ERSTE SCHRITTE MIT DEN LABVIEW™ VIS FÜR PUNKT-FÜR-PUNKT-ANALYSEN

Inhaltsverzeichnis

370433A-01 *Januar 2002*

LabVIEW bietet eine Reihe von VIs, die Punkt-für-Punkt - Analysen durchführen. Die Punkt-für-Punkt-Analyse eignet sich hervorragend für die Echtzeitdatenverarbeitung. Wenn Ihre Datenerfassung also eine deterministische Echtzeitdatenverarbeitung erfordert, können Sie nun ein Programm erstellen, das statt der Array-basierten LabVIEW-VIs die Punkt-für-Punkt-VIs zur Datenanalyse verwendet.

In der Messtechnik ist die Echtzeit-Datenerfassung eine Realität. Mit der Punkt-für-Punkt-Analyse können die Daten nun auch in Echtzeit analysiert und verarbeitet werden. Für schnelle deterministische Echtzeitsysteme können die diskreten Stufen der Array-basierten Analyse, wie zum Beispiel Puffervorbereitung, Analyse und Ausgabe, zu langsam sein. Hier können kontinuierliche Analysen erforderlich sein, in denen für jeden einzelnen Datenwert eine Punkt-für-Punkt-Analyse stattfindet.

Dieses Dokument erläutert die Konzepte und einige Programmierungsdetails der Punkt-für-Punkt-Datenanalyse. Die Benutzung der Punkt-für-Punkt-VIs in LabVIEW bietet folgende Vorteile:

- Erfassen von Echtzeitereignissen und die Reaktion darauf.
- Der Analysevorgang ist direkt mit dem eingehenden Signal verbunden und garantiert damit hohe Geschwindigkeit und minimalen Datenverlust.
- Aufgaben können viel einfacher programmiert werden, weil keine Arrays zugewiesen und die Abtastraten weniger oft angepasst werden müssen.
- Automatische Synchronisation von Analyse und Datenerfassung, weil direkt mit einem Einzelsignal gearbeitet wird.

Die Benutzung der Punkt-für-Punkt-VI-Bibliotheken von LabVIEW

Die Punkt-für-Punkt-VIs von LabVIEW entsprechen den für die kontinuierliche Datenerfassung relevanten Array-basierten Analyse-VIs. Programmierungsunterschiede müssen jedoch berücksichtigt werden. Normalerweise ist die Anzahl der zu programmierenden Aufgaben geringer, wenn Punkt-für-Punkt-VIs verwendet werden. Die folgende

Tabelle beschreibt die charakteristischen Eingangs- und Ausgangsparameter eines Punkt-für-Punkt-VIs in LabVIEW.

Tabelle 1. Charakteristische Eingangs- und Ausgangsparameter eines Punkt-für-Punkt-VIs

Parameter	Beschreibung
Eingabedaten	Eingehende Daten
Ausgabedaten	Ausgehende, analysierte Daten
Initialisieren	Routine die den internen Status eines VIs zurücksetzt.
Sample-Länge	Einstellung die einen signifikanten Teil aus dem Datensatz des Datenerfassungssystems oder des Analysesystems am besten darstellt.

Im Abschnitt *Fallstudie einer Punkt-für-Punkt-Analyse* finden Sie ein Beispiel für ein Analysesystem mit Punkt-für-Punkt-VIs.

Initialisieren der Punkt-für-Punkt-VIs

Dieses Kapitel beschreibt, wann und wie der Parameter **Initialisieren** in vielen Punkt-für-Punkt-VIs verwendet wird. Dieses Kapitel beschreibt auch die Funktion Erster Aufruf?, die seit LabVIEW 6.0 verfügbar ist.

Zweck der Initialisierung in Punkt-für-Punkt-VIs

Durch den Parameter **Initialisieren** kann der interne Status von VIs ohne Unterbrechung des Datenflusses oder der Berechnung zurückgesetzt werden. Ein VI kann als Antwort auf folgende Ereignisse zurückgesetzt werden:

- Der Anwender ändert die Werte eines Parameters.
- Die Anwendung erzeugt ein bestimmtes Ereignis oder erreicht einen Schwellwert.

Beispiel: Das VI Wertänderung (Punkt-für-Punkt) [Value has Changed PtByPt VI] aus der Palette **Funktionen»Analyse»Punkt-für-Punkt» Weitere Funktionen (Punkt-für-Punkt)** kann wie folgt auf Ereignisänderungen reagieren:

- Eingabedaten empfangen.
- Änderung erkennen.
- Den Booleschen Wert TRUE erzeugen, der eine Initialisierung in einem anderen VI auslöst.
- Eingabedaten zur Verarbeitung in ein anderes VI übertragen.

Abbildung 1 zeigt das VI Wertänderung (Punkt-für-Punkt) [Value has Changed PtByPt VI], wie es die Initialisierung eines anderen VI auslöst und die Daten dorthin überträgt. In diesem Fall sind die Eingabedaten ein Parameterwert für das Ziel-VI.

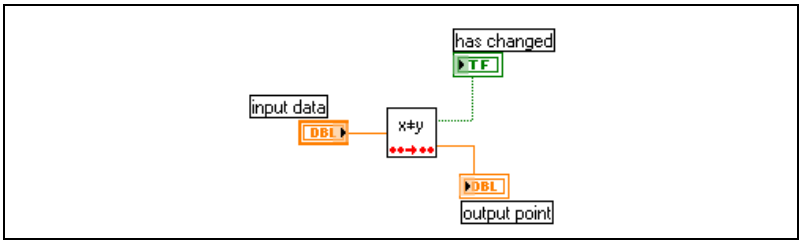


Abbildung 1. Typische Aufgaben des VIs Wertänderung (Punkt-für-Punkt) [Value Has Changed PtByPt VI]

Verwenden der Funktion Erster Aufruf?

In vielen Punkt-für-Punkt-Anwendungen ist es nicht erforderlich den Parameter **Initialisieren** zu verwenden, da die Initialisierung immer dann automatisch stattfindet, wenn der Benutzer die Anwendung beendet und neu startet. Es gibt zum Beispiel Anwendungen, die sich alle 24 Stunden automatisch abschalten. Das Abschalten bewirkt dabei eine neue Initialisierung des Systems.

Falls erforderlich, enthalten die Punkt-für-Punkt-VIs die Funktion Erster Aufruf?. Enthält ein VI diese Funktion, wird der interne Status einmal zurückgesetzt, sobald das VI das erste Mal aufgerufen wird. Der Wert des Parameters **Initialisieren** in der Funktion Erster Aufruf? ist für den ersten Aufruf TRUE. Für die restliche Laufzeit des VIs ist der Wert des Parameters FALSE. Sie sollten die Funktion Erster Aufruf? aus der Palette **Funktionen»Fortgeschritten»Synchronisation** benutzen, wenn Sie VIs für die Punkt-für-Punkt-Analyse programmieren wollen. Abbildung 2 zeigt die typische Anwendung der Funktion Erster Aufruf? in einer WHILE-Schleife

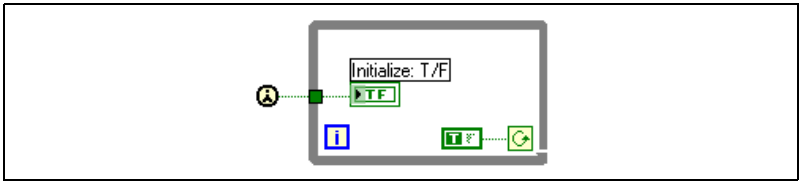


Abbildung 2. Verwenden der Funktion Erster Aufruf? in einer While-Schleife

Fehlerprüfung und Initialisierung

Punkt-für-Punkt-VIs erzeugen Fehlermeldungen, die Ihnen helfen, fehlerhafte Stellen in der Konfiguration der von ihnen erstellten Anwendung zu finden. Zusätzlich zu den Standard-LabVIEW-Fehlercodes gibt es hierfür noch weitere spezielle Punkt-für-Punkt-Fehlercodes.

Normalerweise identifizieren Fehlercodes ungültige Parameter und Einstellungen. Für eine genauere Fehlerprüfung sollten Sie Ihr Programm so konfigurieren, dass es die Datenerfassung und Berechnung überwacht und bei Unregelmäßigkeiten Meldungen ausgibt. So können Sie zum Beispiel eine weitere Form der Fehlerüberwachung hinzufügen, indem Sie eine Bereichsprüfung Ihrer Daten vornehmen.

Ein Punkt-für-Punkt-VI erzeugt nur ein einziges Mal einen Fehlercode, und zwar entweder nach dem ersten Aufruf des VIs oder beim ersten Aufruf nach der Initialisierung der Applikation. Da Punkt-für-Punkt-VIs nur einmal Fehlercodes erzeugen, können sie hervorragend in deterministischen Echtzeitanwendungen verwendet werden.

Der Fehlercode, den die Punkt-für-Punkt-VIs bei der Erkennung eines Fehlers nach dem ersten Aufruf des VI erzeugen, dient dazu, den Anwender über ungültige Parameter oder fehlerhafte Einstellungen zu informieren. In nachfolgenden Aufrufen setzen Punkt-für-Punkt-VIs den Fehlercode auf Null und laufen ohne weitere Fehlercodes zu erzeugen weiter. Die Anwendung kann so programmiert werden, dass sie eine der folgenden Funktionen als Reaktion auf den ersten Fehler ausführt:

- Fehler melden und Ausführung fortsetzen.
- Fehler melden und anhalten.
- Fehler ignorieren und Ausführung fortsetzen. (Voreinstellung)

Folgende Programmstruktur beschreibt wie das VI Wertänderung (Punkt-für-Punkt) [Value has Changed PtByPt VI] genutzt werden kann, um Mechanismen zur Fehlerprüfung bei Punkt-für-Punkt-VIs zu erstellen, die einen **Fehler**-Parameter enthalten.

1. Parameter auswählen, der auf Fehler überwacht werden soll.
2. Verbinden Sie den Parameterwert als **Eingangsdaten** mit dem VI Wertänderung (Punkt-für-Punkt) [Value has Changed PtByPt VI].
3. Leiten Sie die **Ausgangsdaten**, die nichts anderes als die unveränderten **Eingangsdaten** des VIs Wertänderung (Punkt-für-Punkt) [Value has Changed PtByPt VI] darstellen, zum Ziel-VI.
4. Das VI Wertänderung (Punkt-für-Punkt) [Value has Changed PtByPt VI] gibt jedes Mal den Wert TRUE aus, wenn sich der Eingangsparameterwert ändert. Nun kann das TRUE-Ereignis an das

Ziel-VI weitergeleitet werden, um dort eine Initialisierung auszulösen (siehe Abbildung 1).

5. LabVIEW prüft nach dem ersten Aufruf, der dieser Initialisierung folgt, auf aufgetretene Fehler. Diese Initialisierungs- und Fehlerprüfschleife läuft nach jeder Veränderung des Eingangsparameters ab.

Häufig gestellte Fragen (FAQs)

Dieses Kapitel beantwortet häufig gestellte Fragen zur Punkt-für-Punkt-Analyse.

Was sind die Unterschiede zwischen Punkt-für-Punkt-Analyse und Array-basierter Analyse in LabVIEW?

Tabellen 2 und 3 betrachten die Array-basierte Analyse mit der Punkt-für-Punkt-Analyse aus unterschiedlichen Perspektiven. Tabelle 2 zeigt den Unterschied zwischen der Array-basierten Datenanalyse und der Punkt-für-Punkt-Analyse bei zwei Fahrzeugkraftstoffversorgungsanlagen mit Gemischbildung und Kraftstoffeinspritzung

Tablelle 2. Vergleich von herkömmlichen mit neuen Methoden

Herkömmliche Methode	Neue Methode
Fahrzeugtechnik	
Gemischbildung • Kraftstoff sammelt sich in einer Wirbelkammer. • Das Vakuum in den Brennkammern saugt den Kraftstoff durch einen Satz von Dosierungsventilen die alle Brennkammern versorgen. • Eine effiziente Verbrennung findet statt.	**Kraftstoffeinspritzung** • Kraftstoff fließt kontinuierlich aus dem Kraftstofftank. • Kraftstoff wird im Augenblick der Verbrennung in die Brennkammer gespritzt. • Dadurch findet eine optimale Verbrennung statt.
Datenanalyse-Technologie	
Array-basierte Analyse • Datenpuffer vorbereiten. • Daten analysieren. • Puffer für analysierte Daten erstellen. • Report erzeugen.	**Punkt-für-Punkt-Analyse** • Empfangen eines kontinuierlichen Datenstromes. • Filtern und analysieren der Daten. • Erzeugen von Echtzeit-Ereignissen und -Protokollen.

Tabelle 3 zeigt weitere Vergleiche zwischen Array-basierten und Punkt-für-Punkt-Analysen.

Tablelle 3. Vergleich von Array-basierter mit Punkt-für-Punkt-Analyse

	Array-basierte Analyse	**Datenerfassung & Analyse mit Punkt-für-Punkt-VIs**
Kompatibilität	Begrenzte Kompatibilität mit Echtzeitsystemen	Kompatibel mit Echtzeitsystemen und Abwärtskompatibel zu Array-basierten Systemen
Dateneingabe	Array-orientiert	Skalar-orientiert
Unterbrechungen	Unterbrechungen sind kritisch	Unterbrechungen sind zulässig
Betrieb	Sie beobachten, Offline	Sie steuern, Online
Leistung und Programmierung	Ausgleichen von anfänglichen Datenverlusten (4–5 Sekunden) bei der Verwendung von komplexen “Statusmaschinen”.	Anfängliche Datenverluste treten nicht auf; das Datenerfassungssystem wird einmal initialisiert und läuft danach kontinuierlich.
Perspektive	Abbildung eines Prozesses	Direkt, natürlicher Ablauf eines Prozesses
Programmierung	Einen Puffer festlegen	Keine bestimmten Puffer
Ergebnisse	Einen Report ausgeben	Einen Report und ein Ereignis in Echtzeit ausgeben
Run-time-Verhalten	Verzögerte Verarbeitung	Echtzeit
Run-time-Verhalten	Stopp	Fortsetzen
Run-time-Verhalten	Warten	Sofort
Arbeitsweise	Asynchron	Synchron

Warum Punkt-für-Punkt Analysen benutzen?

Punkt-für-Punkt-Analysen arbeiten sehr gut mit computergestützter Echtzeitdatenerfassung. Der Eingabe-Analyse-Ausgabe-Prozess in Array-basierten Analysen findet in Teilmengen von größeren Datensätzen statt. Der Eingabe-Analyse-Ausgabe-Prozess in Punkt-für-Punkt-Analysen findet durchgehend in Echtzeit statt.

Was ist neu bei der Punkt-für-Punkt-Analyse?

Beim Ausführen von Punkt-für-Punkt-Analysen sollten folgende Konzepte bedacht werden:

- Initialisierung—Die Anwendung zur Punkt-für-Punkt-Analyse muss zur Vermeidung von Störungen durch Einstellungen aus vorherigen Sitzungen initialisiert werden.
- Ablaufinvariante Ausführung—Für Punkt-für-Punkt-Analysen muss die ablaufinvariante Ausführung aktiviert sein. Ablaufinvariante Ausführung weist einem einzelnen Analysevorgang festgelegten Speicher zu. Das stellt sicher, dass keine Störungen auftreten, wenn zwei Prozesse dieselbe Analysefunktion nutzen.

Hinweis Wenn Sie also eigene VIs in Ihrer Punkt-für-Punkt-Applikation verwenden, sollten Sie sicherstellen, dass die ablaufinvariante Ausführung aktiviert ist. Die ablaufinvariante Ausführung ist in fast allen Punkt-für-Punkt-VIs standardmäßig aktiviert.

- Deterministische Performance—Punkt-für-Punkt-Analyse ist ein natürlicher Bestandteil vieler deterministischer Systeme, weil sie sich wirkungsvoll mit dem Fluss von Echtzeitsignalen kombinieren lässt.

Welche Gemeinsamkeiten haben Punkt-für-Punkt-VIs und Array-basierte VIs?

Die VIs zur Punkt-für-Punkt-Analyse werden den meisten Anwendern sehr vertraut erscheinen, da der Ansatz der meisten Analyseoperationen der gleiche ist, wie der bei den Array-basierten VIs. So können Filter-, Integrations-, Mittelwert- und alle anderen Algorithmen in der gleichen Weise eingesetzt werden, wie sie auch in der Array-basierten Datenanalyse eingesetzt werden. Im Gegensatz dazu ist die Nullstellenberechnung in polynomischen Funktionen bei der Punkt-für-Punkt-Analyse irrelevant und Punkt-für-Punkt-Versionen dieser Array-basierten VIs sind daher nicht notwendig.

Wie ist es möglich Analysen ohne Datenpuffer durchzuführen?

Aus Analysefunktionen ergeben sich Lösungen die das Verhalten von Datensätzen kennzeichnen. Große Datensätze können in der Array-basierten Datenerfassung und -analyse zum Beispiel in zehn kleine Puffer aufgeteilt werden. Die Analyse der zehn Datensätze ergibt zehn Lösungen. Diese zehn Lösungen können dann zu einer Lösung zusammengefasst werden, die das Verhalten des gesamten Datensatzes kennzeichnet.

In der Punkt-für-Punkt-Analyse wird ein kompletter Datensatz in Echtzeit analysiert. Ein Datum bestimmter Länge ersetzt dabei den Puffer. Das Datum kann die Länge eines signifikanten Ereignisses aus dem zu analysierenden Datensatz haben. Weitere Informationen und ein Beispiel, das mehrere tausend Datenwerte pro Sekunde zur Erkennung fehlerhafter Räder aufnimmt, finden Sie im Abschnitt *Fallstudie einer Punkt-für-Punkt-Analyse*. In diesem Beispiel kommt das Signal von einem Zug, der sich mit einer Geschwindigkeit von 60 bis 70 km pro Stunde fortbewegt. Die Länge einer Dateneinheit dieser Anwendung entspricht dem kleinsten Abstand zwischen den Rädern.

Eine Applikation für eine Punkt-für-Punkt-Analyse analysiert typischerweise eine lange Serie von Datensätzen, von denen vielleicht nur einige von Interesse sind. Um diese interessanten Datenbereiche zu erkennen, analysiert die Applikation Übergänge, wie das **Ende des relevanten Signals**.

Im Kapitel *Fallstudie einer Punkt-für-Punkt-Analyse* nutzt eine Anwendung, die Räder von Zügen abtastet, das **Signalende** um die entsprechenden, kritischen Datenbereiche zu erkennen. Sobald ein Übergang erkannt wird, ermittelt die Applikation die maximale Amplitude aus dem aktuellen Datensatz. Diese gerade ermittelte Amplitude entspricht dem Gesamtsignal des Zugrads, dessen Signalende gerade erreicht wurde. Das Auslesen von Amplituden in Echtzeit kann zur Erzeugung eines Ereignisses oder zur Erstellung eines Berichts über das Rad und den Zug benutzt werden.

Warum ist die Punkt-für-Punkt-Analyse in Echtzeitanwendungen so effektiv?

Punkt-für-Punkt-Analyse wird hauptsächlich zur Verarbeitung von kontinuierlichen, schnellen Datenflüssen eingesetzt. Beispiel: In der industriellen Automatisierung fließen kontinuierliche Datenströme und Computer nutzen eine Reihe von Analyse- und Übertragungsfunktionen um die realen Vorgänge zu steuern. Punkt-für-Punkt-Analysen können diese technischen Aufgaben in Echtzeit übernehmen.

Es gibt Echtzeitanwendungen, für die eine Hochgeschwindigkeitsdatenerfassung und -analyse nicht notwendig ist. Stattdessen werden eher einfache und zuverlässige Programme benötigt. Die Punkt-für-Punkt-Analyse bietet diese Einfachheit und Zuverlässigkeit, weil Arrays nicht ausdrücklich zugewiesen werden müssen und die zu analysierenden Daten unverfälscht und kontinuierlich fließen.

Benötige ich eine Punkt-für-Punkt-Analyse?

Solange in den zu steuernden Prozessen keine deterministischen Hochgeschwindigkeitsdatenerfassungen vorgesehen sind, kann eine Applikation auch ohne Punkt-für-Punkt-Analyse auskommen. Wenn jedoch Ressourcen in Echtzeit-Datenerfassungsanwendungen reserviert werden sollen, sollte die Punkt-für-Punkt-Analyse herangezogen werden, um die maximale Leistungsfähigkeit der Applikation zu erhalten. Wird die Rate der Abtastungen pro Sekunde um den Faktor 10 erhöht, steigt auch die Notwendigkeit der Verwendung von Punkt-für-Punkt-Analysen an.

Die Punkt-für-Punkt-Analyse vereinfacht den Design-, Implementierungs- und Prüfungsprozess, weil der Programmablauf der Anwendung mit dem Fluss des realen Prozesses, der überwacht und gesteuert werden soll, übereinstimmt.

Was ist die langfristige Bedeutung von Punkt-für-Punkt-Analysen?

Echtzeit-Datenerfassungen und -Analysen erfordern auch in Zukunft rationellere und stabilere Anwendungen. Die Punkt-für-Punkt-Analyse ist rationell und stabil, weil sie direkt in den Erfassungs- und Analyseprozess eingebunden ist. Rationelle und stabile Punkt-für-Punkt-Analysen ermöglichen es dem Erfassungs- und Analyseprozess näher an die Möglichkeiten von FPGA (field programmable gate array)-Chips, DSP-Chips, eingebetteten Controllern und zweckbestimmten CPUs und ASICs zu rücken.

Beispiele

Echtzeit-Analysen und -Steuerungen sind nur einige der Vorteile der Punkt-für-Punkt-Methode. Folgendes Beispiel zeigt die Vorteile bei der Benutzung von Punkt-für-Punkt-VIs und -Funktionen.

Beispiel für Punkt-für-Punkt- und Array-basierte Filter

Das VI “Punkt-für-Punkt und Array-basierter Filter” [The PtByPt Array Based Filter VI] in Abbildung 3 erzeugt ein Signal, das ein unerwünschtes Rauschen enthält. Das VI filtert das Signal, um das Rauschen zu entfernen und eine bessere Darstellung des eigentlichen Signals zu erhalten. Der Filter blockiert unerwünschte Frequenzen im Signal und lässt wichtige Frequenzen passieren. Das VI benutzt zwei Methoden, um unerwünschtes Rauschen aus dem Signal herauszufiltern. Methode eins entfernt unerwünschtes Rauschen durch den Einsatz einer Punkt-für-Punkt-Version des Filters. Methode zwei benutzt die Array-basierte Version des Filters.

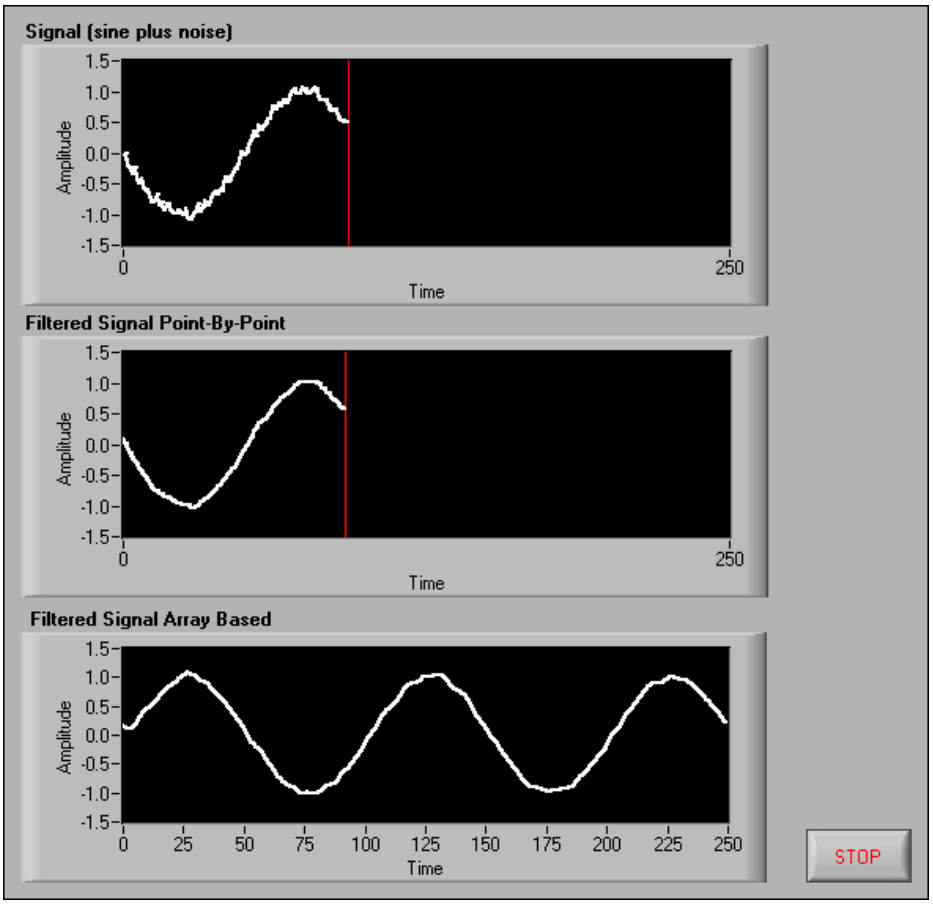


Abbildung 3. Punkt-für-Punkt im Gegensatz zu Array-basierten Filtern

Punkt-für-Punkt-basierter Filter

In der Punkt-für-Punkt-Programmierung von LabVIEW erfasst ein VI einen einzelnen Datenpunkt, analysiert diesen und stellt ihn für einen Report zur Verfügung. Zur gleichen Zeit erfasst das VI einen weitern Datenpunkt und wiederholt den Vorgang Punkt-für-Punkt. Im Punkt-für-Punkt-Filter analysiert das VI jeden Punkt des eingehenden Signals und gibt einen nach dem anderen in Echtzeit aus.

Array-basierter Filter

In der Array-basierten LabVIEW-Programmierung erfasst ein VI einen Datensatz, analysiert diesen und erstellt einen Report. Mit dem Array-basierten Filter erfasst und analysiert das VI die Länge eines Signals. Nach der Analyse entfernt der Filter die unerwünschten Frequenzen aus

dem Datensatz und das VI zeigt einen Bericht des überarbeiteten, gefilterten Datensatzes im Graphen an. Der Graph zeigt das vollständige, gefilterte Signal an. Danach wartet das VI bis ein weiteres Signal erfasst ist und wiederholt den Vorgang.

Beispiel eines Echtzeit-Amplituden-Spektrums

Das VI Echtzeit-Amplituden-Spektrum [The Realtime Amplitude Spectrum VI] in Abbildung 4 erzeugt ein Signal und fügt ein Rauschen hinzu. Das VI analysiert das Signal und zeigt das Leistungsspektrum in einem Graph an.

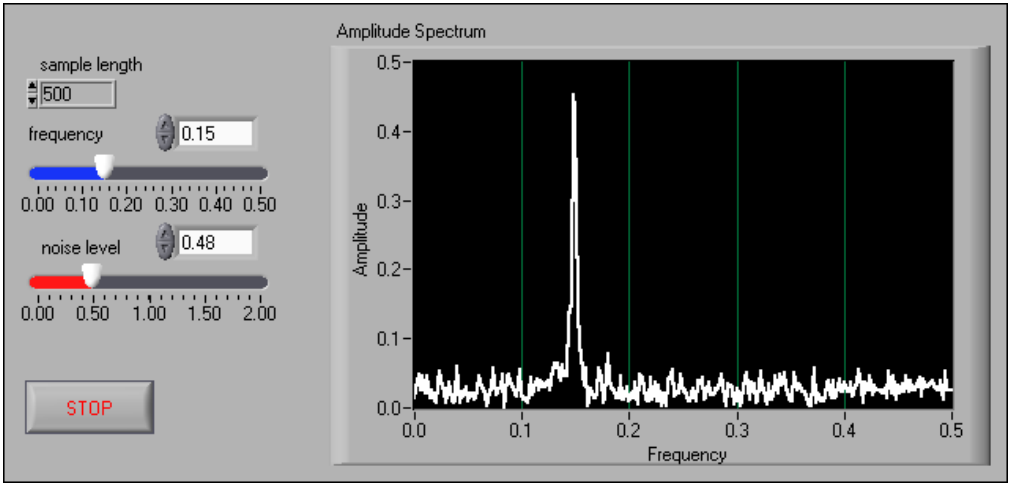


Abbildung 4. Leistungsspektrum-Analyse mit Echtzeitsteuerung

Dieses VI nutzt die Punkt-für-Punkt-VIs so, dass Frequenz, Rauschpegel und Signallänge in Echtzeit geändert werden können. Diese Änderungen beeinflussen das Leistungsspektrum in Echtzeit. Die Punkt-für-Punkt-Methode ist bei der Datenerfassung und Analyse sehr hilfreich, wenn Detailaufgaben im Echtzeitbereich ausgeführt werden müssen.

Fließendes Histogramm (Punkt-für-Punkt)

Das VI Fließendes Histogramm (Punkt-für-Punkt) [Moving Histogram PtByPt.vi] in Abbildung 5 erzeugt ein Signal, analysiert es und erstellt ein Histogramm. Intervalle und Signallänge können in Echtzeit eingestellt werden.

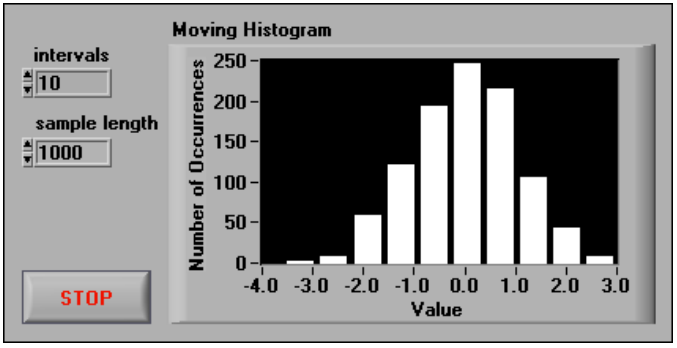


Abbildung 5. Fließendes Histogramm erzeugt aus Punkt-für-Punkt-VIs

Ein fließendes Histogramm kann auch mit den Array-basierten Funktionen von LabVIEW erstellt werden, was allerdings sehr aufwendig ist. Die Punkt-für-Punkt-Methode erlaubt die einfache Erstellung solcher VIs mit Echtzeitverhalten.

Fallstudie einer Punkt-für-Punkt-Analyse

Die Fallstudie in diesem Kapitels zeigt eine vollständige LabVIEW-Anwendung einer Punkt-für-Punkt-Analyse. Die Anwendung mit Echtzeitdatenerfassung, die fehlerhafte Räder von Zügen erkennt, demonstriert die Einfachheit und Anpassungsfähigkeit der Punkt-für-Punkt-Datenanalyse. Das VI Zugrad [The Train Wheel PtByPt VI] verwendet die Punkt-für-Punkt-VIs.

Punkt-für-Punkt-Analyse von Zugrädern

In diesem Beispiel muss das Wartungspersonal fehlerhafte Räder eines Zuges erkennen. Ein Bahnarbeiter kann mit einem Hammer auf ein fehlerhaftes Rad schlagen und an einer veränderten Resonanz einen Schaden erkennen. Eine automatisierte Überwachung soll das manuelle Testen ersetzen, weil es zeitaufwendig, fehleranfällig und zu ungenau ist, um kleinste Fehler zu finden. Außerdem fügt eine automatisierte Lösung dem Testvorgang die Leistung des dynamischen Testens hinzu, weil die Räder des Zuges während des Tests in Betrieb sein können (und somit auch der gesamte Zug) und nicht still stehen müssen.

Eine Lösung zur Erkennung von fehlerhaften Rädern eines Zuges muss auch kleinste Anzeichen eines Schadens schnell und genau erkennen. Die Anwendung sammelt also Daten während einer normalen Fahrt. Die Daten werden jedoch nicht nur gesammelt und analysiert, sondern der Vorgang findet zudem in Echtzeit statt, um die Programmierung zu vereinfachen und um die Schnelligkeit und Genauigkeit der Ergebnisse zu erhöhen.

Das VI Zugrad [The Train Wheel PtByPt VI] nutzt die Punkt-für-Punkt-Analyse von LabVIEW um eine Lösung für das Erkennen von fehlerhaften Rädern bei Zügen zu entwickeln.

Abbildung 6 und 7 zeigen das entsprechende Frontpanel und das Blockdiagramm für das VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]. Das VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI] und andere Punkt-für-Punkt-Beispiel-VIs aus diesem Handbuch können aus der National Instruments Developer Zone unter `ni.com/zone` heruntergeladen werden.

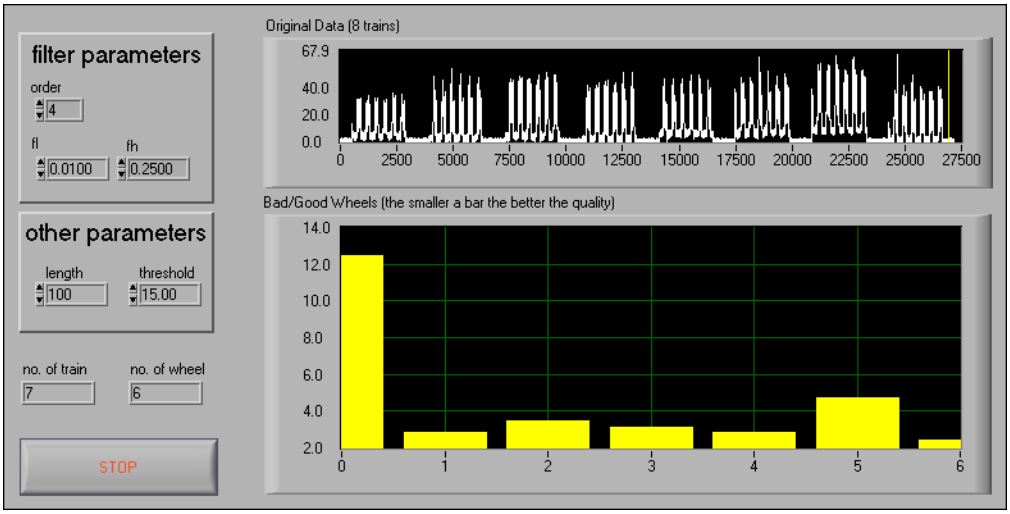


Abbildung 6. Frontpanel des VIs Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

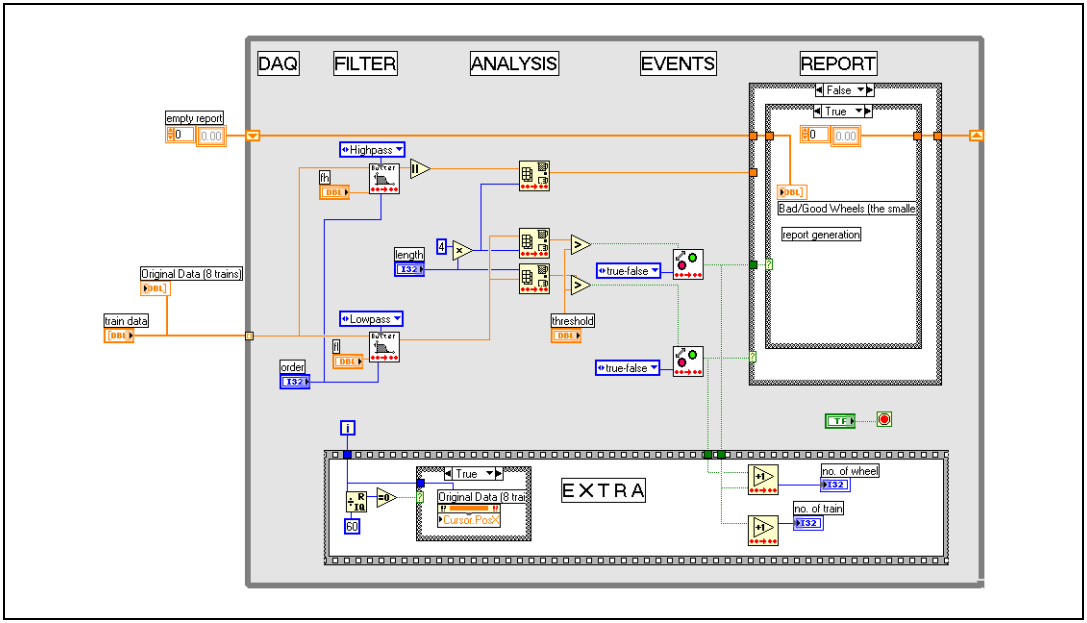


Abbildung 7. VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Hinweis Dieses Beispiel zeigt die Implementierung eines Punkt-für-Punkt-Analyseprogramms in LabVIEW. Ausführungen darüber, wie korrekte Abtastperioden ermittelt und die richtige Signalkonditionierung angewendet wird, gehen über den Rahmen dieser Einführung hinaus und sind daher auch nicht Bestandteil dieses Beispiels.

Übersicht über die Punkt-für-Punkt-Lösung mit LabVIEW

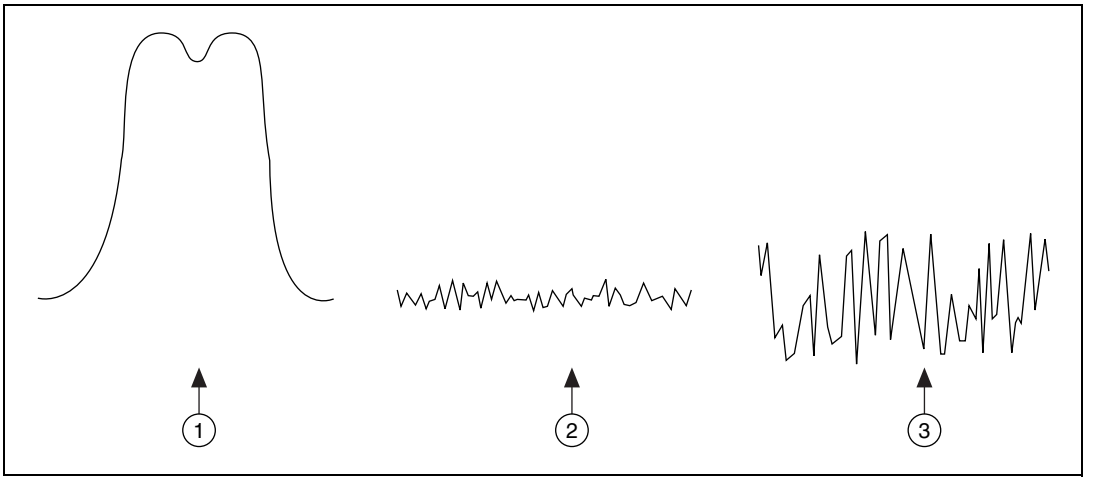
Die von der Anwendung aufgezeichneten Daten fließen kontinuierlich durch eine While-Schleife. Folgender Prozess beschreibt, was in der Schleife geschieht.

1. DAQ—Fluss von Signalverlaufs-Daten
2. Filter—Trennung von Nieder- und Hochfrequenzanteilen
3. Analyse—Erkennung von Zug, Rad und Energiepegel des Signalverlaufs für jedes Rad.
4. Ereignisse—Antworten auf Signalübergänge.
5. Bericht—Protokollierung von Zügen, Rädern und Zügen die fehlerhafte Räder haben könnten.

Die Anwendung verwendet Standardobjekte der LabVIEW-Programmierung, wie Case-Strukturen, While-Schleifen, numerische Bedienelemente und numerische Operatoren. Das VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI] benötigt zusätzlich die im folgenden Kapitel aufgezählten VIs.

Eigenschaften eines Signalverlaufs eines Zugrads

Der charakteristische Signalverlauf den ein Rad des Zuges aussendet bestimmt, wie der Signalverlauf Punkt-für-Punkt analysiert und gefiltert werden muss. Ein sich bewegendes Zugrad ergibt ein Signal, das Nieder- und Hochfrequenzanteile enthält. Wenn ein Dehnungsmessstreifen an einer Bahnschiene befestigt wird, wird ein verrauschtes Signal ähnlich einer Glockenkurve gemessen. Abbildung 8 zeigt die Nieder- und Hochfrequenzanteile der Kurve.



1 Tiefpass-Komponente eines typischen Zugradsignals (Zwillingsrad)
2 Hochpass-Komponente eines typischen Zugradsignals
3 Hochpass-Komponente eines defekten Zugrads (höhere Amplitude)

Abbildung 8. Nieder- und Hochfrequenzanteile eines Zugradsignals

Die Niederfrequenzanteile der Bewegung des Zugrades stellen das Rauschen im normalen Betrieb dar. Heile und fehlerhafte Räder erzeugen die gleichen Niederfrequenzanteile im Signal. Der Spitzenwert der Kurve zeigt den Augenblick an, in dem sich das Rad direkt über dem Dehnungsmessstreifen bewegt. Die niedrigsten Punkte der glockenförmigen Kurve zeigen den Anfang und das Ende des Rades, wenn das Rad den Dehnungsmessstreifen passiert.

Das Signal des Zugrades enthält einen Hochfrequenzanteil der den Zustand des Rades wiedergibt. Ein fehlerhaftes Rad erzeugt im Betrieb mehr Energie als ein unbeschädigtes Rad. Das bedeutet, dass der Hochfrequenzanteil eines fehlerhaften Rades eine höhere Amplitude hat.

Parameter von Signalverläufen im VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Die Signalverläufe aller Zugräder, einschließlich der fehlerhaften, fallen in einen vorhersehbaren Bereich. Das vorhersehbare Verhalten ermöglicht die Auswahl der Parameter aus Tabelle 4 Diese Parameter beziehen sich auf die fünf Vorgänge aus dem Kapitel *Übersicht über die Punkt-für-Punkt-Lösung mit LabVIEW*.

Hinweis Die Parameter in der Tabelle helfen die Details der Programmierung mit der Punkt-für-Punkt-Methode zu verstehen. Die Punkt-für-Punkt-Erfassung und -Analyse kann aber auch ohne nähere Betrachtung der Parameter verstanden werden. Die Parameter für Implementierungen des VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI] müssen für andere Applikationen neu eingestellt werden, da sich die Eigenschaften jedes Datenerfassungssystems unterscheiden.

Tablelle 4. Zulässige Parameter für Signalverläufe von Zugrädern im VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Parameter	Name	Vorgang und betroffenes VI	Zweck im VI Zugrad (Punkt-für-Punkt)
Eingangsdaten	**Eingabedaten**	Datenerfassung - VI DAQ (Eingang) [DAQ In VI]	Datenquelle, Gerät, Kanal, Abtastungen pro Sekunde und Skalierung.
Filterauflösung	**Ordnung**	Filterung - VI Butterworth-Filter (Punkt-für-Punkt) [Butterworth Filter PtByPt]	Menge der Signalverlaufsdaten, die das VI zu einer vorgegebenen Zeit filtert. `2` ist für Zugrad (Punkt-für-Punkt) zulässig. **Ordnung** gilt für beide Butterworth-Filter im VI Zugrad (Punkt-für-Punkt).
Untere Grenzfrequenz	**f1**	Filterung - VI Butterworth-Filter (Punkt-für-Punkt) [Butterworth Filter PtByPt]	Minimale Signalstärke die identifiziert werden kann, wenn das Zugrad den Dehnungsmessstreifen verlässt. `0,01` ist für Zugrad (Punkt-für-Punkt) zulässig.

Tablelle 4. Zulässige Parameter für Signalverläufe von Zugrädern im VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI] (Fortsetzung)

Parameter	Name	Vorgang und betroffenes VI	Zweck im VI Zugrad (Punkt-für-Punkt)
Obere Grenzfrequenz	**fh**	Filterung - VI Butterworth-Filter (Punkt-für-Punkt) [Butterworth Filter PtByPt]	Minimale Signalstärke, die das Ende der hochfrequenten Signalverlaufsdaten identifiziert. `0,25` ist für Zugrad (Punkt-für-Punkt) zulässig.
Länge des zu analysierenden Signalverlaufs	**Sample-Länge**	Analyse-VI Max & Min von Array (Punkt-für-Punkt) [Array Max & Min PtByPt]	Größe des Bereichs eines Signalverlaufs der vom VI Zugrad (Punkt-für-Punkt) analysiert wird. Um die optimale Abtastdauer zu ermitteln, müssen die Zuggeschwindigkeit, der kleinste Abstand zwischen den Rädern und die Anzahl der Abtastungen pro Sekunde berücksichtigt werden. `100` ist für Zugrad (Punkt-für-Punkt) zulässig. Das VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI] benutzt **Sample-Länge**, um die Werte für alle drei VIs Max & Min von Array (Punkt-für-Punkt) [Array Max & Min PtByPt VIs] zu berechnen.
Initialisierung	**Initialisieren**	Alle Punkt-für-Punkt-VIs	Routine, die ein VI für eine neue kontinuierliche Datenerfassung zurücksetzt.

Tablelle 4. Zulässige Parameter für Signalverläufe von Zugrädern im VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI] (Fortsetzung)

Parameter	Name	Vorgang und betroffenes VI	Zweck im VI Zugrad (Punkt-für-Punkt)
(Faktor)	(keiner)	Analyse-VI Max & Min von Array (Punkt-für-Punkt) [Array Max & Min PtByPt]	Bestimmt einen längeren Teil eines Signalverlaufs zur Analyse. Zeigt dieser längere Teil keine Signalaktivität für die Räder des Zuges an, identifiziert das VI das Zugende. 4 ist für Zugrad (Punkt-für-Punkt) zulässig.
(Schwellwert)	(keiner)	Analyse-VI Max & Min von Array (Punkt-für-Punkt) [Array Max & Min PtByPt]	Liefert einen Vergleichspunkt zur Identifikation des Falls, dass kein Signal eines Zugrades im gerade erfassten Signal vorhanden ist. 3 ist für Zugrad (Punkt-für-Punkt) zulässig.

Abtastrate im VI Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Die Punkt-für-Punkt-Anwendung arbeitet mit einem kontinuierlichen Strom aus Signalverlaufsdaten, die von den Rädern des sich bewegenden Zuges kommen. Für einen Zug, der 60–70 km/h fährt, können einige hundert bis einige tausend Abtastungen pro Sekunde notwendig sein, um ausreichende Informationen zur Erkennung von fehlerhaften Rädern zu erhalten.

Filteranforderungen des VIs Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Die Applikation zur Punkt-für-Punkt-Analyse muss Nieder- und Hochfrequenzanteile aus dem Signalverlauf des Zugrads herausfiltern. Zwei Butterworth-Filter führen daher folgende Aufgaben durch:

- Niederfreqzenzanteile des Signalverlaufs extrahieren.
- Hochfreqzenzanteile des Signalverlaufs extrahieren.

Analysekomponenten des VIs Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Die Anwendung muss die Nieder- und Hochfrequenzanteile getrennt analysieren. Für diesen Fall extrahiert eine Punkt-für-Punkt-Version des VI Max & Min von Array [Array Max & Min VI] die gewonnenen

Energiewerte in den Signalverläufen für jedes Rad, das Zugende und das Radende.

Drei separate VIs Max & Min von Array (Punkt-für-Punkt) führen die folgende Aufgaben durch.

- Identifikation der maximalen Signalfrequenz für jedes Rad.
- Identifikation des Zugendes.
- Identifikation des Radendes.

Hinweis Der Name des VI Max & Min von Array (Punkt-für-Punkt) [Array Max & Min PtByPt VI] enthält das Wort Array nur, um den Zusammenhang mit dem Array-basierten VI zu verdeutlichen. Für die Punkt-für-Punkt-Version dieses VIs muss jedoch kein Speicher reserviert werden.

Nachdem die Analyse die Maxima und Minima identifiziert hat, erkennt eine separate Ereignis-Erfassung, wenn diese Werte einen festgelegten Schwellwert überschreiten.

Ereigniskomponenten des VIs Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Die Anwendung protokolliert jedes erkannte Rad und jeden erkannten Zug. Das VI Boolescher Wechsel [Boolean Crossing VI] erzeugt jedes Mal ein Ereignis, wenn das VI Max & Min von Array (Punkt-für-Punkt) ein Radende erkennt. Ebenso erzeugt das Datenerfassungssystem ein Ereignis, wenn das VI Max & Min von Array (Punkt-für-Punkt) ein Zugende erkennt. Durch die Analyse des hochfrequenten Signals kann erkannt werden, welches Rad möglicherweise fehlerhaft ist. Stößt das VI auf ein potentiell fehlerhaftes Rad wird das Ereignis unverzüglich in den Report, der auch das Radende-Ereignis enthält, weitergegeben.

Zwei Punkt-für-Punkt-VIs Boolescher Wechsel [Boolean Crossing VIs] führen folgende Aufgaben durch:

- Erkennung des Übergangs im Signal, der das Radende anzeigt.
- Erkennung des Übergangs im Signal, der das Zugende anzeigt.

Das VI Boolescher Wechsel [Boolean Crossing PtByPt VI] reagiert auf Übergänge. Wenn die Amplitude eines einzelnen Signalverlaufs eines Rades unter einen bestimmten Wert fällt, ist das Radende beim Dehnungsmessstreifen angekommen. Für dieses Datenerfassungssystem ist `3` ein guter Schwellwert um das Radende zu identifizieren. Wenn die Signalstärke unter diesen Wert fällt, erkennt das VI Boolescher Wechsel [Boolean Crossing VI] ein Übergangsereignis und gibt das Ereignis an den Report weiter.

Reportkomponenten des VIs Zugrad (Punkt-für-Punkt) [Train Wheel PtByPt VI]

Die Anwendung zeichnet alle Räder für alle Züge auf, die das Datenerfassungssystem passieren. Das System meldet darüber hinaus jedes Rad, das defekt sein könnte.

Immer wenn ein Rad den Dehnungsmessstreifen überquert erfasst die Anwendung dessen Signalverlauf, analysiert ihn und zeichnet das Ereignis auf. Tabelle 5 beschreibt die Komponenten eines Reports für ein einzelnes Rad eines Zuges.

Tablelle 5. Beispielreport eines einzelnen Zugrades

Informationsquelle	Bedeutung der Ergebnisse
Zählmechanismus für Signalverlaufsereignisse	Erster Vorgang: Rad Nr. 4 hat den Dehnungsmessstreifen passiert.
Analyse der Hochpassfilterdaten.	Zweiter Vorgang: Rad Nr. 4 hat den Dehnungsmessstreifen passiert und könnte fehlerhaft sein.
Zählmechanismus für das Ereignis Ende-des-Zuges.	Dritter Vorgang: Rad Nr. 4 in Wagen Nr. 8 hat den Dehnungsmessstreifen passiert und könnte fehlerhaft sein.

Das VI Zugrad (Punkt-für-Punkt) benutzt die Punkt-für-Punkt-Analyse lediglich um einen Report zu erzeugen, nicht um einen maschinellen Prozess zu steuern. Sie können jedoch dieses Datenerfassungssystem, das die Daten in Echtzeit erfasst, so modifizieren, dass es in Echtzeit reagiert und zum Beispiel Steuersignale ausgibt, die den Zug stoppen, wenn ein potentiell fehlerhaftes Rad erkannt wird.

Fazit

Bei der Echtzeitdatenerfassung hilft die Punkt-für-Punkt-Analyse die Daten in Echtzeit zu analysieren. Eine Punkt-für-Punkt-Analyse findet kontinuierlich und unverzögert statt. Während der Datenerfassung werden die gewünschten Informationen Punkt-für-Punkt gefiltert und analysiert, um entsprechend zu reagieren. Die hier erläuterte Fallstudie demonstriert die Effektivität der Punkt-für-Punkt-Methode für die Erzeugung von Ereignissen und Reporten in Echtzeit.