivants expliquent comment utiliser PPC pour réaliser chaque composante du modèle client général.

Utilisez les VIs PPC Open Connection et PPC Open Session pour ouvrir une connexion vers un serveur. Ceci demande que vous spécifiez l'ID de destination du serveur, que vous pouvez obtenir au moyen du VI PPC Browser ou du VI Get Target ID. Vous obtenez ainsi un refnum de port et un refnum de session qui sont utilisés pour communiquer avec le serveur.

Pour exécuter une commande sur le serveur, utilisez le VI PPC Write pour envoyer la commande au serveur. Utilisez ensuite le VI PPC Read pour lire les résultats à partir du serveur. Avec le VI PPC Read, vous devez préciser le nombre de caractères que vous souhaitez lire. Comme avec TCP, ceci peut s'avérer difficile parce que la longueur de la réponse peut varier. Le serveur aura peut-être le même problème parce que la longueur de la commande peut varier.

Plusieurs méthodes pour régler le problème des différentes tailles de commandes sont présentées ci-après. Ces méthodes peuvent aussi être utilisées avec TCP.

- Faites précéder la commande et le résultat d'un paramètre de taille déterminé qui précise la taille de la commande ou du résultat. Dans ce cas, lisez le paramètre de taille, puis lisez le nombre de caractères spécifié par la taille. Cette option est efficace et souple.
- Fixez une taille à chaque commande et chaque résultat. Lorsqu'une commande est plus petite que la taille fixée, vous pouvez la rallonger jusqu'à la taille fixée.
- Faites suivre chaque commande et chaque résultat d'un caractère de terminaison spécifique. Vous devez ensuite lire les données par petits morceaux jusqu'à ce que vous obteniez le caractère de terminaison.

Utilisez les VIs PPC Close Session et PPC Close Connection pour fermer la connexion avec le serveur.

Exemple de serveur PPC

Les paragraphes suivants expliquent comment utiliser PPC pour réaliser chaque composante du modèle serveur général.

Utilisez le VI PPC Open Port dans la phase d'initialisation pour ouvrir une communication avec le port.

Utilisez le VI PPC Inform Session pour attendre une connexion. Avec PPC, vous pouvez accepter automatiquement des connexions entrantes ou choisir d'accepter ou de rejeter la session en utilisant le VI PPC Accept Session. Ce processus d'attente puis d'acceptation de la session vous permet de filtrer les connexions.

Lorsqu'une connexion est établie, vous pouvez lire dans cette session pour extraire une commande. Comme présenté dans la section *Exemple de client PPC*, vous devez décider du format des commandes. Si les commandes sont précédées d'un champ de longueur, vous devez d'abord lire le champ de longueur puis lire la quantité de données.

Etant donné qu'elle est effectuée sur un ordinateur local, l'exécution d'une commande doit être indépendante du protocole. Une fois terminé, vous transférez les résultats à la prochaine étape, où ils sont transmis au client.

Utilisez le VI PPC Write pour renvoyer le résultat. Comme présenté dans la section *Exemple de client PPC*, les données doivent être dans un format acceptable par le client.

Utilisez le VI PPC Close Session pour fermer la connexion.

Lorsque le serveur a terminé, utilisez le VI PPC Close Port pour fermer le port que vous avez ouvert au cours de la phase d'initialisation.

Serveur PPC avec plusieurs connexions

PPC gère facilement plusieurs sessions et plusieurs ports. Les méthodes permettant d'implémenter chaque composante d'un serveur, comme décrit dans la section précédente, fonctionnent aussi pour un serveur avec plusieurs connexions. La figure 25-1 illustre l'ordre dans lequel vous devez utiliser les VIs PPC.

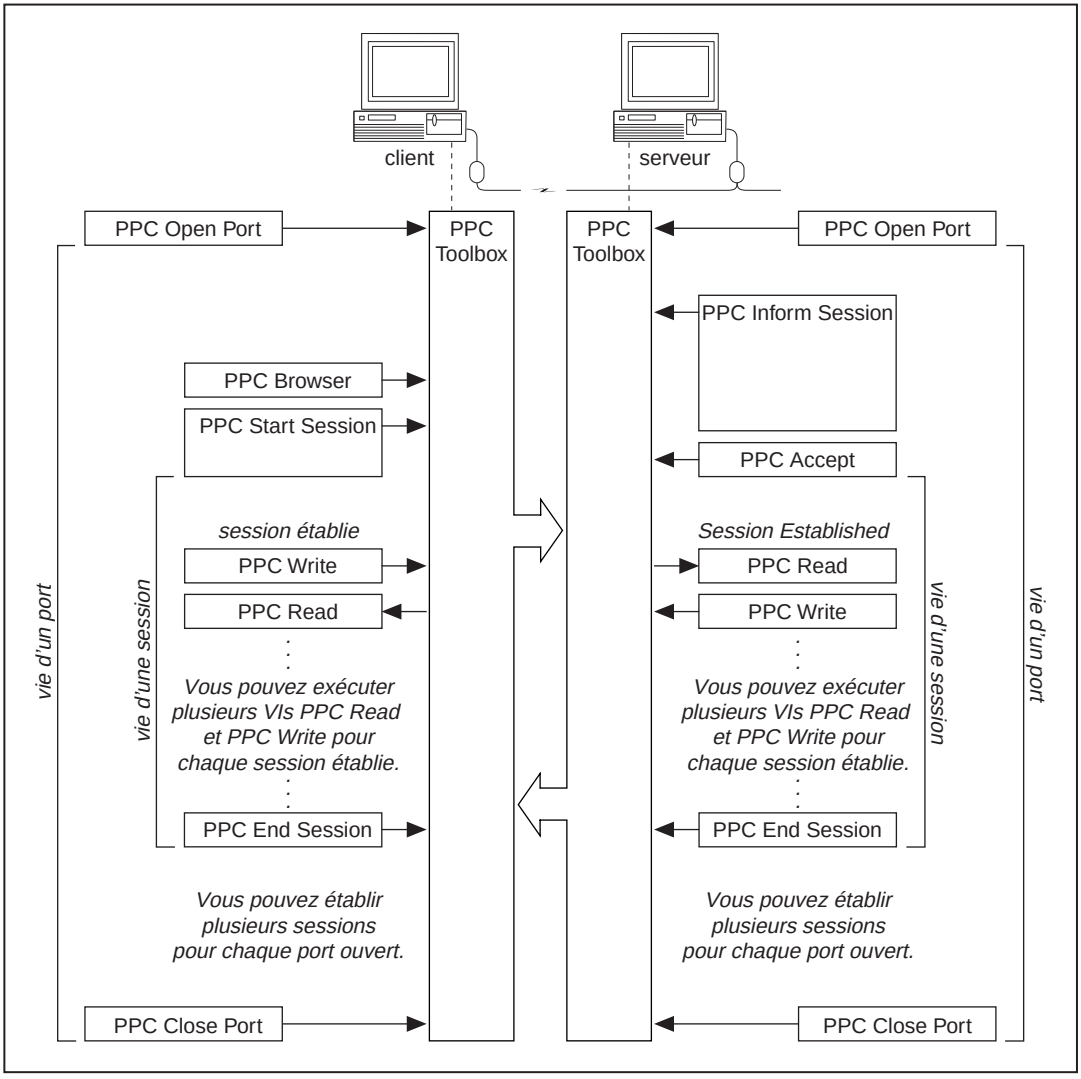


Figure 25-1. Ordre d'exécution des VIs PPC (utilisé avec l'autorisation d'Apple Computer, Inc.)

Partie V

Programmation en G avancée

Cette section contient des informations sur la personnalisation de VIs, la configuration d'objets de la face-avant par programme et la conception d'applications complexes.

La Partie V, *Programmation en G avancée*, contient les chapitres suivants.

- Le chapitre 26, *Personnalisation de VIs*, vous explique comment personnaliser vos VIs. Il contient également des exercices qui illustrent comment utiliser les options **Configuration du VI…** et **Configuration du nœud du sous-VI…** pour personnaliser l'apparence et le comportement d'un VI lorsqu'il s'exécute.
- Le chapitre 27, *Attributs d'objets de la face-avant*, décrit les objets appelés Attribute Nodes, qui sont des **nœud**s de diagramme spéciaux contrôlant l'apparence et les caractéristiques fonctionnelles des commandes et des indicateurs.
- Le chapitre 28, *Conception des programmes*, présente des techniques à utiliser lors de la création de programmes et donne des conseils concernant le style de programmation.
- Le chapitre 29, *Autres ressources disponibles*, fournit des informations sur les ressources que vous pouvez utiliser pour créer vos applications avec succès.

☞ **Remarque** **(Windows 3.1)** ***Vous devez enregistrer les VIs que vous créez dans la partie V dans les bibliothèques de VIs. Les bibliothèques de VIs vous permettent d'utiliser des noms de fichier comportant plus de huit caractères. Egalement, les VIs nécessaires aux exercices de la partie V sont situés dans la bibliothèque de VIs*** `LabVIEW\Activity\Activity.llb`***. Reportez-vous à la section*** *Enregistrement de VIs au* chapitre 2***,*** *Edition de VIs****, du*** *Manuel de référence de programmation en G,* ***pour plus d'informations sur les bibliothèques de VIs.***

En plus de configurer les objets de la face-avant par programmation, vous pouvez également configurer et contrôler les VIs et LabVIEW lui-même par programmation. Par exemple, vous pouvez modifier l'apparence d'un VI, son comportement au cours de l'exécution, les paramètres d'impression pour LabVIEW et déterminer tous les VIs chargés dans la mémoire. Certaines options que vous pouvez définir interactivement dans **Préférences** et **Configuration du VI…** peuvent également être programmées. Les options disponibles dans **Préférences** sont considérées comme des paramètres d'application parce qu'elles affectent tous les VIs de LabVIEW. Les options **Configuration du VI…** sont des paramètres de VI parce qu'elles n'affectent qu'un VI ainsi que tous les cas dans lesquels vous l'utilisez comme sous-VI. En plus de changer la configuration d'un VI et de LabVIEW, vous pouvez aussi spécifier des actions ou des méthodes. Par exemple, vous pouvez réinitialiser tous les objets de la face-avant sur leur valeur par défaut.

Le chargement et l'appel dynamiques d'un VI constituent un autre exemple de contrôle par programmation. Si vous avez une application de taille importante qui a beaucoup de sous-VIs, tous les sous-VIs sont chargés dans la mémoire lorsque vous chargez le VI principal. Charger tous les sous-VIs dans la mémoire peut ne pas être pratique pour de larges applications. A la place, vous pouvez charger et appeler des sous-VIs de manière dynamique en utilisant le nœud d'Appel par référence.

Reportez-vous au chapitre 21, *VI Serveur*, du *Manuel de référence de programmation en G*, et à la *Référence en ligne* de LabVIEW pour plus d'informations sur la configuration et le contrôle de VIs et de LabVIEW par programmation.

En plus de configurer et de contrôler des VIs et LabVIEW sur un ordinateur local, vous pouvez également configurer et contrôler des VIs et LabVIEW sur un ordinateur à distance via un réseau TCP/IP. Reportez-vous au chapitre 7, *Personnalisation de votre environnement*, dans le *Manuel de référence de programmation en G*, pour plus d'informations. Vous pouvez aussi configurer et contrôler des VIs et LabVIEW via une autre application ActiveX. Reportez-vous au chapitre 22, *Support d'ActiveX*, de ce manuel pour des informations sur le support ActiveX dans LabVIEW.

26

Personnalisation de VIs

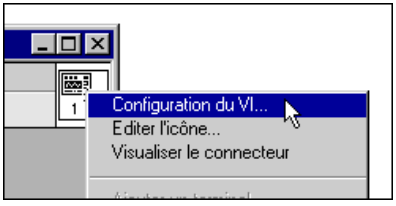
Ce chapitre explique comment personnaliser vos VIs. Il contient aussi des exercices qui illustrent comment accomplir les tâches suivantes :

- Utiliser l'option **Configuration du VI…**
- Utiliser l'option **Configuration du nœud du sous-VI…**

Pour des exemples de VIs personnalisés, reportez-vous à `Examples\General\viopts.llb`.

Comment personnaliser un VI ?

Vous pouvez configurer l'exécution d'un VI de plusieurs façons. Accédez à ces options en ouvrant le menu local du cadre icône en haut à droite de la face-avant et en choisissant **Configuration du VI…**



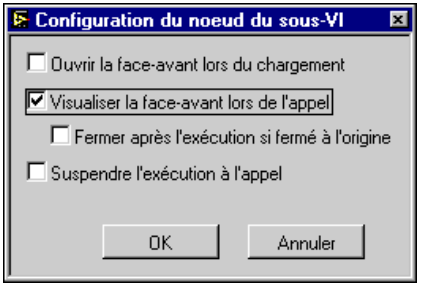
Une boîte de dialogue de configuration du VI apparaît, indiquant les options de configuration pour l'exécution du VI, l'apparence du panneau, de la documentation et de la barre de menus au cours de l'exécution. Vous pouvez apprendre comment utiliser ces options dans l'exercice 26-1. Pour des informations plus détaillées, reportez-vous au chapitre 6, *Configuration des VIs et des sous-VIs*, dans le *Manuel de référence de programmation en G*.

Configuration des options des fenêtres

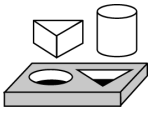
Utilisez **Options des fenêtres** pour contrôler l'apparence du VI lors de son exécution. Pour passer d'**Options d'exécution** à **Options des fenêtres**, cliquez sur la flèche de la barre de menus.

Configuration du nœud d'un sous-VI

Vous pouvez également configurer la manière dont un sous-VI s'exécute. Les options de configuration sont disponibles en ouvrant le menu local de l'icône du sous-VI dans le diagramme du VI appelant et en choisissant **Configuration du nœud du sous-VI…** L'illustration suivante montre la boîte de dialogue Configuration du nœud du sous-VI.



☞ **Remarque** ***Si vous sélectionnez une option dans la boîte de dialogue Configuration du VI…, cette option concerne chaque instance de ce VI. Si vous sélectionnez une option dans la boîte de dialogue Configuration du nœud du sous-VI, l'option ne s'applique qu'à ce nœud particulier.***



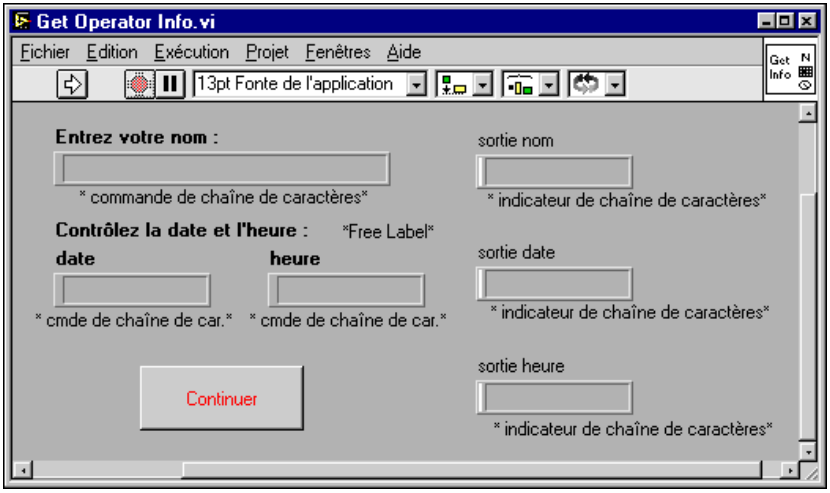
Exercice 26-1. Utilisation des options de configuration d'un sous-VI

Votre objectif est de construire un VI qui demande à l'opérateur d'entrer des informations.

Vous allez créer un VI qui ouvre une boîte de dialogue pour obtenir des informations de l'utilisateur sur l'exécution. Une fois que l'utilisateur a entré des informations et cliqué sur un bouton, la boîte de dialogue disparaît.

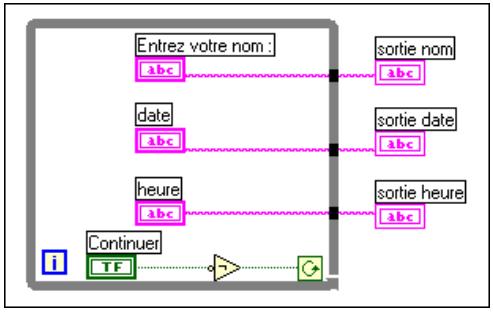
Face-avant

1. Ouvrez une nouvelle face-avant et ajoutez les commandes de chaînes de caractères et le bouton présentés dans l'illustration suivante.



Diagramme

2. Construisez le diagramme présenté dans l'illustration suivante.

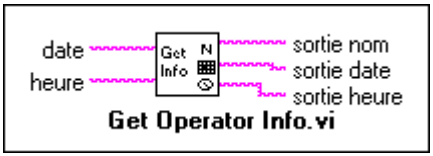


3. Créez l'icône du VI comme présenté ci-contre. Pour accéder à l'Editeur d'icône, ouvrez le menu local du cadre icône de la face-avant et sélectionnez **Editer l'icône**.

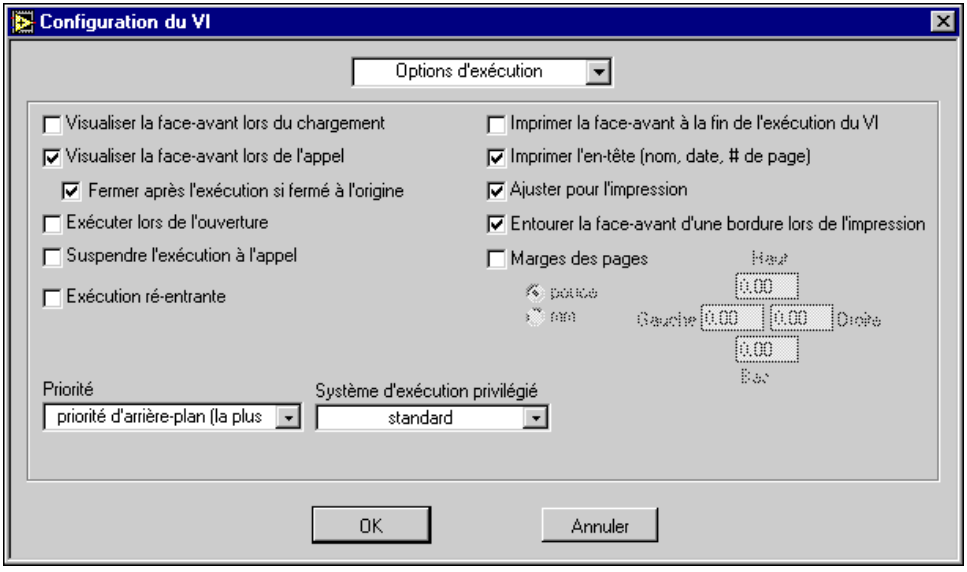
4. Passez au cadre connecteur en ouvrant le menu local du cadre icône et en sélectionnant **Visualiser le connecteur**.

5. Construisez le connecteur. Remarquez que le cadre connecteur par défaut n'est pas le même que le cadre connecteur illustré ci-contre.

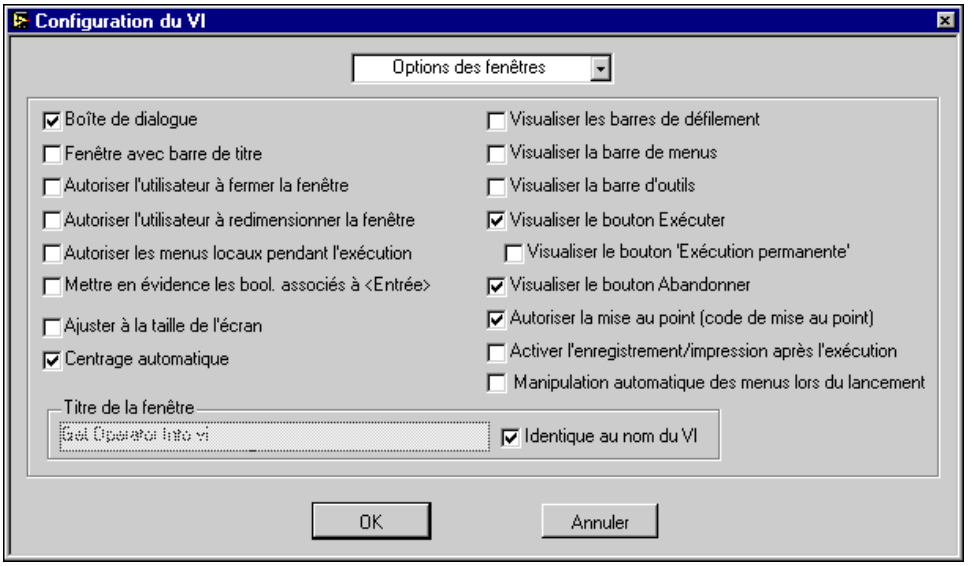
Pour créer le cadre connecteur correct, choisissez **Modèles** dans le menu local du connecteur. Choisissez le motif avec trois entrées et deux sorties. Choisissez ensuite **Basculement horizontal**. Vous pouvez désormais connecter les commandes **Date** et **Heure** aux deux connecteurs sur le côté gauche de l'icône, et les indicateurs **sortie nom**, **sortie date** et **sortie heure** aux trois connecteurs sur le côté droit de l'icône, comme présenté dans l'illustration suivante. Après avoir créé le connecteur, retournez à l'écran de l'icône.



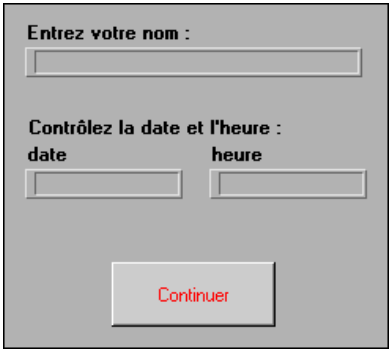
6. Enregistrez le VI sous le nom `Get Operator Info.vi` dans le répertoire `LabVIEW\Activity`.
7. Vous pouvez désormais personnaliser le VI grâce aux options de **Configuration du VI** pour le faire ressembler à une boîte de dialogue.
   a. Ouvrez le menu local de l'icône et sélectionnez **Configuration du VI**. Configurez les **Options d'exécution** comme indiqué dans l'illustration suivante.



b. Sélectionnez **Options des fenêtres** et effectuez les sélections présentées dans l'illustration suivante.



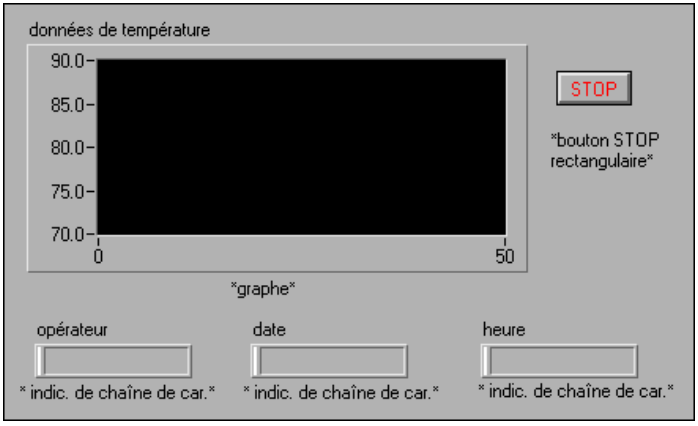
8. Après avoir sélectionné les options de **Configuration du VI**, modifiez la taille de la face-avant, comme présenté dans l'illustration suivante, pour ne pas afficher les trois indicateurs de chaînes de caractères.



9. Enregistrez et fermez le VI. Vous pouvez désormais utiliser ce VI comme un sous-VI.

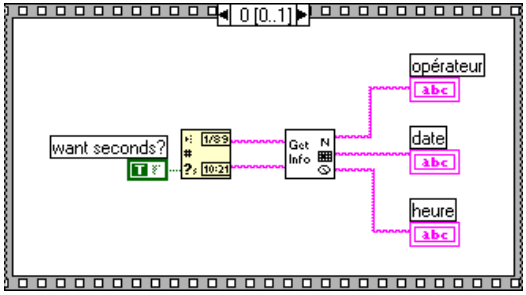
Face-avant

10. Ouvrez une nouvelle face-avant.
11. Placez un graphe déroulant (**Commandes»Graphe**) sur la face-avant et appelez-le `Données de température`.
12. Modifiez l'échelle du graphe de façon à ce que la limite supérieure soit `90,0` et la limite inférieure, `70,0`. Ouvrez le menu local du graphe déroulant et choisissez **Visualiser»Légende** pour masquer la légende. Ouvrez à nouveau le menu local du graphe et choisissez **Visualiser»Palette** pour masquer la palette.
13. Construisez le reste de la face-avant comme présenté dans l'illustration suivante.



Diagramme

14. Créez une structure Séquence et ajoutez les objets suivants à l'étape 0, comme présenté dans l'illustration suivante.



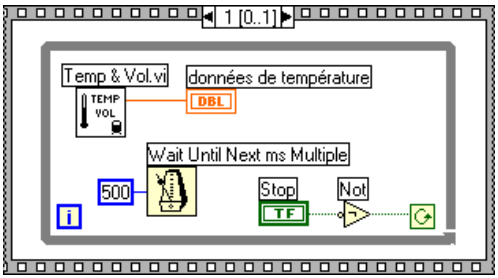
La fonction “Obtenir la chaîne de caractères de la date/heure” (**Fonctions»Temps & Dialogue**) indique la date et l’heure actuelles.

Le VI “Obtenir des informations de l’opérateur” (Get Operator Info.vi) (**Fonctions»Sélectionner un VI…** dans le répertoire `LabVIEW\Activity`) ouvre la face-avant et demande à l’utilisateur d’entrer un nom, la date et l’heure.

La constante Booléenne (**Fonctions»Booléen**) contrôle si la chaîne de caractères de la date et de l’heure est vraie (TRUE). Pour définir cette option sur Vrai (TRUE), cliquez sur la constante avec l’outil Doigt.

15. Ouvrez le menu local de la structure Séquence et sélectionnez **Ajouter une étape après**.

16. Placez une boucle While à l’étape 1 de la structure Séquence.
17. Ajoutez les objets indiqués dans l’illustration suivante.



Le VI “Température et volume” (Temp & Vol.vi) (**Fonctions» Sélectionner un VI…** dans le répertoire `LabVIEW\Activity`) retourne des mesures de température à partir d’un capteur de température simulé.

Le VI “Attendre un multiple de ms” (Wait Until Next ms Multiple.vi) (**Fonctions»Temps & Dialogue**) déclenche l’exécution de la boucle While en ms.

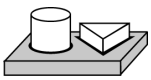
Constante numérique (**Fonctions»Numérique**) : vous pouvez également ouvrir le menu local de la fonction “Attendre un multiple d’impulsion” et sélectionner “Créer une constante” pour créer et connecter automatiquement la constante numérique. La constante numérique retarde l’exécution de la boucle pendant 500 ms (0,5 secondes).

Fonction Non (**Fonctions»Booléen**) : inverse la valeur du bouton **STOP** pour que la boucle While s’exécute de manière répétée jusqu’à ce que vous cliquiez sur **STOP**.

18. Enregistrez le VI sous le nom `Pop-up Panel Demo.vi` dans le répertoire `LabVIEW\Activity`.
19. Exécutez le VI. La face-avant du VI “Obtenir des informations de l’opérateur” (Get Operator Info.vi) s’ouvre et vous invite à entrer votre nom, la date et l’heure. Cliquez sur le bouton **Continuer** pour revenir au VI appelant. Les données sur la température sont ensuite acquises jusqu’à ce que vous cliquiez sur le bouton **STOP**.

☞ **Remarque** ***La face-avant du VI “Obtenir des informations de l’opérateur” (Get Operator Info.vi) s’ouvre à cause des options que vous avez sélectionnées dans la boîte de dialogue Configuration du VI. N’essayez pas d’ouvrir la face-avant du sous-VI à partir du diagramme du VI “Démo de panneau local” (Pop-up Panel Demo.vi).***

20. Fermez toutes les fenêtres.



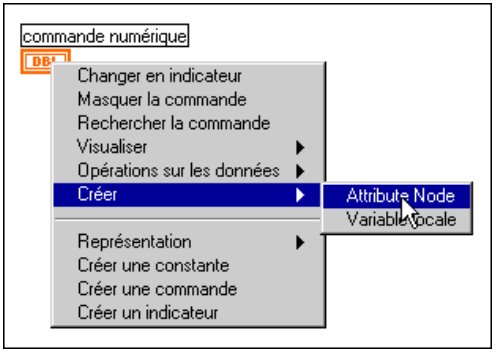
Fin de l’exercice 26-1.

27

Attributs d'objets de la face-avant

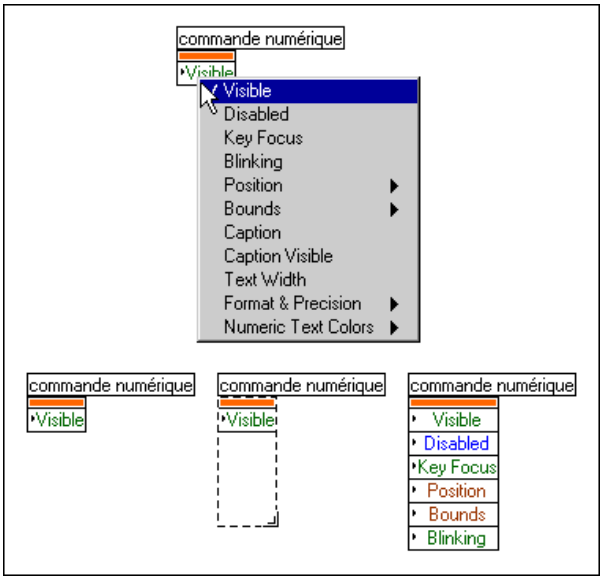
Ce chapitre décrit des objets appelés Attribute Nodes, qui sont des nœuds de diagramme spéciaux qui contrôlent l'apparence et les caractéristiques fonctionnelles des commandes et des indicateurs.

Avec les Attribute Nodes, vous pouvez définir des attributs tels que la couleur, la visibilité, la position, le clignotement de l'écran et bien d'autres choses encore. Pour créer un Attribute Node, sélectionnez **Créer» Attribute Node** dans le menu local de l'objet de la face-avant ou dans le terminal du diagramme, comme présenté dans l'illustration suivante.



Initialement, l'Attribute Node affiche une seule caractéristique. Vous pouvez agrandir le nœud pour afficher plusieurs caractéristiques. Pour agrandir le nœud, sélectionnez l'Attribute Node avec l'outil Flèche. Mettez votre curseur sur le nœud près de l'angle inférieur droit, et lorsque votre curseur se change en cadre, faites-le glisser pour créer le nombre souhaité de caractéristiques. Vous pouvez ensuite modifier les attributs en cliquant

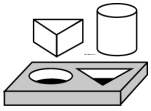
sur le nœud avec l'outil Flèche et en choisissant le nouvel attribut dans le menu local, comme présenté dans l'illustration suivante.



Parce qu'il y a beaucoup d'attributs différents pour les objets de la face-avant, vous pouvez utiliser la fenêtre d'aide pour afficher les descriptions, les types de données et les valeurs acceptables des attributs. Ouvrez la fenêtre d'aide en sélectionnant **Aide»Visualiser l'aide**.

Pour plus d'informations sur l'accès à l'aide dans LabVIEW, reportez-vous à *Obtenir de l'aide* au chapitre 1, *Introduction à la programmation en G*, du *Manuel de référence de la programmation en G*.

Avec les Attribute Nodes, vous pouvez affecter des caractéristiques ou lire l'état actuel d'un attribut en ouvrant le menu local de l'attribut et en sélectionnant **Changer en lecture**.

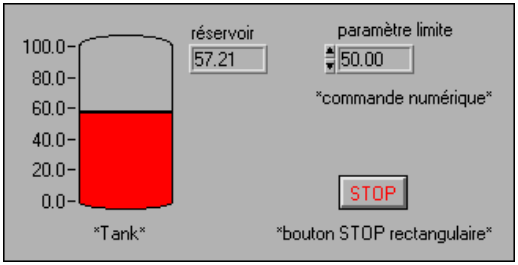


Exercice 27-1. Utilisation d'un Attribute Node

Votre objectif est de créer un VI qui indique une condition de limite haute en utilisant les Attribute Nodes. Vous utiliserez l'attribut* Fill Color (couleur de remplissage) *d'un* `indicateur réservoir` *pour indiquer si un niveau de réservoir généré au hasard a dépassé la limite définie par l'utilisateur.

Face-avant

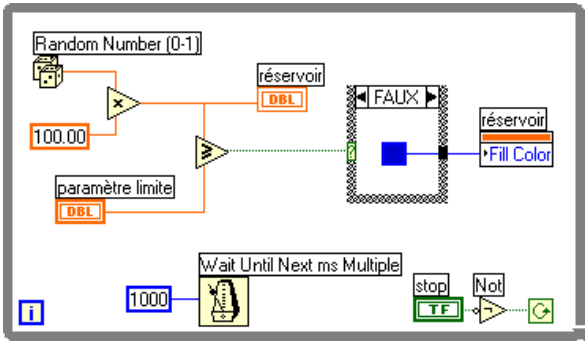
1. Ouvrez une nouvelle face-avant et construisez-la comme présenté dans l'illustration suivante.



2. Faites passer l'échelle du réservoir de `0,0` à `100,0`.
3. Définissez la commande par défaut **Valeurs limites** sur `50,00`.

Diagramme

4. Créez le diagramme comme présenté ci-dessous.



Fonction Non (**Fonctions»Booléen**) : dans cet exercice, la fonction Non change la valeur du bouton **STOP** pour que la boucle While s'exécute de manière répétée jusqu'à ce que vous cliquiez sur le bouton **STOP**. (L'état par défaut du bouton est Faux [FALSE].)

Générateur de nombre aléatoire (**Fonctions»Numérique**) : génère des données brutes entre 0 et 1 pour remplir le réservoir sur votre face-avant. Multipliez cette valeur par 100 pour créer une valeur entre 0 et 100.

Supérieur ou égal ? (**Fonctions»Comparaison**) : compare les données brutes avec l'entrée **Valeur limite**. Si la valeur est supérieure ou égale à

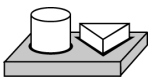
l'entrée limite, une valeur Vrai (TRUE) est transmise à la structure Condition.

Attribute Node (ouvre le menu local du terminal Réservoir) : sélectionnez **Créer»Attribute Node dans le** terminal Réservoir. Ouvrez le menu local de l'attribut et choisissez **Sélectionner»Fill Color**.

Constante Boîte de couleurs (**Fonctions»Numérique»Constantes numériques supplémentaires**) : câblez cette constante pour définir une couleur rouge pour **Fill Color** dans la condition Vrai (TRUE) et une couleur bleue dans la condition Faux (FALSE). Cliquez sur la constante avec l'outil Flèche pour sélectionner la couleur.

Attendre un multiple de ms (**Fonctions»Temps & Dialogue**) : connectez une constante numérique de 1000 pour exécuter la boucle chaque seconde.

5. Exécutez le VI. Le niveau du réservoir est comparé à la commande **Valeur limite**. Si la valeur du réservoir est supérieure ou égale à la valeur **Valeur limite**, le réservoir devient rouge. Si les données tombent en dessous de la limite, le réservoir devient bleu.
6. Enregistrez le VI sous le nom `Tank Limit.vi` dans le répertoire `LabVIEW\Activity`.



Fin de l'exercice 27-1.

28

Conception des programmes

Puisque de nombreux aspects de la programmation en G vous sont désormais familiers, vous devez utiliser vos connaissances pour développer vos propres applications. Ce chapitre propose quelques techniques à utiliser pour la création de programmes, ainsi que des recommandations concernant le style de programmation.

Utilisation de la conception descendante

Lorsque vous devez gérer un projet important, utilisez une *conception descendante*. Comparé aux autres langages de programmation, le G dispose d'un avantage concernant la conception déroulante car vous pouvez commencer avec l'interface utilisateur finale, puis l'animer.

Faites une liste de ce qui est nécessaire à l'utilisateur

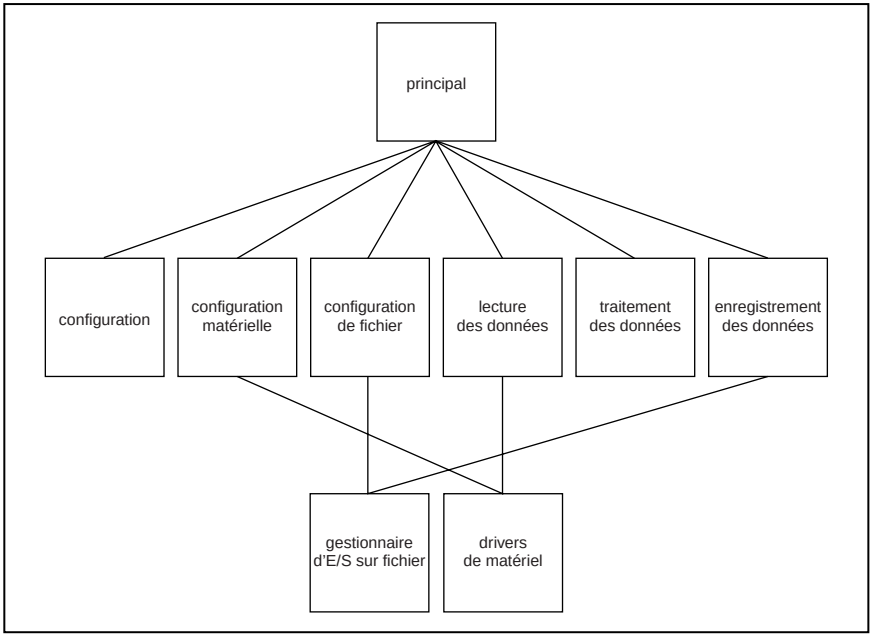
Créez une liste des faces-avant que l'utilisateur peut utiliser, le nombre et le type de commandes et d'indicateurs pour ces faces-avant, le besoin d'analyse en temps réel, la présentation des données, etc. Créez ensuite des maquettes de face-avant que vous pouvez présenter aux utilisateurs potentiels (ou manipuler vous-même, si vous êtes l'utilisateur). Réfléchissez aux fonctions et aux différentes caractéristiques. Utilisez ce processus interactif pour concevoir à nouveau l'interface utilisateur si c'est nécessaire. Vous pouvez avoir besoin, à niveau-là, de faire des recherches superficielles pour vous assurer de pouvoir répondre à ces spécifications.

Concevez la hiérarchie des VIs

La puissance du langage G repose sur la nature hiérarchique des VIs. Après avoir créé un VI, vous pouvez l'utiliser comme sous-VI dans le diagramme de VIs de niveau supérieur. Vous pouvez avoir un nombre illimité de couches dans la hiérarchie.

Divisez la tâche à accomplir en parties logiques et gérables. Comme l'illustre l'organigramme suivant, vous pouvez vous attendre à plusieurs

blocs importants d'une forme ou d'une autre, pour chaque système d'acquisition de données.



Dans certains cas, vous pouvez ne pas avoir besoin de tous ces blocs ou vous pouvez avoir besoin de blocs différents. Par exemple, certaines applications n'incluent pas d'opérations d'E/S sur fichiers. A l'inverse, vous pouvez avoir besoin de blocs supplémentaires, comme des blocs représentant des messages pour l'utilisateur. Votre premier objectif est de diviser votre tâche de programmation en blocs principaux que vous pouvez gérer facilement.

Après avoir déterminé les blocs principaux dont vous avez besoin, essayez de créer un diagramme qui utilise ces blocs principaux. Pour chaque bloc, créez un nouveau *VI à souche* (un prototype non fonctionnel représentant un futur sous-VI). Pour ce VI à souche, créez une icône ainsi qu'une face-avant qui contient les entrées et sorties nécessaires. Vous n'avez pas à créer déjà un diagramme pour ce VI. Voyez en revanche si ce VI souche est un élément nécessaire à votre diagramme principal.

Après avoir assemblé un groupe de VIs à souche, essayez de comprendre, en termes généraux, la fonction de chaque bloc et la manière dont chaque bloc fournit les résultats désirés. Demandez-vous si un bloc donné génère les informations nécessaires à un autre VI. Si tel est le cas, assurez-vous que

l'ébauche de votre diagramme principal contienne des fils de liaison pour transmettre les données entre les VIs.

Essayez d'éviter l'utilisation de variables globales superflues car elles masquent la dépendance des données entre VIs. Votre système s'élargissant, la mise au point devient difficile si vous dépendez de variables globales pour transférer des informations entre VIs.

Créez le programme

Vous êtes désormais prêt à créer le programme en G :

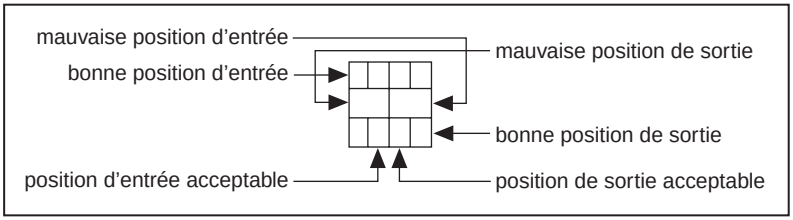
- Utilisez une approche modulaire en construisant des sous-VIs là où vous trouvez une division du travail logique ou un potentiel de réutilisation de code.
- Réglez vos problèmes généraux et spécifiques.
- Testez vos sous-VIs lorsque vous les créez. Vous pouvez avoir besoin de construire des routines de test de niveau supérieur, mais vous pouvez effectuer la mise au point dans un petit module plus facilement que dans une hiérarchie de plusieurs VIs.

En examinant les détails de vos sous-VIs, vous pouvez vous rendre compte que votre conception initiale est incomplète. Par exemple, vous pouvez réaliser que vous avez besoin de transférer davantage d'informations d'un sous-VI à un autre. Vous pouvez avoir à réévaluer votre conception de niveau principal. L'utilisation de sous-VIs modulaires pour accomplir des tâches spécifiques facilite la gestion de vos réorganisations de programmes.

Planification avec des cadres connecteurs

Si vous estimez avoir besoin ultérieurement d'ajouter des entrées ou des sorties, sélectionnez un motif de cadre connecteur avec des terminaux supplémentaires. Vous n'avez pas besoin de connecter ces terminaux supplémentaires. Avec eux, vous n'avez pas à changer le cadre connecteur pour votre VI si vous pensez avoir peut-être besoin plus tard d'autres entrées ou sorties. Cette souplesse vous permet d'effectuer ces changements avec un effet minimal sur votre hiérarchie.

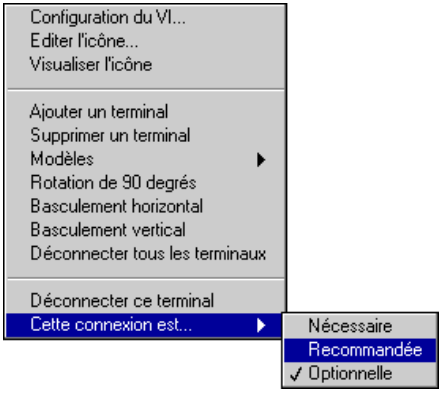
Lorsque vous liez les commandes et les indicateurs au connecteur, mettez les entrées sur la gauche et les sorties sur la droite. Ceci évite les motifs de câblage compliqués et peu clairs dans vos VIs.



Si vous créez un groupe de sous-VIs qui sont souvent utilisés ensemble, essayez de donner aux sous-VIs un cadre connecteur cohérent, avec des entrées communes dans le même emplacement. Vous pouvez ensuite vous rappeler plus facilement où se situe chaque entrée sans utiliser la fenêtre d'aide. Si vous créez un sous-VI qui produit une sortie utilisée comme l'entrée d'un autre sous-VI, essayez d'aligner les connexions d'entrée et de sortie. Cette technique simplifie vos motifs de câblage.

Sous-VIs avec entrées nécessaires

Sur la face-avant, vous pouvez éditer les entrées nécessaires des sous-VIs en cliquant sur le cadre icône sur le côté supérieur droit de la fenêtre et en choisissant **Visualiser le connecteur»Cette connexion est**. Dans le sous-menu, faites votre choix parmi les options **Nécessaire**, **Recommandée**, ou **Optionnelle**. L'illustration suivante affiche les options du sous-menu.



Si vous souhaitez revenir au cadre icône de la face-avant, ouvrez le menu local du cadre connecteur et sélectionnez **Visualiser l'icône**.

Style de diagramme correct

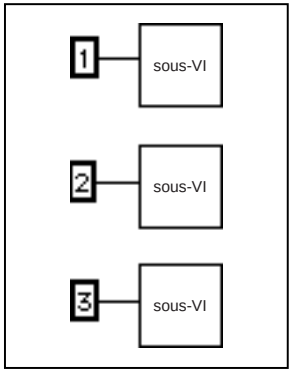
En général, évitez de créer un diagramme qui occupe plus d'un ou deux écrans. Si un diagramme devient très large, décidez si vous souhaitez réutiliser certaines composantes de votre diagramme dans d'autres VIs ou si une section de votre diagramme correspond à une composante logique. Si c'est le cas, vous pouvez envisager de diviser votre diagramme en sous-VIs.

Avec une planification prévoyante et minutieuse, il est plus facile de concevoir des diagrammes utilisant des sous-VIs pour mener des tâches spécifiques. Utiliser des sous-VIs vous aide à gérer des changements et à la mise au point rapide de vos diagrammes. Il vous suffit d'un bref examen pour déterminer la fonction d'un programme bien structuré.

Repérez les opérations courantes

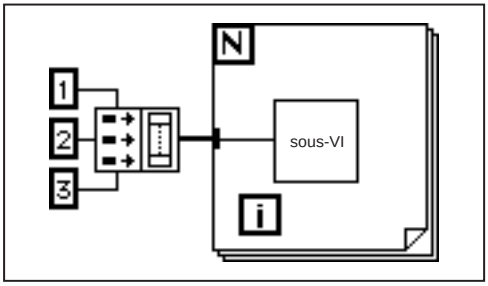
Lorsque vous concevez vos programmes, vous pouvez vous apercevoir que vous réalisez fréquemment certaines opérations. Selon la situation, pensez à utiliser des sous-VIs ou des boucles pour réaliser une action de manière répétée.

Par exemple, examinez le diagramme suivant dans lequel trois opérations semblables s'exécutent indépendamment.



Il est également possible de concevoir une boucle qui réalise l'opération trois fois. Vous pouvez construire un tableau des différents arguments et

utiliser l'auto-indexation pour définir la valeur correcte pour chaque itération de la boucle.



Si les éléments du tableau sont constants, vous pouvez utiliser une constante du tableau au lieu de construire le tableau sur le diagramme.

Utilisez une présentation horizontale

Le langage G est conçu pour une présentation horizontale (et parfois verticale). Organisez tous les éléments de votre programme selon cette présentation lorsque c'est possible.

Recherchez les erreurs

Lorsque vous réalisez n'importe quelle opération d'E/S, envisagez le risque d'erreur. Presque toutes les fonctions d'E/S retournent des informations d'erreur. Si vous utilisez des E/S directes, assurez-vous que votre programme détecte les erreurs et que vous les gérez correctement.

LabVIEW ne gère pas les erreurs automatiquement parce que les utilisateurs souhaitent généralement des méthodes de gestion d'erreur très spécifiques. Par exemple, si un VI d'E/S de votre diagramme dépasse le temps imparti, vous pouvez désirer ou non que votre programme entier s'arrête. Vous pouvez également souhaiter que le VI fasse de nouvelles tentatives pendant un certain temps. Dans LabVIEW, vous pouvez prendre ces décisions de gestion d'erreur dans le diagramme de votre VI.

La liste suivante décrit des situations dans lesquelles des erreurs surviennent fréquemment :

- Initialisation de la communication incorrecte ou données mal écrites sur un périphérique externe
- Perte de courant dans un périphérique externe, ou périphérique externe en panne ou défectueux

- Changement dans la fonctionnalité d'une application ou d'une bibliothèque lors de la mise à niveau du logiciel du système d'exploitation

Lorsqu'une erreur survient, vous ne souhaiterez peut-être pas que d'autres opérations se produisent. Par exemple, si une opération de sortie analogique échoue parce que vous avez spécifié le mauvais périphérique, vous ne souhaiterez peut-être pas qu'une opération d'entrée analogique soit effectuée à la suite.

Une méthode pour gérer un tel problème consiste à déceler les erreurs après chaque fonction et de placer des fonctions ultérieures à l'intérieur des structures Condition. Cependant, cette méthode peut compliquer vos diagrammes et masquer finalement le but de votre application.

Une autre approche, qui a été utilisée avec succès dans de nombreuses applications et bibliothèques de VIs, est d'incorporer la gestion d'erreur dans les sous-VIs qui réalisent des E/S. Chaque VI peut avoir une entrée et une sortie d'erreur. Vous pouvez concevoir le VI pour qu'il vérifie l'entrée d'erreur pour voir si une erreur s'est déjà produite. Si une erreur existe, vous pouvez configurer le VI pour qu'il ne fasse rien et transmette l'entrée d'erreur à la sortie d'erreur. S'il n'y a pas d'erreur, le VI peut exécuter l'opération et transmettre les résultats à la sortie d'erreur.

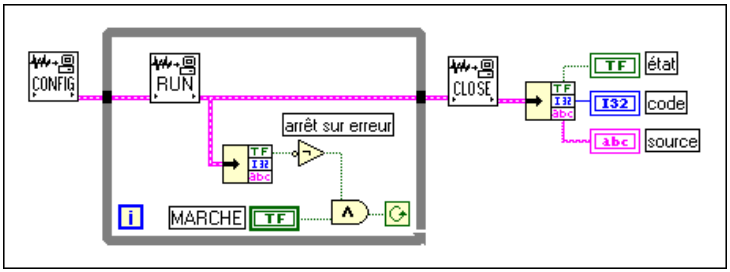
☞ **Remarque** ***Dans certains cas, comme celui d'une opération Fermer, vous pouvez souhaiter que le VI réalise l'opération quelle que soit l'erreur rencontrée.***

En utilisant la technique précédente, vous pouvez câbler plusieurs VIs ensemble, en connectant des entrées et des sorties d'erreur pour propager les erreurs d'un VI à un autre. A la fin des séries de VIs, vous pouvez utiliser le VI "Gestionnaire simple d'erreurs" (Simple Error Handler.vi) pour ouvrir une boîte de dialogue si une erreur survient. Le VI "Gestionnaire simple d'erreurs" se trouve dans **Fonctions»Temps & dialogue**. En plus d'inclure la gestion d'erreur, vous pouvez utiliser cette technique pour déterminer l'ordre de plusieurs opérations d'E/S.

Un des principaux avantages de l'utilisation des clusters d'entrée et de sortie d'erreur est de pouvoir les utiliser pour contrôler l'ordre d'exécution d'opérations différentes.

Les informations d'erreur sont généralement représentées en utilisant un cluster contenant un code d'erreur numérique, une chaîne de caractères contenant le nom de la fonction qui a généré l'erreur et un booléen d'erreur pour un test rapide. L'illustration suivante vous indique comment utiliser

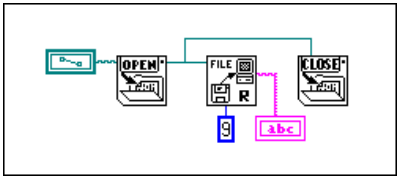
cette technique dans vos propres applications. Remarquez que la boucle While s'arrête si elle détecte une erreur.



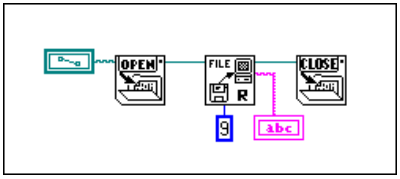
Repérez les dépendances manquantes

Assurez-vous d'avoir défini explicitement la séquence d'événements lorsque c'est nécessaire. Ne supposez pas une exécution horizontale ou verticale lorsqu'aucune dépendance de données n'existe.

Dans l'exemple suivant, aucune dépendance n'existe entre le VI "Lire un fichier" (Read File.vi) et le VI "Fermer un fichier" (Close File.vi). Ce programme ne fonctionnera peut-être pas comme prévu.

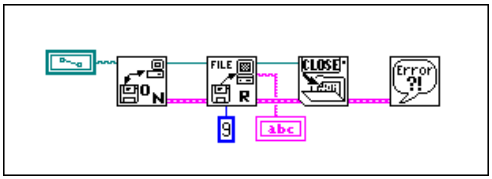


La version suivante du diagramme établit une dépendance en câblant une sortie du VI "Lire un fichier" (Read File.vi) au VI "Fermer un fichier" (Close File.vi). L'opération ne peut pas se terminer avant que le VI "Fermer un fichier" ne reçoive la sortie du VI "Lire un fichier".



Remarquez que l'exemple précédent ne détecte pas encore les erreurs. Par exemple, si le fichier n'existe pas, le programme n'affiche aucun avertissement. La version suivante du diagramme illustre une technique pour la gestion de ce problème. Dans cet exemple, le diagramme utilise les

entrées et les sorties d'E/S d'erreur de ces fonctions pour propager des erreurs au VI "Gestionnaire simple d'erreurs" (Simple Error Handler.vi).



Evitez l'utilisation excessive des structures Séquence

Vu que les VIs peuvent fonctionner avec un grand nombre d'opérations internes parallèles, évitez l'utilisation des structures Séquence. L'utilisation d'une structure Séquence garantit l'ordre d'exécution mais interdit des opérations parallèles. Par exemple, des tâches asynchrones qui utilisent des périphériques d'E/S (GPIB, ports série et cartes d'acquisition de données) peuvent s'exécuter en même temps que d'autres opérations si les structures Séquence ne les empêchent pas de le faire.

Les structures Séquence ont tendance à masquer des parties du programme et à interrompre le flux horizontal naturel des données. Vous ne sacrifiez pas les performances en utilisant les structures Séquence. Cependant, lorsque vous devez séquencer des opérations, vous pouvez envisager d'utiliser le flux de données à la place. Par exemple, dans des opérations d'E/S, vous pouvez utiliser la technique d'E/S d'erreur décrite précédemment pour vous assurer qu'une opération d'E/S survient avant une autre.

Etudiez les exemples

Pour plus d'informations sur la conception de programmes, vous pouvez examiner les nombreux exemples de diagrammes inclus dans LabVIEW. Ces échantillons de programmes vous permettent de mieux comprendre le style et la technique de la programmation en G. Pour afficher ces diagrammes, ouvrez n'importe quel VI du répertoire `Examples`.

29

Autres ressources disponibles

Vous avez terminé les exercices qui vous ont préparé à créer des applications LabVIEW. Avant de lancer vos propres applications, vous souhaiterez peut-être examiner les ressources supplémentaires à votre disposition.

Autres ressources utiles

Les sections suivantes présentent des ressources que vous pouvez utiliser pour créer vos applications avec succès.

“Assistant Solutions” et “Recherche d’exemples”

Pour trouver des exemples similaires à votre application, ouvrez les options **Assistant Solutions** et **Recherche d’exemples** dans la boîte de dialogue LabVIEW. L’**Assistant Solutions** crée des exemples d’acquisition de données et d’E/S d’instrument basés sur les critères que vous précisez. L’option **Recherche d’exemples** ouvre des exemples illustrant plusieurs concepts de programmation en G ainsi que l’analyse, la mise en réseau, l’acquisition de données et les E/S d’instrument.

Applications d’acquisition de données

(Windows, Macintosh) Le *Manuel de base d’acquisition des données LabVIEW* fournit des informations sur le lancement d’applications d’acquisition de données qui utilisent les entrées et les sorties analogiques, les E/S numériques et les compteurs/timers. Le *Manuel de base d’acquisition des données LabVIEW* explique également les concepts fondamentaux de l’acquisition de données et les VIs utilisés pour implémenter ces concepts.

Techniques de programmation en G

Dans ce manuel, la partie I, *Introduction à la programmation en G*, et la partie V, *Programmation en G avancée*, vous ont présenté les techniques de programmation en G fondamentales. Pour en apprendre davantage sur les capacités du langage G, reportez-vous au *Manuel de référence de programmation en G*, qui fournit des informations supplémentaires sur

l'exécution et la mise au point, la configuration des VIs, les objets de la face-avant, les fils de liaison, les structures et les Attribute Nodes. Il fournit également des informations qui ne sont pas présentes dans le *Manuel de l'utilisateur LabVIEW*, telles que l'impression, la personnalisation de l'environnement G en utilisant les **Préférences**, les commandes personnalisées, le multithreading et les questions de performance.

Références concernant les fonctions et les VIs

Pour un aperçu des fonctions et des VIs disponibles dans LabVIEW, reportez-vous au *Manuel de référence des VIs et des fonctions de LabVIEW*, qui fournit de brèves descriptions des fonctions et des VIs, organisées dans un ordre identique à celui dans lequel ils apparaissent dans la palette **Fonctions**.

Ressources pour les sujets avancés

Le *Manuel de l'utilisateur LabVIEW vous apprend les fondements de* la construction d'une application LabVIEW. LabVIEW contient plusieurs fonctions avancées qui ne sont pas présentées ou seulement en partie dans ce manuel. Les sections suivantes donnent une vue générale de ces fonctions et fournissent des ressources supplémentaires pour que vous puissiez les appliquer au besoin dans vos applications.

Attribute Nodes

Le chapitre 27 de ce manuel, *Attributs d'objets de la face-avant*, décrit brièvement les Attribute Nodes. En utilisant les Attribute Nodes, vous pouvez gérer des paramètres relatifs aux commandes et aux indicateurs par programmation. Par exemple, vous pouvez changer la visibilité des commandes. Utilisez un Attribute Node si vous devez changer les options dans un menu déroulant ou une commande de type liste, effacer le contenu d'un graphe déroulant, ou modifier les échelles d'un graphe ou d'un graphe déroulant par programmation. Le chapitre 22, *Attribute Nodes*, du *Manuel de référence de programmation en G* décrit les Attribute Nodes en détail.

Configuration et préférences des VIs

Vous pouvez contrôler par programme différents attributs de VIs de même que des attributs de LabVIEW en utilisant la fonction "VI Serveur". Vous pouvez charger des VIs en mémoire, exécuter des VIs, modifier l'apparence des VIs et modifier la façon dont un VI s'exécute. Certaines des options que vous pouvez définir interactivement dans **Préférences** et **Configuration du VI…** peuvent également être programmées. Les options

disponibles dans **Préférences** sont considérées comme des paramètres d'application parce qu'elles affectent tous les VIs de LabVIEW. Les options **Configuration du VI…** sont des paramètres de VI parce qu'elles n'affectent qu'un VI, ainsi que tous les cas dans lesquels vous utilisez le VI comme sous-VI.

En plus du changement d'attributs, qui sont appelés propriétés, vous pouvez réaliser une action sur LabVIEW ou sur un VI individuel. Par exemple, vous pouvez définir une action (appelée également méthode) pour enregistrer un VI. Vous pouvez définir des propriétés et des méthodes pour LabVIEW et des VIs sur un ordinateur local, via un réseau TCP/IP ou une autre application ActiveX. Pour plus d'informations sur les capacités ActiveX, reportez-vous au chapitre 22, *Support d'ActiveX*, dans ce manuel. Pour plus d'informations sur les paramètres TCP/IP et sur la manière d'utiliser cette fonction, reportez-vous au chapitre 7, *Personnalisation de votre environnement en G*, et au chapitre 21, *VI Serveur*, dans le *Manuel de référence de programmation en G*.

Variables locales et globales

Utilisez des variables locales pour lire les commandes qui se trouvent dans plusieurs positions du diagramme. Utilisez également des variables locales lorsque vous devez traiter un objet de la face-avant comme une commande dans certaines positions et comme un indicateur dans d'autres. Utilisez des variables locales avec parcimonie parce qu'elles masquent le flux de données de vos diagrammes. Il devient ainsi difficile de voir l'objectif de votre programme et de mettre au point les variables locales. Reportez-vous au chapitre 23, *Variables locales et globales*, dans le *Manuel de référence de programmation en G* pour une présentation des variables locales.

Les variables globales enregistrent les données utilisées par plusieurs VIs. Utilisez les variables globales judicieusement parce qu'elles masquent le flux de données de votre diagramme. Bien que vous ayez besoin des variables globales dans certaines applications, ne les utilisez pas si vous pouvez structurer votre programme en utilisant une autre méthode de flux de données. Reportez-vous au chapitre 23, *Variables locales et globales*, dans le *Manuel de référence de programmation en G* pour plus de détails.

Création de sous-VIs

Vous pouvez créer des sous-VIs à partir d'une sélection sur le diagramme en choisissant **Edition»Créer un sous-VI**. En outre, LabVIEW câble automatiquement les entrées et les sorties correctes au sous-VI. Dans certains cas, vous ne pouvez pas créer un sous-VI à partir d'un VI.

Reportez-vous au chapitre 3, *Utilisation de sous-VIs*, du *Manuel de référence de programmation en G*, pour une présentation détaillée de cette fonction.

Profils de VI

Vous pouvez utiliser la fonction de profil de VI (**Projet»Fenêtre d'optimisation**) pour accéder aux informations détaillées sur les statistiques et les détails temporels d'un VI. Cette fonction vous aide à optimiser les performances de vos VIs. Reportez-vous au chapitre 28, *Performance*, du *Manuel de référence de programmation en G*, pour une présentation détaillée de la fonction de profil.

Editeur de commandes

Utilisez l'Editeur de commandes pour personnaliser l'apparence des commandes de votre face-avant. Vous pouvez utiliser l'éditeur pour enregistrer des commandes personnalisées afin de pouvoir les réutiliser dans d'autres applications. Reportez-vous au chapitre 24, *Commandes personnalisées et définitions types*, du *Manuel de référence de programmation en G*, pour une présentation détaillée de l'Editeur de commandes.

Commandes de liste et de menu déroulant

Utilisez des commandes de liste et de menu déroulant lorsque vous devez présenter à l'utilisateur une liste d'options. Reportez-vous au chapitre 13, *Commandes et indicateurs de liste et de menu déroulant*, du *Manuel de référence de programmation en G*, pour une présentation détaillée de ces objets de la face-avant.

Fonction "Appeler une fonction d'une DLL"

LabVIEW fournit une fonction "Appeler une fonction d'une DLL" que vous pouvez utiliser pour appeler une bibliothèque partagée ou DLL. Avec cette fonction, vous pouvez créer une interface dans LabVIEW pour appeler un code ou un driver existant. Reportez-vous au chapitre 25, *Appel d'un code depuis d'autres langages*, dans le *Manuel de référence de programmation en G*, pour une présentation détaillée des *fonctions* "Appeler une bibliothèque".

Code Interface Nodes

Vous pouvez utiliser des code interface nodes (CIN) comme une autre méthode pour appeler un code source écrit dans un langage de programmation basé texte conventionnel, à partir des diagrammes LabVIEW. Utilisez les CIN pour des tâches que les langages de programmation conventionnels peuvent réaliser plus rapidement que LabVIEW, pour des tâches que vous ne pouvez pas réaliser directement à partir du diagramme et pour lier un code existant à LabVIEW. Cependant, la fonction “Appeler une fonction d’une DLL” est généralement plus facile à utiliser que les CIN lorsque vous appelez un code source. Utilisez les CIN lorsque vous avez besoin d’une meilleure intégration avec LabVIEW et le code source. Reportez-vous au document *LabVIEW Code Interface Reference Manual*, disponible uniquement au format document portable (PDF), et au chapitre 25, *Appel d’un code depuis d’autres langages*, dans le *Manuel de référence de programmation en G*, pour plus d’informations sur les CIN.

A

Références d'analyse

Cette annexe liste les documents de référence utilisés pour créer les VIs d'analyse de ce manuel. Ces références contiennent davantage d'informations sur les théories et les algorithmes implémentés dans la bibliothèque d'analyse.

Baher, H. *Analog & Digital Signal Processing*. New York : John Wiley & Sons, 1990.

Bates, D.M. et Watts, D.G. *Nonlinear Regression Analysis and its Applications*. New York : John Wiley & Sons, 1988.

Bracewell, R.N. "Numerical Transforms." *Science* 248 (11 mai 1990).

Burden, R.L. et Faires, J.D. *Numerical Analysis. 3ème éd*. Boston : Prindle, Weber & Schmidt, 1985.

Chen, C.H. et al. *Signal Processing Handbook*. New York : Marcel Decker, Inc., 1988.

DeGroot, M. *Probability and Statistics. 2ème éd*. Reading, Massachusetts : Addison-Wesley Publishing Co., 1986.

Dowdy, S. et Wearden, S. *Statistics for Research. 2ème éd*. New York : John Wiley & Sons, 1991.

Dudewicz, E.J. et Mishra, S.N. *Modern Mathematical Statistics.* New York : John Wiley & Sons, 1988.

Duhamel, P. et al. "On Computing the Inverse DFT." *IEEE Transactions on ASSP* 34, 1 (février 1986).

Dunn, O. et Clark, V. *Applied Statistics: Analysis of Variance and Regression. 2ème éd*. New York : John Wiley & Sons, 1987.

Elliot, D.F. *Handbook of Digital Signal Processing Engineering Applications*. San Diego : Academic Press, 1987.

Golub, G.H. et Van Loan, C.F. *Matrix Computations.* Baltimore : The John Hopkins University Press, 1989.

Harris, Fredric J. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform." *Proceedings of the IEEE-66* (1978)-1.

Maisel, J.E. "Hilbert Transform Works With Fourier Transforms to Dramatically Lower Sampling Rates." *Personal Engineering and Instrumentation News* 7, 2 (février 1990).

Miller, I. et Freund, J.E. *Probability and Statistics for Engineers*. Englewood Cliffs, New Jersey : Prentice-Hall, Inc., 1987.

Neter, J. et al. *Applied Linear Regression Models*. Richard D. Irwin, Inc., 1983.

Neuvo, Y., Dong, C.-Y. et Mitra, S.K. "Interpolated Finite Impulse Response Filters" *IEEE Transactions on ASSP*. ASSP-32, 6 (juin 1984).

O'Neill, M.A. "Faster Than Fast Fourier." *BYTE*. (avril 1988).

Oppenheim, A.V. et Schafer, R.W. *Discrete-Time Signal Processing*. Englewood Cliffs, New Jersey : Prentice Hall, 1989.

Parks, T.W. et Burrus, C.S. *Digital Filter Design*. New York : John Wiley & Sons, Inc., 1987.

Pearson, C.E. *Numerical Methods in Engineering and Science*. New York : Van Nostrand Reinhold Co., 1986.

Press, W.H. et al. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge : Cambridge University Press, 1988.

Rabiner, L.R. et Gold, B. *Theory and Application of Digital Signal Processing*. Englewood Cliffs, New Jersey : Prentice Hall, 1975.

Sorensen, H.V. et al. "On Computing the Split-Radix FFT." *IEEE Transactions on ASSP*. ASSP-34, 1 (février 1986).

Sorensen, H.V. et al. "Real-Valued Fast Fourier Transform Algorithms." *IEEE Transactions on ASSP*. ASSP-35, 6 (juin 1987).

Spiegel, M. *Schaum's Outline Series on Theory and Problems of Probability and Statistics*. New York : McGraw-Hill, 1975.

Stoer, J. et Bulirsch, R. *Introduction to Numerical Analysis*. New York : Springer-Verlag, 1987.

Vaidyanathan, P.P. *Multirate Systems and Filter Banks.* Englewood Cliffs, New Jersey : Prentice Hall, 1993.

Wichman, B. et Hill, D. "Building a Random-Number Generator : A Pascal Routine for Very-Long-Cycle Random-Number Sequences." *BYTE* (mars 1987) : 127–128.

B

Questions fréquemment posées

Cette annexe répond à des questions courantes sur les communications sur réseau et les E/S d'instrument LabVIEW, notamment les E/S GPIB et série.

Questions courantes sur la communication

Cette section répond aux questions courantes sur les communications sur réseau par LabVIEW. Les questions sont divisées en sections en fonction de la plate-forme concernée : Toutes, Windows et Macintosh. Contactez National Instruments si vous avez d'autres questions ou des suggestions concernant LabVIEW.

Questions concernant toutes les plates-formes

Comment dois-je utiliser LabVIEW pour communiquer avec d'autres applications ?

La communication avec d'autres applications, souvent appelée communication inter-processus ou inter-application, peut être réalisée avec les protocoles de mise en réseau standard sur chaque plate-forme. LabVIEW supporte TCP (protocole de gestion de transmission) et UDP (protocole datagramme utilisateur) sur toutes les plates-formes.

(Windows) LabVIEW pour Windows supporte DDE (échange dynamique de données).

(Macintosh) LabVIEW pour Macintosh supporte IAC (communication inter-application). IAC comprend AppleEvents et PPC (communication programme à programme).

(UNIX) LabVIEW pour UNIX ne supporte que TCP et UDP.

Référez-vous à la Partie II, *Interfaces d'E/S*, pour plus d'informations sur l'utilisation de LabVIEW pour communiquer avec d'autres applications. De plus, pour beaucoup d'applications d'instrumentation, les E/S sur fichier fournissent une méthode simple et adéquate pour l'envoi d'informations entre applications.

Comment dois-je lancer une autre application avec LabVIEW ?

Sous Windows et UNIX, utilisez le VI “Exec. système” (`System Exec.vi`) (**Fonctions»Communication**). Sur Macintosh, utilisez le VI AESend Finder Open (**Fonctions»Communication»AppleEvent**).

Quand dois-je utiliser UDP au lieu de TCP ?

En général, UDP est utilisé dans des applications où la fiabilité n’est pas d’une importance cruciale. Par exemple, une application peut transmettre des données informatives à une destination suffisamment souvent pour que la perte de quelques segments de données ne soit pas problématique. UDP peut également être utilisé pour diffuser vers n’importe quel ordinateur souhaitant écouter le serveur.

Quels numéros de port puis-je utiliser avec TCP et UDP ?

Un port est représenté par un nombre compris entre 0 et 65535. Avec UNIX, les numéros de port inférieurs à 1024 sont réservés aux applications privilégiées (par exemple, ftp). Lorsque vous spécifiez un port local, vous pouvez utiliser la valeur 0 ou choisir un port inutilisé.

Pourquoi ne puis-je pas diffuser en utilisant UDP ?

Comme les adresses de diffusion varient entre les domaines, vous devez vous renseigner auprès de l’administrateur de votre système pour connaître l’adresse de diffusion à utiliser. Par exemple, l’adresse de diffusion 0xFFFFFFFF n’est pas correcte pour votre domaine. En outre, votre ordinateur par défaut peut ne pas permettre la diffusion si le processus n’est pas exécuté par l’utilisateur à la source.

Windows uniquement

Quels DLL WinSock puis-je utiliser avec LabVIEW ?

Cette question ne concerne que Windows 3.*x* parce que Windows 95 et Windows NT comprennent ce fichier.

Tout driver WinSock conforme au standard 1.1 devrait fonctionner avec LabVIEW.

Recommandé

National Instruments recommande l'utilisation de DLL WinSock fournie par Microsoft pour Windows pour Workgroups. Avant la mise sur le marché de cette DLL WinSock particulière, National Instruments a testé plusieurs DLL WinSock. Les DLL suivantes ont alors été recommandées.

- TCPOpen version 1.2.2 de Lanera Corporation
- Trumpet version 1.0
- Super-TCP version 3.0 R1 de Frontier Technologies Corporation
- NEWT/Chameleon version 3.11 de NetManage, Inc.

Non recommandé

Les tests limités de ces produits par National Instruments ont généré divers problèmes et pannes lors de tentatives de communication TCP/IP. A l'heure actuelle, nous ne pouvons ni recommander ces produits ni supporter les clients essayant d'établir une communication TCP/IP avec ces DLL WinSock.

- Distinct TCP/IP version 3.1 de Distinct Corporation
- PCTCP version 2.*x* de FTP Software, Inc.

Comment appeler une macro Excel en utilisant DDE ?

Utilisez le VI `DDE Execute`. Ce VI ordonne au serveur DDE d'exécuter une commande dans laquelle vous spécifiez l'action qu'Excel doit mener et le nom de la macro. Assurez-vous d'inclure les parenthèses et les crochets corrects autour de la commande. Référez-vous au *Guide de l'utilisateur d'Excel pour plus d'*informations. Quelques exemples courants sont présentés ci-dessous :

Commande	Action
[RUN("MACRO1")]	Exécute MACRO1
[RUN("MACRO1!R1C1")]	Exécute MACRO1 en commençant à la ligne 1, colonne 1
[OPEN("C:\EXCEL\SURVEY.XLS")]	Ouvre `SURVEY.XLS`

Pourquoi DDE Poke ne fonctionne-t-il pas avec Microsoft Access ?

Microsoft Access ne peut pas accepter de données directement de la part des clients DDE. Pour obtenir des données dans une base de données Access, vous devez créer une macro dans la base de données afin d'importer les données à partir d'un fichier. Dans un cas simple, ces macros ne comportent que deux actions. Faites d'abord un SetWarnings pour supprimer les boîtes de dialogue Access, puis effectuez TransferSpreadsheet ou TransferText pour recevoir les données. Après la définition de cette macro, vous pouvez l'appeler en envoyant un ordre d'exécution à cette base de données, avec le nom de la macro comme donnée. Comme exemple, reportez-vous au VI, "Envoi de données à Access" `(Sending Data to Access.vi)`, situé dans la bibliothèque `examples\network\ access.llb`.

Quelles commandes dois-je utiliser pour communiquer avec une application non LabVIEW en utilisant DDE ?

Les commandes DDE sont spécifiques à l'application avec laquelle vous faites l'interface. Consultez la documentation de l'application pour voir si des commandes sont disponibles.

Comment dois-je installer LabVIEW comme application partagée sur un serveur de fichiers ?

Si vous possédez une licence pour chaque client, suivez ces procédures :

- Installez le système de développement complet LabVIEW sur le serveur. (Sauf s'il y a du matériel NI sur le serveur, il n'est pas nécessaire d'installer NI-DAQ ou `GPIB.DLL`.)
- Chaque ordinateur local doit utiliser son propre fichier `labview.ini` pour les préférences de LabVIEW. Si un fichier `labview.ini` n'existe pas déjà sur un ordinateur local, vous pouvez créer ce document texte (vide) avec un éditeur de texte, tel que le Bloc-Notes de Microsoft. La première ligne de `labview.ini` doit être `[labview]`. Pour définir un paramètre local pour `labview.ini`, LabVIEW a besoin d'un argument de ligne de commande contenant le chemin menant aux préférences. Par exemple, si le fichier `labview.exe` est sur le lecteur `W:\LABVIEW` et que le fichier `labview.ini` est sur `C:\LVWORK` (le disque dur de l'ordinateur local), modifiez l'option de ligne de commande de l'icône LabVIEW dans le Gestionnaire de programmes pour qu'elle devienne

  ```
  W:\LABVIEW\LABVIEW.EXE -pref C:\LVWORK\LABVIEW.INI
  ```

☞ **Remarque** `pref` ***doit être en minuscules. En outre, chaque ordinateur local doit avoir son propre répertoire temporaire LabVIEW, que vous pouvez spécifier dans LabVIEW en choisissant*** **Edition»Préférences...**

- Vous n'avez pas besoin de `GPIB.DLL` sur le serveur, à moins que vous n'utilisiez une carte GPIB sur ce serveur. Vous aurez besoin du fichier `gpibdrv` dans le répertoire LabVIEW. Sur chaque ordinateur qui possède une carte GPIB, vous devrez installer le driver pour cette carte. Utilisez les drivers fournis avec la carte ou réalisez une installation personnalisée de LabVIEW, en n'installant que le driver GPIB désiré sur l'ordinateur local.
- La même procédure pour `GPIB.DLL` s'applique à NI-DAQ.

Sur Windows NT, pourquoi le Client/Serveur DDE Synch se bloque-t-il après plusieurs transferts ?

Il y a quelques problèmes avec DDE dans LabVIEW pour Windows NT qui font que des VIs se bloquent pendant les opérations DDE Poke et DDE Request. Ceci ne concerne que Windows NT.

Macintosh uniquement

Qu'est-ce qu'une ID de destination ?

L'ID de destination est utilisée dans les VIs AppleEvents et PPC sur les Macintosh. Elle sert de référence à l'application que vous essayez de lancer, exécuter ou annuler. Vous pouvez obtenir l'ID de destination d'une application en utilisant un des VIs suivants :

- Le VI Get Target ID prend le nom et la position de l'application comme entrée, la recherche sur le réseau et retourne l'ID de destination.
- Le VI PPC Browser ouvre le menu local d'une boîte de dialogue qui vous permet de sélectionner une application, pouvant être dans le réseau ou sur votre ordinateur.

Vous pouvez utiliser l'ID de destination ainsi obtenue comme une entrée de toutes les fonctions AppleEvents ultérieures pour ouvrir, imprimer, fermer ou exécuter une application.

Pourquoi ne puis-je pas voir mon application dans la boîte de dialogue générée par le VI PPC Browser ?

Si l'application que vous désirez connecter ne peut pas être utilisée avec AppleEvents, elle n'apparaîtra pas dans la boîte de dialogue `Navigateur PPC`. Si vous êtes certain(e) que l'application désirée supporte AppleEvents, ouvrez le menu **Apple** sur l'ordinateur où se trouve l'application. Sélectionnez **Panneaux de configuration»Partage de fichier (MacOS 8)** ou **Panneaux de configuration»Configuration du partage (System 7 et versions précédentes)** et vérifiez que **Lier des programmes (Program Linking)** est activé.

Comment puis-je fermer le Finder après avoir utilisé AppleEvents ?

Utilisez le VI `AESend Quit Application` pour quitter le Finder ou toute autre application.

GPIB

Toutes les plates-formes

Lors de l'utilisation d'un driver d'instrument LabVIEW, j'ai des problèmes pour communiquer avec mon instrument.

Assurez-vous que l'interface GPIB fonctionne correctement. Utilisez l'exemple `LabVIEW<->GPIB.vi` qui se trouve dans la bibliothèque `examples\instr\smplgpib.llb`. Essayez une commande simple pour l'instrument. Par exemple, si l'instrument est de type 488.2, la chaîne de caractères `*IDN?` demandera à l'instrument d'envoyer sa chaîne de caractères d'identification (environ 100 caractères).

Une fois la communication établie, vous ne devriez pas avoir de problèmes avec le driver d'instrument.

Je reçois des erreurs de timeout avec GPIB Read/Write dans LabVIEW.

Essayez d'exécuter un programme simple pour établir la communication entre LabVIEW et GPIB. Utilisez l'exemple `LabVIEW<->GPIB.vi`, qui se trouve dans la bibliothèque `examples\instr\smplgpib.llb`. Essayez une commande simple pour l'instrument. Par exemple, si l'instrument est de type 488.2, la chaîne de caractères `*IDN?` demandera à l'instrument d'envoyer sa chaîne de caractères d'identification (environ 100 caractères).

Si vous recevez encore des messages d'erreur avec GPIB, vous pouvez avoir un problème de configuration. Ouvrez l'utilitaire de configuration GPIB (Windows : `wibconf` ; Mac : `NI-488 Config` ; Sun : `ibconf` ; HP-UX : `ibconf`). Vérifiez que les paramètres correspondent à la configuration de votre matériel. Quittez l'utilitaire de configuration et exécutez l'utilitaire `ibic` (contrôle interactif de bus d'interface) pour votre plate-forme Windows : `wibic` ; (Mac : `ibic`, livré avec votre carte GPIB ; Sun : `ibic` ; HP-UX : `ibic`).

Essayez la séquence suivante :

`: ibfind gpib0`	Trouver l'interface GPIB.
`id = 32000` `gpib0: ibsic`	Réinitialise le bus GPIB. (Envoyer la commande "Réinitialiser l'interface" [Interface Clear].)
`[0130] [cmpl cic atn]`	Opération terminée avec succès.
`gpib0: ibfind dev1`	Trouver l'appareil 1. Utilisez l'adresse d'instrument appropriée.
`id = 32xxx` `dev1: ibwrt "*IDN?"`	Ecrire la chaîne de caractères dans l'instrument. Les instruments 488.2 reconnaissent cette commande et retournent leur chaîne de caractères d'identification.
`count = 5` `dev1: ibrd 100`	Cinq octets ont été envoyés. Vous pouvez lire jusqu'à 100 octets depuis l'instrument.
`Fluke xxx Multimeter…`	L'instrument retourne la chaîne de caractères d'identification.

Si vous avez des erreurs de configuration, vous recevrez un message d'erreur à l'une de ces étapes. Le *Manuel de référence du logiciel NI 488.2* livré avec votre carte GPIB comporte des descriptions détaillées des messages d'erreur.

Pourquoi puis-je communiquer avec mon instrument GPIB avec un VI LabVIEW qui s'exécute en mode Animation mais pas avec un VI qui s'exécute à pleine vitesse ?

C'est sans doute un problème de cadencement. Les VIs s'exécutent beaucoup plus lentement avec l'exécution en mode Animation activée. Votre instrument peut avoir besoin de davantage de temps pour préparer les données à envoyer. Ajoutez une fonction de retard ou utilisez des requêtes de service avant la fonction `GPIB Read` pour donner à l'instrument assez de temps pour générer les données dont il a besoin pour les renvoyer à l'ordinateur.

Pourquoi puis-je écrire dans mon instrument GPIB mais pas lire sa réponse ?

Lorsque la fonction `GPIB Write` s'exécute, l'ordinateur est dans un mode "talk" et l'instrument est en mode "listen". Lorsque la fonction `GPIB Read` s'exécute, le périphérique est supposé passer en mode "talk" et l'ordinateur, en mode "listen". Le système demande au périphérique de changer de mode à l'aide d'un signal de terminaison pouvant être un caractère (End of String) ou une ligne de bus GPIB (End or Identify). Ainsi, si la fonction `GPIB Read` dépasse son délai d'attente ou retourne une erreur EABO (opération abandonnée), cela signifie que le périphérique ne reçoit pas le signal de terminaison correct. Pour déterminer le mode de terminaison pour un instrument donné, reportez-vous à son manuel. De manière empirique, tous les périphériques IEEE 488.2 se terminent par `<CR><LF>` et une assertion de la ligne EOI (End Or Identify - terminer ou identifier) dans le bus GPIB.

Utilisez l'utilitaire de configuration de votre plate-forme (Windows : `wibconf` ; Mac : `NI-488 Config` ; Sun : `ibconf` ; HP-UX : `ibconf`) pour changer le caractère de terminaison.

Un VI qui communique avec un instrument GPIB fonctionne bien sur une plate-forme mais pas sur une autre.

Assurez-vous que l'instrument est configuré correctement dans `wibconf`, `NI-488 Config` ou `ibconf`. Certains instruments 488.1 plus anciens ne se mettent pas automatiquement sur le mode à distance. Passez à l'utilitaire de configuration GPIB de votre plate-forme et définissez le champ `Assert REN when SC (assertion de REN quand SC)`. Cela garantira la mise automatique de l'instrument sur le mode à distance (à l'opposé du mode local) lorsqu'il est adressé.

Windows uniquement

Je peux communiquer avec mon instrument en utilisant un programme Quick Basic mais pas à partir de LabVIEW.

La carte GPIB a des handlers séparés pour DOS et Windows. Quick Basic accède au handler de DOS, mais LabVIEW accède au handler de Windows. Assurez-vous que la carte et le périphérique sont configurés correctement avec `wibconf.exe`.

E/S série

Toutes les plates-formes

Pourquoi mon instrument ne répond-il pas aux commandes envoyées avec le VI “Ecrire sur le port série” (Serial Port Write.vi) ?

De nombreux instruments attendent un retour chariot ou un retour à la ligne pour terminer la chaîne de caractères de commande. Le VI “Ecrire sur le port série” (Serial Port Write.vi) de LabVIEW n’envoie que les caractères inclus dans l’entrée de chaîne de caractères ; aucun caractère de terminaison n’est ajouté à la chaîne de caractères. Beaucoup de progiciels d’émulation de terminal (par exemple, Windows Terminal) ajoutent automatiquement un retour chariot à la fin de toutes les transmissions. Avec LabVIEW, vous devrez inclure le caractère de terminaison correct à votre entrée de chaîne de caractères dans le VI “Ecrire sur le port série” (Serial Port Write.vi) si votre instrument en a besoin.

Certains instruments nécessitent un retour chariot (`\r`) ; d’autres, un retour à la ligne (`\n`). Lorsque vous entrez un retour sur le clavier (sur les claviers de PC, c’est la touche Entrée du pavé alphanumérique principal), LabVIEW insère `\n`. Pour insérer un retour chariot, utilisez la fonction “Concaténer des chaînes de caractères” et ajoutez une constante de retour chariot à la chaîne de caractères ou entrez manuellement `\r` après avoir sélectionné **Affichage des codes '\'** dans le menu local de la chaîne de caractères.

Assurez-vous que votre câble fonctionne. Parmi les questions soumises au support technique, beaucoup sont liées à de mauvais câbles. Dans la communication ordinateur à ordinateur avec E/S série, utilisez un modem nul pour inverser les signaux de réception et de transmission.

Utilisez l’exemple `LabVIEW<->Serial.vi` pour établir une communication avec votre instrument. Il se trouve dans la bibliothèque `examples\instr\smplserl.llb`. Le VI présente également l’utilisation du VI “Octets sur port série” (Bytes at Serial Port.vi) avant la lecture des données du port série.

Comment fermer un port série pour que d'autres applications puissent l'utiliser ?

Vous pouvez souhaiter fermer le port série après son utilisation. Par exemple, sous Windows, un VI peut écrire des informations en utilisant le VI "Ecrire sur le port série" (Serial Port Write.vi) à lpt1, connecté à une imprimante. A la fin de l'opération, LabVIEW a toujours le contrôle du port série. D'autres applications ne peuvent pas utiliser ce port tant que LabVIEW n'a pas relâché son contrôle.

LabVIEW contient le VI "Fermer un driver série" (Close Serial Driver.vi) sur toutes les plates-formes. Ce VI demande à LabVIEW de céder son contrôle du port spécifié. Le VI "Fermer un driver série" (Close Serial Driver.vi) n'est pas dans la palette **Série** ; il est situé dans la bibliothèque `vi.lib\Instr\_sersup.llb`. Pour accéder au VI, utilisez les commandes **Fonctions**»**VI...** ou **Fichier**»**Ouvrir...**

Comment effacer le buffer de port série ?

Lisez les données restantes dans le buffer et ignorez-les.

Comment puis-je ajouter des ports série supplémentaires à mon ordinateur ?

Vous pouvez ajouter des ports série à votre PC compatible IBM en utilisant des cartes d'interface de bus série AT de National Instruments. Si vous utilisez une autre plate-forme, vous pouvez utiliser une carte de tierce partie pour ajouter des ports série à votre ordinateur. Certaines cartes de tierce partie requièrent une interface de langage spécial qui ne se conforme pas au standard API pour les ports série de la plate-forme. Dans ce cas, vous devrez écrire votre propre interface, probablement via un CIN ou une DLL, dans le driver. Les sections suivantes expliquent comment utiliser les VIs de port série de LabVIEW pour adresser des cartes qui utilisent l'interface de port série standard sur des plates-formes différentes.

(Windows 3.*x*) LabVIEW pour Windows utilise l'interface standard Windows de Microsoft pour le port série. Ainsi, les limitations d'utilisation des ports série dans LabVIEW sont dues aux limitations de Windows. Par exemple, comme Windows peut seulement adresser les ports COM1 à COM9, LabVIEW peut seulement adresser ces ports. De la même façon, Windows n'accède qu'à huit ports série à la fois ; donc, LabVIEW ne peut contrôler que huit ports série à la fois.

National Instruments propose désormais des cartes série AT Plug and Play pour une variété de solutions pour les communications série. Les cartes d'interface série asynchrones AT-485 et AT-232 sont disponibles dans des

configurations à 2 ou 4 ports. La pleine compatibilité Plug and Play vous donne les bénéfices d'une configuration sans commutateur, pour une installation et une maintenance simplifiées. Les cartes AT-485 et AT-232 comportent les composantes logicielles suivantes pour une utilisation avec Windows : driver de périphérique, test de diagnostic de matériel et utilitaire de configuration. Ces cartes ont été testées avec LabVIEW pour Windows. Contactez National Instruments pour plus d'informations :

Fabricant :	National Instruments
Nom du produit :	AT-485 et AT-232
Téléphone :	01 48 14 24 24
Télécopie :	01 48 14 24 14
FaxBack (Etats-Unis) :	512 418 1111 numéro de commande 1430

Une carte de tierce partie utilisant des moyens de dépannage pour dépasser les limitations de Windows ne fonctionnera probablement pas bien, ou pas du tout, avec LabVIEW. Il est possible d'écrire un CIN ou une DLL pour accéder à de telles cartes, mais ce n'est pas recommandé. En général, les cartes qui utilisent l'interface Windows standard doivent être complètement compatibles avec LabVIEW.

(Windows 95 et Windows NT) Contrairement à Windows 3.*x*, Windows 95 et Windows NT ne se limitent pas à huit ports série. Sous Windows 95 et Windows NT, LabVIEW peut adresser 256 ports série maximum. Le paramètre de numéro de port par défaut est 0 pour COM1, 1 pour COM2, 2 pour COM3, etc.

Le fichier `labview.ini contient les options de` configuration `de` LabVIEW. Pour configurer les périphériques qui seront utilisés par les VIs de ports série, définissez l'option de configuration `labview.serialDevices` sur la liste de périphériques à utiliser. Par exemple, pour configurer vos périphériques d'une façon semblable à Windows 3.*x*, écrivez la ligne :

```
serialDevices="COM1;COM2;COM3;COM4;COM5;COM6;COM7;COM8;
COM9;\\.\COM10;\\.\COM11;\\.\COM12;\\.\COM13;\\.\COM14;
\\.\COM15;\\.\COM16;LPT1;LPT2"
```

La ligne précédente doit apparaître comme une ligne unique dans votre configuration.

(Macintosh) LabVIEW utilise des INIT système standard pour communiquer avec les ports série. Par défaut, LabVIEW utilise les drivers `.aIn` et `.aOut`, le port de modem, comme le port 0 et les drivers `.bIn` et `.bOut`, le port d'imprimante, pour le port 1. Pour accéder à des ports

supplémentaires, vous devez installer des cartes supplémentaires avec les INIT qui les accompagnent.

Une fois que les cartes et les INIT sont installés, indiquez à LabVIEW comment utiliser les ports supplémentaires. La méthode utilisée dans LabVIEW 5.0 est légèrement différente de celle utilisée dans les versions précédentes.

Modifiez le VI serpOpen.vi global (situé dans la bibliothèque `vi.lib\Instr\_sersup.llb`) pour prendre en charge les ports supplémentaires. Tous les VIs de ports série appellent le VI “Ouvrir un driver série” (Open Serial Driver.vi), qui lit l’entrée appropriée et/ou les noms de drivers de sortie dans le VI serpOpen.vi. La face-avant du VI serpOpen.vi contient deux tableaux de chaînes de caractères, appelés noms de drivers d’entrée et noms de drivers de sortie.

Lorsque le VI “Ouvrir un driver série” (Open Serial Driver.vi) est appelé, il utilise l’entrée de numéro de port comme l’indice des tableaux de chaînes de caractères. Ainsi, pour permettre à LabVIEW de reconnaître des ports série supplémentaires, ajoutez des éléments supplémentaires de la chaîne de caractères dans les noms de drivers d’entrée et de sortie, puis sélectionnez **Prendre la valeur actuelle par défaut** et enregistrez les changements du VI serpOpen.vi.

Chaque port série d’une carte enfichable possède deux noms, un pour l’entrée et un pour la sortie. Les noms et les instructions exacts pour l’installation des drivers sont dans la documentation de la carte. Contactez le fabricant de la carte si les instructions manquent ou sont peu claires.

Les cartes suivantes fonctionnent avec LabVIEW pour Macintosh :

Fabricant :	Creative Solutions, Inc.
Nom du produit :	Hurdler HQS (4 ports) ou HDS (2 ports)
Téléphone :	301 984 0262
Télécopie :	301 770 1675

Fabricant :	**Greensprings**
Nom du produit :	RM 1280 (4 ports)
Téléphone :	415 327 1200
Télécopie :	415 327 3808

(UNIX) Sur un Sun SPARCstation sous Solaris 1 et sur Concurrent PowerMAX, le paramètre **numéro de port** pour les VIs de port série est 0 pour `/dev/ttya`, 1 pour `/dev/ttyb`, etc. Sous Solaris 2, le port 0 fait référence à `/dev/cua/a, 1 à /dev/cua/b`, etc. Sous HP-UX, le numéro de port 0 fait référence à `/dev/tty00`, 1 à `/dev/tty01`, etc.

Sur Concurrent PowerMAX, le port 0 fait référence à `/dev/console`, le port 1 fait référence à `/dev/tty1`, le port 2 fait référence à `/dev/tty2`, etc.

Comme des cartes de ports série de fabricants différents peuvent porter des noms de périphériques arbitraires, LabVIEW a développé une interface pratique pour faciliter la numérotation de ports. Dans LabVIEW pour Sun, HP-UX et Concurrent PowerMAX, il existe une option de configuration permettant à LabVIEW d'adresser les ports série. LabVIEW supporte toute carte qui utilise les périphériques UNIX standard. Certains fabricants suggèrent l'utilisation des nœuds de périphériques `cua` plutôt que `tty` avec leurs cartes. LabVIEW peut adresser les deux types de nœuds.

Le fichier `.labviewrc` contient les options de configuration de LabVIEW. Pour définir les périphériques à utiliser par les VIs de port série, mettez l'option de configuration `labview.serialDevices` sur la liste de périphériques que vous avez l'intention d'utiliser.

Par exemple, le paramètre par défaut est :

```
labview.serialDevices:/dev/ttya:/dev/ttyb:/dev/
ttyc:...:/dev/ttyz.
```

☞ **Remarque** ***Pour cela, toute installation de carte série de tierce partie doit comporter une méthode de création de*** `standard/dev file` ***(nœud) et l'utilisateur doit connaître le nom de ce fichier.***

Les cartes suivantes doivent fonctionner avec LabVIEW pour Sun :

Fabricant :	**Sun**
Nom du produit :	**Contrôleur parallèle série SBus** (8 série, 1 parallèle)
Fabricant :	**Artecon, Inc.**
Nom du produit :	**Carte** SBus ArtePort **SUNX-SB-300P** avec 3 ports série/1 port parallèle **Carte** SBus ArtePort SUNX-SB-400P avec 4 ports série/1 port parallèle **Carte** SBus ArtePort SUNX-SB-1600 avec 16 ports série

Pour chacun de ces produits, contactez SunExpress au (800) 873-7869.

Comment puis-je contrôler les lignes série DTR et RTS ?

Le VI “Initialiser le port série” (Serial Port Init.vi) peut être utilisé pour configurer le port série pour le handshaking de matériel ; toutefois, certaines applications peuvent demander un basculement manuel des lignes DTR et RTS. Vu que l’interface des ports série est dépendante de la plate-forme, chaque plate-forme a un mécanisme séparé pour contrôler les lignes.

(Windows) LabVIEW pour Windows contient un VI que vous pouvez utiliser pour gérer les lignes série DTR et RTS. Le VI “Contrôler des lignes série” (serial line ctrl.vi), situé dans la bibliothèque `vi.lib\Instr\_sersup.llb`, peut être utilisé pour contrôler ces lignes. Le VI bascule ces lignes selon l’entrée de la fonction. Les codes valides de l’entrée sont :

0 noop (pas d’opération)
1 clear DTR (désactiver DTR)
2 set DTR (activer DTR)
3 clear RTS (désactiver RTS)
4 set RTS (activer RTS)
5 set DTR protocol (activer le protocole DTR)
6 clr DTR protocol (désactiver le protocole DTR)
7 noop2 (pas d’opération)

(Macintosh) Sur Macintosh, vous pouvez utiliser la fonction Device Control/ Status pour contrôler le port série. Le manuel intitulé *Inside Macintosh* (reportez-vous au Volume II, pages 245–259 et au Volume IV, pages 225–228) contient des informations spécifiques sur les codes qui peuvent être envoyés au port série. Un résumé des codes et de leurs fonctions est listé ci-après :

Code	Paramètre	Effet
13	baudRate	Définit le débit de transmission en baud (vitesse réelle, en nombre entier)
14	serShk	Définit les paramètres de handshaking
16	octet	Définit diverses options de contrôle
17		Active DTR
18		Désactive DTR
19	caractère	Remplace les erreurs de parité
20	2 car.	Remplace les erreurs de parité par des caractères
21		Active de manière inconditionnelle XOff pour le contrôle de flux de sortie
22		Désactive de manière inconditionnelle XOff pour le contrôle de flux de sortie
23		Envoie XOn pour le contrôle de flux d'entrée si XOff a été envoyé en dernier
24		Envoie de manière inconditionnelle XOn pour le contrôle de flux d'entrée
25		Envoie XOff pour le contrôle de flux d'entrée si XOn a été envoyé en dernier
26		Envoie de manière inconditionnelle XOff pour le contrôle de flux d'entrée
27		Remet le canal SCC à zéro

(Sun) LabVIEW pour Sun ne contient pas de support spécifique pour basculer les lignes de handshaking matériel des ports série. Pour contrôler ces lignes manuellement, vous devez écrire un CIN. Reportez-vous à la section *Steps for Creating a CIN* du chapitre 1, *CIN Overview*, dans le *LabVIEW Code Interface Reference Manual*, disponible au format PDF (Portable Document Format - format de document portable), pour plus de précisions sur la façon d'écrire un CIN.

Pourquoi ne puis-je pas affecter un buffer série supérieur à 32 Koctets ?

Avec le VI "Initialiser le port série" (Serial Port Init.vi), vous ne pouvez pas utiliser une taille de buffer supérieure à 32 ko car Windows et Macintosh limitent le buffer de port série à 32 ko ; c'est pourquoi, si vous affectez un buffer supérieur à cette valeur, LabVIEW tronque la taille du buffer à 32 ko. Ce problème n'existe pas sur Sun.

Windows uniquement

Comment puis-je accéder au port parallèle ?

Dans LabVIEW pour Windows 3.*x*, le port 10 est LPT1, le port 11 est LPT2, etc. Sous Windows 95/NT, vous pouvez définir un port sur LPT1 dans Périphériques série. Pour envoyer des données sur une imprimante connectée à un port parallèle, utilisez le VI "Ecrire sur un port série" (Serial Port Write.vi).

Que signifient les numéros d'erreurs reçus des VIs de ports série ?

Les VIs de ports série de LabVIEW pour Windows retournent les erreurs reportées par la fonction "Windows GetCommError". Les numéros d'erreurs retournés par les VIs de ports série sont le "ou logique" de 0x4000 (16.384) et des numéros d'erreurs du tableau suivant. Remarquez que l'erreur retournée reflète l'état du port série ; l'erreur peut avoir été générée

par une fonction de port série précédente. Les valeurs retournées peuvent être une combinaison des erreurs suivantes :

Valeur hexadécimale	Nom de l'erreur	En-tête
0x0001	CE_RXOVER	La file d'attente de réception a dépassé sa limite. Il n'y avait plus de place dans la file d'attente d'entrée ou un caractère a été reçu après le caractère de fin de fichier (EOF).
0x0002	CE_OVERRUN	Un caractère n'a pas été lu dans le matériel avant que le prochain caractère n'arrive. Le caractère a été perdu.
0x0004	CE_RXPARITY	Le matériel a détecté une erreur de parité.
0x0008	CE_FRAME	Le matériel a détecté une erreur de cadrage.
0x0010	CE_BREAK	Le matériel a détecté une condition d'arrêt.
0x0020	CE_CTSTO	Timeout de CTS (clear-to-send). Lors de la transmission d'un caractère, CTS était bas pour la durée spécifiée par l'élément fCtsHold de COMSTAT.
0x0040	CE_DSRTO	Timeout de DSR (data-set-ready). Lors de la transmission d'un caractère, DSR était bas pour la durée spécifiée par l'élément fDsrHold de COMSTAT.

Valeur hexadécimale	Nom de l'erreur	En-tête
0x0080	CE_RLSDTO	Timeout de RLSD (receive-line-signal-detect). Lors de la transmission d'un caractère, RLSD était bas pour la durée spécifiée par l'élément fRlsdHold de COMSTAT.
0x0100	CE_TXFULL	La file d'attente de transmission était pleine lorsqu'une fonction a essayé de mettre un caractère dans la file.
0x0200	CE_PTO	Un timeout s'est produit lors d'une tentative de communication avec un périphérique parallèle.
0x0400	CE_IOE	Une erreur d'E/S s'est produite lors d'une tentative de communication avec un périphérique parallèle.
0x0800	CE_DNS	Aucun périphérique parallèle n'a été sélectionné.
0x1000	CE_OOP	Un périphérique parallèle a signalé qu'il n'a plus de papier.
0x8000	CE_MODE	Le mode demandé n'est pas supporté ou le paramètre `idComDev` est invalide. Si définie, CE_MODE est la seule erreur valide.

Pour utiliser ce tableau, prenez les numéros d'erreurs et décomposez-les en leurs composantes d'erreur. Par exemple, si le VI "Ecrire sur le port série" (Serial Port Write.vi) retourne l'erreur16.408, les erreurs retournées sont `CE_BREAK et CE_FRAME` (16.408 = 16.384 + 16 + 8 = 0x4000 + 0x0010 + 0x0008).

Sun uniquement

Je reçois une erreur –37 lorsque je réalise une E/S série.

L'erreur –37 signifie que LabVIEW ne peut pas trouver le périphérique série approprié. Ceci indique que a) les fichiers `/dev/tty?` n'existent pas sur votre ordinateur ou b) LabVIEW ne parvient pas à trouver `serpdrv`.

Par défaut, LabVIEW adresse `/dev/ttya` comme port 0, `/dev/ttyb` comme port 1, etc. Ces périphériques doivent exister et l'utilisateur doit avoir les permissions de lecture et d'écriture pour accéder à ces périphériques. Vous pouvez modifier les accès de LabVIEW aux périphériques avec les VIs de ports série en ajoutant l'option de configuration serialDevices à votre fichier `.Xdefaults`. Reportez-vous aux *Notes d'informations LabVIEW* pour l'utilisation de cette option.

Le fichier `serpdrv` est envoyé avec LabVIEW et sert d'interface entre LabVIEW et les ports série Sun. Ce fichier doit se trouver à l'emplacement spécifié par l'option de configuration libdir, défini sur le répertoire LabVIEW par défaut. Cela signifie que `serpdrv` doit se trouver dans le même répertoire que `gpibdrv` et `vi.lib`.

L'E/S série se bloque sur un ordinateur Solaris 1.*x*.

LabVIEW pour Sun utilise des appels d'E/S asynchrones lors d'opérations de port série. Dans le kernel `Generic_Small`, une E/S asynchrone a été constatée. Pour accéder aux ports série depuis LabVIEW pour Sun, l'utilisateur doit utiliser le kernel Generic standard (et non `Generic_Small`) ou reconstruire le kernel `Generic_Small` et réinitialiser le SPARC.

C

Informations à l’attention du client

Cette annexe contient les formulaires qui vous permettront de réunir les renseignements nécessaires pour vous aider à résoudre vos problèmes techniques, ainsi qu’une fiche que vous pouvez utiliser pour nous signaler vos remarques sur nos documentations de produits. Lorsque vous nous contactez, veuillez nous fournir les renseignements contenus dans le formulaire de support technique et, le cas échéant, dans la fiche de configuration de votre système de façon à ce que nous puissions répondre à vos questions dans les plus brefs délais.

National Instruments offre une assistance technique par téléphone, télécopie et par l’intermédiaire de systèmes électroniques pour que vous puissiez obtenir rapidement l’information que vous souhaitez. Nos services de support électroniques comprennent un BBS, un site FTP, un système Fax-on-demand et un service de courrier électronique. Si vous rencontrez des problèmes de matériel ou de logiciel, essayez tout d’abord les services de support électroniques. Si les renseignements fournis par ces systèmes sont insuffisants, nos centres de support technique pourvus d’ingénieurs d’application, offrent également un support téléphonique et par télécopie.

Services électroniques

Support par site FTP

Pour accéder à notre site FTP, ouvrez une session sur l’hôte Internet, `ftp.natinst.com`, comme `anonyme` et utilisez votre adresse Internet, par exemple `pauldupont@ailleurs.com`, comme mot de passe. Les fichiers et documents de support se trouvent dans les répertoires `/support`.

Support par le système Fax-on-Demand

Le système Fax-on-Demand est un système de recherche documentaire accessible 24 heures sur 24 contenant une bibliothèque documentaire portant sur une vaste gamme d’informations techniques. Vous pouvez accéder au système Fax-on-Demand avec un téléphone à touches en composant le 512 418 1111.

Support par courrier électronique (disponible aux Etats-Unis seulement)

Vous pouvez poser vos questions techniques à l’équipe d’ingénieurs d’applications par courrier électronique à l’adresse Internet indiquée ci-dessous. N’oubliez pas de faire figurer vos nom, adresse et numéro de téléphone, de façon à ce que nous puissions vous communiquer des solutions et suggestions.

`support@natinst.com`

Support par affichage électronique

National Instruments dispose de sites BBS et FTP spécialisés en support technique accessible 24 heures sur 24, contenant un large éventail de fichiers et de documents pour répondre aux questions les plus courantes de ses clients. Ces sites vous permettent également de télécharger les tout derniers drivers d'instruments, mises à jour et programmes d'exemple. Pour obtenir des instructions enregistrées sur la façon d'utiliser les services BBS et FTP, et pour obtenir les services d'informations automatisés BBS, composez le 512 795 6990. Vous pouvez accéder à ces services de la manière suivante :

En France : 01 48 65 15 59
Jusqu'à 9,600 bauds, 8 bits de données, 1 bit d'arrêt, aucune parité

Aux Etats-Unis : 512 794 5422
Jusqu'à 14,400 bauds, 8 bits de données, 1 bit d'arrêt, aucune parité

Au Royaume-Uni : 01635 551422
Jusqu'à 9,600 bauds, 8 bits de données, 1 bit d'arrêt, aucune parité

Support par téléphone et télécopie

National Instruments dispose de bureaux répartis dans le monde entier. Utilisez la liste suivante pour localiser le numéro de support technique de votre pays. Si National Instruments n'a pas de bureau de représentation dans votre pays, contactez le revendeur de votre logiciel pour obtenir de l'aide.

Pays	Téléphone	Télécopie
Allemagne	089 741 31 30	089 714 60 35
Australie	03 9879 5166	03 9879 6277
Autriche	0662 45 79 90 0	0662 45 79 90 19
Belgique	02 757 00 20	02 757 03 11
Brésil	011 288 3336	011 288 8528
Canada (Ontario)	905 785 0085	905 785 0086
Canada (Québec)	514 694 8521	514 694 4399
Corée	02 596 7456	02 596 7455
Danemark	45 76 26 00	45 76 26 02
Espagne	91 640 0085	91 640 0533
Etats-Unis	512 795 8248	512 794 5678
Finlande	09 725 725 11	09 725 725 55
France	01 48 14 24 24	01 48 14 24 14
Hong Kong	2645 3186	2686 8505
Israël	03 6120092	03 6120095
Italie	02 413091	02 41309215
Japon	03 5472 2970	03 5472 2977
Mexique	5 520 2635	5 520 3282
Norvège	32 84 84 00	32 84 86 00
Pays-bas	0348 433466	0348 430673
Royaume-Uni	01635 523545	01635 523154
Singapour	2265886	2265887
Suède	08 730 49 70	08 730 43 70
Suisse	056 200 51 51	056 200 51 55
Taïwan	02 377 1200	02 737 4644

Formulaire de support technique

Photocopiez ce formulaire et mettez-le à jour chaque fois que vous faites des changements d'ordre logiciel ou matériel, et utilisez la copie de ce formulaire complété comme référence pour votre configuration actuelle. Complétez ce formulaire avec soin avant de contacter National Instruments pour obtenir le support technique nécessaire, afin de permettre à nos ingénieurs d'application de répondre plus efficacement à vos questions.

Si vous utilisez des produits logiciels et matériels de National Instruments ayant rapport au problème, ajoutez les fiches de configuration se trouvant dans les manuels de l'utilisateur correspondants. Ajoutez toute page supplémentaire si nécessaire.

Nom_____

Société_____

Adresse _____

_____

Téléphone (___) _____Télécopie (___) _____

Marque d'ordinateur _____ Modèle _____Processeur_____

Système d'exploitation (inclure le numéro de version) _____

Fréquence d'horloge _____MHz RAM _____Mo Carte graphique _____

Souris ___oui ___non Autres cartes installées _____

Capacité du disque dur _____ Mo Marque_____

Instruments utilisés _____

_____

Modèle de produit matériel de National Instruments _____ Révision _____

Configuration _____

Produit logiciel de National Instruments_____ Version _____

Configuration _____

Description du problème :_____

_____

_____

_____

_____

Listez tout message d'erreur : _____

_____

_____

Le problème apparaît chaque fois que les opérations suivantes sont effectuées : _____

_____

_____

_____

_____

Fiche de configuration logicielle et matérielle LabVIEW

Inscrivez les paramètres et les modifications apportées à votre matériel et à votre logiciel. Complétez une nouvelle copie de ce formulaire chaque fois que vous modifiez la configuration de votre logiciel ou de votre matériel, et utilisez ce formulaire comme référence pour votre configuration actuelle. Complétez ce formulaire avec soin avant de contacter National Instruments pour obtenir un support technique afin de permettre à nos ingénieurs d'application de répondre plus efficacement à vos questions.

Produits de National Instruments

Matériel DAQ _____

Niveau d'interruption du matériel (IRQ) _____

Canaux DMA du matériel _____

Adresse de base d'E/S du matériel _____

Choix de programmation_____

Version NI-DAQ ou LabVIEW _____

Autres cartes dans le système _____

Adresse de base d'E/S des autres cartes _____

Canaux DMA des autres cartes _____

Niveau d'interruption des autres cartes _____

Autres produits

Marque et modèle de l'ordinateur _____

Microprocesseur _____

Fréquence ou vitesse d'horloge_____

Type de carte vidéo installée _____

Version du système d'exploitation _____

Mode du système d'exploitation _____

Langage de programmation _____

Version du langage de programmation _____

Autres cartes présentes dans le système _____

Adresse de base d'E/S des autres cartes _____

Canaux DMA des autres cartes _____

Niveau d'interruption des autres cartes _____

Vos commentaires sur la documentation…

National Instruments vous encourage à nous signaler vos remarques sur les documentations livrées avec nos produits. Ces informations nous aideront à vous garantir la qualité que vous attendez de nos produits.

Titre : *Manuel de l'utilisateur LabVIEW™*

Date d'édition : Juillet 1998

Référence : 321190B-01

Veuillez apporter des commentaires sur l'exhaustivité, la clarté et l'organisation de ce manuel :

_____

_____

_____

_____

_____

_____

Si vous avez trouvé des erreurs, indiquez les numéros de pages correspondants et décrivez la nature de ces erreurs :

_____

_____

_____

_____

_____

_____

Nous vous remercions de votre contribution.

Nom _____

Fonction _____

Société _____

Adresse _____

_____

Adresse électronique _____

Téléphone (___) _____ Télécopie (___) _____

A envoyer à : National Instruments
Centre d'Affaires Paris-Nord
Immeuble "Le Continental"
BP 217
93153 Le Blanc-Mesnil Cedex

A télécopier au : numéro de fax de
National Instruments France
01 48 14 24 14

Glossaire

Préfixe	Signification	Valeur
n-	nano-	10^{-9}
m-	milli-	10^{-3}
µ-	micro-	10^{-6}

Nombres/ Symboles

•	Infini.
D	Delta. Différence. Δx désigne la valeur dont x change d'un indice au suivant.
p	Pi.
1D	Unidimensionnel.
2D	Bidimensionnel.

A

à deux dimensions	Possédant deux dimensions, comme un tableau ayant des lignes et des colonnes.
acquisition de données	DAQ. Processus d'acquisition de données, généralement par l'intermédiaire de cartes enfichables d'entrée numérique ou analogique.
adapter	Changer le descripteur de type d'un élément de données sans affecter l'image mémoire des données.
Aide	Instructions en ligne qui expliquent comment utiliser une application Windows. Le menu **Aide** affiche des sujets d'aide spécifiques. Appuyez sur `<F1>` pour afficher une liste des sujets d'aide.
A/N	Analogique/numérique.

ANSI	American National Standards Institute (Institut national des normes, aux Etats-Unis).
architecture logicielle d'instruments virtuels	Bibliothèque d'interface unique pour le contrôle GPIB, VXI, RS-232 et pour d'autres types d'instruments.
ASCII	American Standard Code for Information Interchange (code américain standard pour l'échange d'informations).
ATE	Automated Test Equipment (Equipement de test automatisé).
auto-indexation	Capacité des structures de boucle à désassembler et assembler des tableaux à leurs bords. Lorsqu'un tableau entre dans une boucle alors que l'auto-indexation est active, la boucle le désassemble automatiquement, c'est-à-dire que les scalaires sont extraits des tableaux à une dimension, les tableaux à une dimension sont extraits des tableaux à deux dimensions, et ainsi de suite. A la fin de leur exécution, les boucles assemblent les données dans des tableaux en suivant la procédure inverse.

B

barre de menus	Barre horizontale qui contient les noms des menus principaux. La barre de menus apparaît sous la barre de titre d'une fenêtre. Chaque application possède une barre de menus distincte, alors que certains menus (et commandes) sont communs à plusieurs applications.
barre d'outils	Barre contenant des boutons de commande que vous pouvez utiliser pour exécuter et mettre au point des VIs.
bibliothèque de VIs	Fichier spécial qui contient une collection de VIs apparentés pour un usage spécifique.
BNF	Backus-Naur Form. Représentation courante pour la grammaire des langages en informatique.
boîte de calcul	Nœud qui exécute des formules que vous entrez sous forme de texte. Particulièrement utile pour les formules longues et trop complexes à représenter sous forme de diagramme.
boîte de description	Documentation en ligne pour un objet du G.
boîte de dialogue	Ecran interactif contenant des messages, dans lequel vous pouvez donner des informations supplémentaires nécessaires pour exécuter une commande.

boucle For — Structure de boucle itérative qui exécute son sous-diagramme un nombre de fois défini. Equivalent au code conventionnel :
`For i=0 to n-1, do ....`

boucle While — Structure de boucle qui répète une section de code jusqu'à ce qu'une condition soit remplie. Comparable à une boucle Do ou à une boucle Repeat-Until dans les langages de programmation conventionnels.

bouton Flèche brisée — Bouton qui remplace le bouton Exécution lorsqu'un VI ne peut pas s'exécuter à cause d'erreurs. Cliquez sur ce bouton pour appeler la boîte de dialogue Liste d'erreurs.

branche de câblage — Section de câblage qui contient tous les segments de câblage entre deux jonctions, entre un terminal et une jonction, ou entre deux terminaux, si aucune jonction n'existe entre les terminaux.

C

cadre connecteur — Zone située dans l'angle supérieur droit de la fenêtre d'une face-avant et qui affiche les terminaux du VI. Il est sous-jacent au cadre icône.

cadre icône — Zone dans l'angle supérieur droit de la face-avant et du diagramme qui affiche l'icône du VI.

capteur — Périphérique qui produit une sortie de tension ou de courant représentant la mesure d'une propriété physique, telle que la vitesse, la température ou le flux.

caractères non affichables — Caractères ASCII qui ne peuvent pas être affichés, tels que les retours à la ligne, les tabulations, etc.

case à cocher — Dans une boîte de dialogue, petite case carrée pouvant être sélectionnée ou désélectionnée. Les cases à cocher sont en général utilisées lorsque plusieurs options peuvent être définies. Il est possible de sélectionner plusieurs cases à cocher.

chemin absolu — Chemin de fichier ou répertoire qui décrit sa position par rapport au niveau principal du système de fichiers.

CIN — *Voir* Code Interface Node.

clonage	Faire une copie d'une commande ou d'un autre objet G en cliquant sur le bouton de la souris tout en appuyant sur la touche <Ctrl> **(Windows)**, <option> **(Macintosh)**, <meta> **(Sun)** ou <Alt> **(HP-UNIX)** et en faisant glisser la copie jusqu'à son nouvel emplacement.
cluster	Ensemble ordonné et non indexé d'éléments de données de n'importe quel type, y compris des nombres, des booléens, des chaînes de caractères, des tableaux ou des clusters. Les éléments doivent être tous des commandes ou tous des indicateurs.
code interface node (CIN)	CIN. Nœud de diagramme spécial par l'intermédiaire duquel vous pouvez lier du code textuel conventionnel à un VI.
coercition	Conversion automatique que le G effectue pour changer la représentation numérique d'un élément de données.
commande	Objet de la face-avant utilisé pour entrer des données dans un VI de façon interactive ou dans un sous-VI par l'intermédiaire d'un programme.
commandes et indicateurs booléens	Objets de la face-avant utilisés pour manipuler et afficher des données booléennes (TRUE ou FALSE). Plusieurs styles sont disponibles : des interrupteurs, des boutons et des LED.
commandes et indicateurs de chaîne de caractères	Objets de la face-avant utilisés pour manipuler et afficher, ou entrer et sortir du texte.
commandes et indicateurs numériques	Objets de face-avant utilisés pour manipuler et afficher, ou entrer et sortir, des données numériques.
commandes et indicateurs PICT personnalisés	Commandes et indicateurs dont des parties peuvent être remplacées par des graphiques et des indicateurs que vous fournissez.
commande de menu déroulant	Commande numérique spéciale qui associe des entiers 32 bits, en commençant par 0 et en augmentant progressivement, avec une série d'étiquettes de texte ou de graphiques.
commande de type graphe	Objet de la face-avant qui affiche des données dans un plan cartésien.
compiler	Processus qui convertit du code de haut niveau en code machine exécutable. Les VIs sont compilés automatiquement avant d'être exécutés pour la première fois après leur création ou après toute modification.

condition	Sous-diagramme d'une structure Condition.
connecteur	Partie du nœud d'un VI ou d'une fonction qui contient ses terminaux d'entrée et de sortie. Les données sont transmises vers le nœud et reçues de celui-ci par l'intermédiaire d'un connecteur.
constante	*Voir* constante universelle et constante définie par l'utilisateur.
constante définie par l'utilisateur	Objet de diagramme qui émet une valeur que vous définissez.
constante universelle	Objet de diagramme non éditable qui émet un caractère ASCII particulier ou une constante numérique standard, par exemple, π.
conversion	Changement de type d'un élément de données.
CPU	Central Processing Unit (unité de calcul).

D

DAQ	*Voir* acquisition de données.
DDE	*Voir* échange dynamique de données.
dépendance de données	Dans un langage de programmation par flux de données, condition dans laquelle un nœud ne peut pas s'exécuter tant qu'il ne reçoit pas des données d'un autre nœud. *Voir aussi* dépendance de données artificielle.
dépendance de données artificielle	Condition dans un langage de programmation à flux de données dans laquelle l'arrivée d'une donnée, plutôt que sa valeur, déclenche l'exécution d'un nœud.
descripteur de type	*Voir* descripteur de type de données.
descripteur de type de données	Code qui identifie les types de données, utilisé pour le stockage et la représentation des données.
développeur de système	Créateur du logiciel d'application qui doit être exécuté.
diagramme	Description ou représentation graphique d'un programme ou d'un algorithme. En G, le diagramme, qui est constitué d'icônes exécutables appelées nœuds et de fils de liaison qui transmettent les données entre les nœuds, est le code source du VI.

dimension	Attribut de taille et de structure d'un tableau.
dimensionnement automatique	Redimensionnement automatique des étiquettes pour s'adapter au texte que vous entrez.
données aplaties	Données de n'importe quel type qui ont été converties en une chaîne de caractères, généralement pour être écrites dans un fichier.
driver d'instrument	VI qui commande un instrument programmable.
DUT	Device under test (Périphérique en cours de test).

E

échange dynamique de données	DDE. Echange de données entre des applications, accompli sans l'implication ni le contrôle de l'utilisateur.
échelle	Partie des commandes et indicateurs à action mécanique, des graphes et des graphes déroulants, qui contient une série de repères ou de points à des intervalles connus pour désigner des unités de mesure.
Editeur d'icône	Interface similaire à celle d'un programme de peinture pour créer des icônes de VIs.
emplacement	Partie inamovible des commandes et indicateurs de face-avant, qui contient des glissières et des échelles.
enregistrement de données	Acquisition et stockage simultanés de données dans un fichier sur disque. Les fonctions d'E/S sur fichier du G peuvent enregistrer des données.
EOF	End-of-File. Offset du caractère de fin de fichier par rapport au début du fichier (c'est-à-dire, l'EOF représente la taille du fichier).
E/S	Entrée/sortie. Transfert de données vers ou à partir d'un système informatique impliquant des canaux de communication, des périphériques d'entrée utilisateur et/ou des interfaces d'acquisition de données et de commande.
étape	Sous-diagramme d'une structure Séquence.
étiquette	Objet de texte utilisé pour nommer ou décrire d'autres objets ou zones sur la face-avant ou le diagramme.

exécution asynchrone — Mode dans lequel plusieurs processus se partagent le temps du processeur. Par exemple, un processus s'exécute pendant que d'autres attendent des interruptions au cours d'une E/S de périphérique ou pendant qu'elles attendent une impulsion d'horloge.

exécution en continu — Mode d'exécution dans lequel un VI est exécuté indéfiniment jusqu'à ce que l'opérateur l'arrête. Vous l'activez en cliquant sur le bouton **Exécution en continu**.

exécution intégrée — Aptitude d'une fonction ou d'un VI à réutiliser de la mémoire au lieu d'en allouer davantage.

exécution pilotée par tableau — Méthode d'exécution dans laquelle les tâches individuelles sont des conditions séparées dans une structure Condition qui est imbriquée dans une boucle While. Les séquences sont spécifiées sous forme de tableaux de numéros de conditions.

exécution ré-entrante — Mode dans lequel des appels à plusieurs instances d'un sous-VI peuvent s'exécuter en parallèle avec un stockage de données distinct et séparé.

exécution en mode Animation — Fonction qui anime l'exécution du VI pour illustrer le flux de données dans le VI.

F

face-avant — Interface utilisateur interactive d'un VI. Modélisée à partir de la face-avant d'instruments physiques, elle est composée d'interrupteurs, de glissières, de vu-mètres, de graphes, de graphes déroulants, de jauges, de LED et d'autres commandes et indicateurs.

fenêtre active — Fenêtre qui est actuellement configurée pour accepter les entrées de l'utilisateur, généralement la fenêtre qui se trouve au premier plan. La barre de titre d'une fenêtre active est en surbrillance. Vous rendez une fenêtre active en cliquant dessus, ou en la sélectionnant dans le menu **Fenêtres**.

fenêtre d'aide — Fenêtre spéciale qui affiche les noms et emplacements des terminaux d'une fonction ou d'un sous-VI, la description des commandes et indicateurs, les valeurs des constantes universelles et les descriptions et les types de données des attributs des commandes.

fenêtre Hiérarchie — Fenêtre qui affiche graphiquement la hiérarchie des VIs et des sous-VIs.

FFT — Fast Fourier transform (Transformée de Fourier rapide).

fichier d'enregistrement de données	Fichier qui stocke des données sous forme d'une séquence d'enregistrements d'un type de données arbitraire unique que vous devez spécifier lorsque vous créez le fichier. Alors que tous les enregistrements d'un fichier d'enregistrement de données doivent être du même type, ce type peut être complexe ; par exemple, vous pouvez spécifier que chaque enregistrement est un cluster contenant une chaîne de caractères, un nombre et un tableau.
fichier de structure d'octet	Fichier qui stocke des données sous forme de séquence de caractères ASCII ou d'octets.
fil de liaison	Chemin de données entre nœuds. *Voir aussi* flux de données.
flux de données	Système de programmation consistant en des nœuds exécutables, dans lequel les nœuds s'exécutent uniquement lorsqu'ils ont reçu toutes les entrées de données requises et produisent des sorties automatiquement lorsque leur exécution est terminée. Le G est un système à flux de données.
fonction	Elément d'exécution intégré, comparable à un opérateur, une fonction ou une instruction dans un langage conventionnel.
formats de stockage de données	Organisation et représentation des données stockées en mémoire.

G

G	Langage de programmation graphique utilisé dans les applications LabVIEW et BridgeVIEW.
galerie de solutions	Option de l'Assistant Solutions DAQ grâce à laquelle vous pouvez effectuer des choix parmi de nombreuses catégories d'applications DAQ courantes.
glissière	Partie mobile des commandes et indicateurs à glissière.
glyphe	Petite image ou icône.
GPIB	General Purpose Interface Bus (bus d'interface universel). Nom courant du système d'interface de communications défini dans la norme ANSI/IEEE 488.1-1987 et dans la norme ANSI/IEEE 488.2-1987. Hewlett-Packard, l'inventeur du bus, l'appelle HP-IB.

graphe à balayage — Indicateur numérique modélisé sur le fonctionnement d'un oscilloscope. Il est similaire à un oscillographe, à ceci près qu'une ligne se déplace sur l'affichage pour séparer les anciennes données des nouvelles.

graphe déroulant — Indicateur de tracé numérique modélisé sur un enregistreur de graphe déroulant à ruban de papier, qui se déroule à mesure qu'il trace les données. *Voir aussi* oscillographe, graphe à balayage et graphe oscilloscopique.

graphe oscilloscopique — Indicateur qui trace des points de données à une certaine vitesse.

H

handle — Pointeur vers un pointeur de bloc de mémoire ; les handles font référence à des tableaux et à des chaînes de caractères. Un tableau de chaînes de caractères est un handle sur un bloc de mémoire contenant des handles de chaînes de caractères.

hex — Hexadécimal. Un système de nombres en base 16.

Hz — Hertz. Cycles par seconde.

I

icône — Représentation graphique d'un nœud sur un diagramme.

IEEE — Institute for Electrical and Electronic Engineers (Institut pour ingénieurs électriciens et électroniciens).

impression contrôlée par programme — Impression automatique de la face-avant d'un VI après exécution.

indicateur — Objet de la face-avant qui affiche une sortie.

Inf — Valeur d'affichage numérique pour une représentation à virgule flottante de l'infini.

info-bulle — Petite zone de texte qui affiche le nom d'un objet, d'une commande ou d'un terminal.

instrument virtuel — VI. Programme dans le langage de programmation graphique G ; ainsi appelé parce qu'il reproduit l'apparence et la fonction d'un instrument physique.

J

jonction de câblage	Point de rencontre de trois (ou davantage) segments de câblage.

L

LabVIEW	Laboratory Virtual Instrument Engineering Workbench. Application de développement de programme basée sur le langage de programmation G utilisée couramment pour des mesures et des tests.
lecteur	Lettre comprise entre a et z, suivie par deux-points (:), indiquant un lecteur logique.
LED	Diode électroluminescente.
légende	Objet appartenant à un graphe ou à un graphe déroulant, qui affiche le nom et le style des tracés qui figurent sur ce graphe ou graphe déroulant.

M

marquise	Cadre mobile, en pointillés, qui entoure des objets sélectionnés.
matrice	Tableau à deux dimensions.
menu déroulant	Menu accessible à partir d'une barre de menus. Les options des menus déroulants sont souvent de nature générale.
menu local	Menu auquel on accède en cliquant sur un objet avec le bouton droit de la souris (Windows) ou en maintenant enfoncée la touche de commande (Macintosh). Les options de ce menu s'appliquent à cet objet.
message d'erreur	Indication d'une erreur logicielle ou matérielle, ou d'une tentative d'entrée de données inacceptable.
mise à l'échelle automatique	Aptitude des échelles à s'ajuster à la gamme de valeurs tracées. Sur les échelles des graphes, cette fonction détermine la valeur maximale et la valeur minimale de l'échelle.
mnémonique	Chaîne de caractères associée à une valeur entière.
Mo	Méga-octets de mémoire.

N

NaN	Valeur d'affichage numérique pour une représentation à virgule flottante de *not a number (pas un nombre.* Il s'agit généralement du résultat d'une opération non définie, telle que log(–1).
nœud	Elément d'exécution d'un programme. Les nœuds sont analogues aux déclarations, aux opérateurs, aux fonctions et aux sous-programmes dans les langages de programmation conventionnels. Dans un diagramme, les nœuds comportent des fonctions, des structures et des sous-VIs.
nœud Assembler	Fonction qui crée des clusters à partir de différents types d'éléments.

O

objet	Terme générique pour n'importe quel élément de la face-avant ou du diagramme, y compris les commandes, les nœuds, les fils de liaison et les images importées.
oscillographe	Indicateur numérique modélisé sur le fonctionnement d'un oscilloscope.
outil	Curseur spécial que vous pouvez utiliser pour effectuer des opérations spécifiques.
outil Bobine	Outil utilisé pour définir des chemins de données entre les terminaux source et destination.
outil Doigt	Outil utilisé pour entrer des données dans des commandes et pour les exécuter. Ressemble à un doigt tendu.
outil Flèche	Outil ressemblant à une flèche et utilisé pour déplacer, sélectionner et redimensionner des objets.
outil Main	Outil utilisé pour faire défiler les fenêtres.
outil Menu local	Outil utilisé pour accéder au menu local d'un objet.
outil Pinceau	Outil que vous utilisez pour définir les couleurs de premier plan et d'arrière-plan.
outil Pipette	Copie des couleurs afin de les coller avec l'outil Pinceau.

outil Point d’arrêt	Outil utilisé pour placer un point d’arrêt sur un VI, un nœud ou un fil de liaison.
outil Sonde	Outil utilisé pour placer des sondes sur des fils de liaison.
outil Texte	Outil utilisé pour créer des étiquettes et entrer du texte dans des fenêtres de texte.
ouvrir un menu local	Invoquer un menu spécial en cliquant sur un objet avec le bouton droit de la souris (Windows) ou en maintenant enfoncée la touche de commande (Macintosh).

P

palette	Affichage des icônes qui représentent les options possibles.
palette de Commandes	Palette contenant des commandes et indicateurs de face-avant.
palette de Fonctions	Palette contenant des structures de diagramme, des constantes, des fonctions de communication et des VIs.
palette d’outils	Palette contenant des outils que vous pouvez utiliser pour éditer et mettre au point les objets de la face-avant et du diagramme.
palette hiérarchique	Menu qui contient des palettes et des sous-palettes.
pas un chemin	Valeur prédéfinie pour la commande de chemin, qui signifie que le chemin n’est pas valide.
pas un refnum	Valeur prédéfinie qui signifie que le refnum n’est pas valide.
pixmap	Format standard pour stocker des images, dans lequel chaque pixel est représenté par une valeur de couleur. Une bitmap est une version noir et blanc d’une pixmap.
plate-forme	Ordinateur et son système d’exploitation.
point d’arrêt	Interruption de l’exécution lors de l’appel à un sous-VI. Vous pouvez placer un point d’arrêt en cliquant sur un VI, un nœud ou un fil de liaison avec l’outil Point d’arrêt qui se trouve dans la palette **Outils**.
point de coercition	Glyphe sur un terminal indiquant qu’un ou deux terminaux câblés ensemble ont été convertis pour correspondre à leur type de données respectif.

polymorphisme	Aptitude d'un nœud à s'ajuster automatiquement à des données de représentation, de type ou de structure différents.
programmation modulaire	Programmation qui utilise des programmes informatiques interchangeables.
programmation séquentielle	Système de programmation dans lequel l'ordre séquentiel des instructions détermine l'ordre d'exécution. La plupart des langages de programmation textuels conventionnels, tels que le C, le Pascal et le BASIC, sont des langages de programmation séquentielle.
pseudo-code	Représentation simplifiée, indépendante du langage, de code de programmation.

R

refnum	Identificateur d'une communication DDE ou d'un fichier en G pouvant être référencé par des VIs.
refnum de fichier	Le G associe cet identificateur à un fichier lorsque vous l'ouvrez. Vous devez utiliser le refnum de fichier pour indiquer que vous souhaitez qu'une fonction ou un VI réalise une opération sur le fichier ouvert.
registre à décalage	Mécanisme optionnel utilisé dans des structures de boucle pour transmettre la valeur d'une variable d'une itération de la boucle à l'itération suivante.
répertoire	Structure qui permet d'organiser des fichiers dans des groupes pratiques. Un répertoire est semblable à une adresse indiquant où les fichiers sont situés. Un répertoire peut contenir des fichiers ou des sous-répertoires de fichiers.
représentation	Sous-type du type de données numérique, qui comprend des entiers octets, mots et longs signés et non signés, ainsi que des nombres à virgule flottante en simple précision, en double précision et en précision étendue, à la fois réels et complexes.
routine externe partagée	Sous-programme qui peut être partagé par plusieurs ressources de code CIN.

S

scalaire	Nombre pouvant être représenté par un point sur une échelle. Une valeur unique, par opposition à un tableau. Les booléens et clusters scalaires sont des instances explicitement singulières de leurs types de données respectifs.
segment de câblage	Morceau de câblage horizontal ou vertical unique.
serveur OPC	OLE pour le contrôle de processus. Standard basé sur COM, défini par la fondation OPC, qui spécifie comment interagir avec des serveurs de périphériques. COM est une technologie 32 bits Windows.
sonde	Fonction de mise au point permettant de vérifier des valeurs intermédiaires dans un VI.
sous-diagramme	Diagramme à l'intérieur des limites d'une structure.
sous-palette	Palette contenue dans une icône d'une autre palette.
sous-VI	VI utilisé dans le diagramme d'un autre VI ; comparable à un sous-programme.
squelette de cluster	Objet de la face-avant qui contient les éléments d'un cluster.
structure	Elément de commande de programme, tel qu'une structure Séquence, une structure Condition, une boucle For ou une boucle While.
structure Condition	Structure de commande à branchement conditionnel, qui exécute un et un seul de ses sous-diagrammes, en fonction de son entrée. Elle est similaire aux instructions IF-THEN-ELSE et CASE des langages de programmation séquentielle.
structure Séquence	Structure de commande de programme qui exécute ses sous-diagrammes par ordre numérique. Couramment utilisée pour forcer des nœuds qui ne dépendent pas de données à s'exécuter dans un ordre désiré.

T

tableau	Liste ordonnée et indexée d'éléments de données du même type.
tableau vide	Tableau contenant zéro élément, mais qui a un type de données défini. Par exemple, un tableau qui a une commande numérique dans sa fenêtre d'affichage de données mais n'a de valeur définie pour aucun élément est un tableau numérique vide.
tableau vierge	Objet de la face-avant qui accueille un tableau. Il comporte un affichage de l'indice, une fenêtre d'objet de données et une étiquette optionnelle. Il peut accepter divers types de données.
temps réel	Utilisé pour décrire les performances d'un calcul pendant la durée réelle au cours de laquelle le processus physique correspondant se produit, de sorte que les résultats du calcul puissent être utilisés pour guider le processus physique.
terminal	Objet ou zone d'un nœud au travers duquel sont transmises des données.
terminal conditionnel	Terminal d'une boucle While contenant une valeur booléenne qui détermine si le VI effectue une autre itération.
terminal de comptage	Terminal d'une boucle For dont la valeur détermine le nombre de fois que la boucle For doit exécuter son sous-diagramme.
terminal de destination	Terminal qui reçoit des données.
terminal d'itération	Terminal d'une boucle For ou d'une boucle While qui contient le nombre d'itération actuellement effectuées.
terminal source	Terminal qui émet des données.
texte libre	Etiquette sur la face-avant ou le diagramme qui n'appartient à aucun autre objet.
tracé	Représentation graphique d'un tableau de données, affichée soit sur un graphe soit sur un graphe déroulant.
tunnel	Terminal d'entrée ou de sortie de données sur une structure.

type de données	Format d'information. Dans BridgeVIEW, les types de données acceptables pour la configuration de service sont les données analogiques, discrètes, de tableaux de bits et de chaînes de caractères. Dans LabVIEW, les types de données acceptables pour la plupart des fonctions sont les données numériques, de tableaux, de chaînes de caractères et de clusters.

U

UUT	Unit Under Test (Unité en cours de test).

V

variable globale	Sous-VI non ré-entrant à mémoire locale qui utilise un registre à décalage non initialisé pour stocker des données entre deux exécutions. La mémoire de copies de ces sous-VIs est partagée et peut ainsi être utilisée pour transférer entre elles des données globales.
variable locale	Variable qui vous permet de lire des commandes ou des indicateurs, ou de leur transmettre des données, sur la face-avant de votre VI.
variable locale de séquence	Terminal qui transmet des données entre les étapes d'une structure Séquence.
VI	*Voir* instrument virtuel.
VI à souche	Prototype non fonctionnel d'un sous-VI. Il a des entrées et des sorties, mais est incomplet. Il est utilisé lors des premiers stades de l'élaboration d'un VI pour réserver des emplacements pour le développement ultérieur du VI.
VI actuel	VI dont la face-avant, le diagramme ou l'Editeur d'icône est la fenêtre active.
VI invalide	VI qui ne peut pas être compilé ni exécuté ; désigné par une flèche brisée dans le bouton Exécution.
VI principal	VI situé en haut de la hiérarchie des VIs. Ce terme distingue le VI de ses sous-VIs.
VISA	*Voir* architecture logicielle d'instruments virtuels.
VXI	VME eXtensions for Instrumentation (extensions VME pour l'instrumentation) (bus).

Z

zone de liste — Boîte à l'intérieur d'une boîte de dialogue listant tous les choix disponibles pour une commande, comme par exemple une liste de noms de fichiers sur un disque. En général, vous sélectionnez un élément dans la zone de liste, puis vous cliquez sur **OK**. Si la zone de liste ne peut pas afficher tous les choix existants, elle dispose de barres de défilement. En sélectionnant la flèche vers le bas à côte du premier élément dans la liste, vous affichez le reste de la zone de liste.

zone de redimensionnement — Zone triangulaire située aux coins d'un objet et indiquant des points de redimensionnement.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z