MATRIXx™

Xmath™ Interactive System Identification Module, Part 2

The MATRIXx products and related items have been purchased from Wind River Systems, Inc. (formerly Integrated Systems, Inc.). These reformatted user materials may contain references to those entities. Any trademark or copyright notices to those entities are no longer valid and any references to those entities as the licensor to the MATRIXx products and related items should now be considered as referring to National Instruments Corporation.

National Instruments did not acquire RealSim hardware (AC-1000, AC-104, PCI Pro) and does not plan to further develop or support RealSim software.

NI is directing users who wish to continue to use RealSim software and hardware to third parties. The list of NI Alliance Members (third parties) that can provide RealSim support and the parts list for RealSim hardware are available in our online KnowledgeBase. You can access the KnowledgeBase at `www.ni.com/support`.

NI plans to make it easy for customers to target NI software and hardware, including LabVIEW real-time and PXI, with MATRIXx in the future. For information regarding NI real-time products, please visit `www.ni.com/realtime` or contact us at `matrixx@ni.com`.

NATIONAL INSTRUMENTS™

April 2004 Edition
Part Number 370759B-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 450 510 3055,
Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838,
Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24,
Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427, India 91 80 51190000, Israel 972 0 3 6393737,
Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Malaysia 603 9131 0918, Mexico 001 800 010 0793,
Netherlands 31 0 348 433 466, New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150,
Portugal 351 210 311 210, Russia 7 095 783 68 51, Singapore 65 6226 5886, Slovenia 386 3 425 4200,
South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51,
Taiwan 886 2 2528 7227, Thailand 662 992 7519, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support Resources and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Contents

Chapter 1

Introduction and Overview

1.1 Introduction

1.1.1 What is ISID ?

ISID is a tool for the interactive identification of discrete-time multi-input multi-output linear time-invariant systems. Given measured data, models are derived using system identification algorithms. These models can then be used to simulate the system or to design a model-based controller for the system. If you are not familiar with system identification, you can get a flavor of the type of problems ISID solves by browsing through the case studies in Chapter 5.
System identification algorithms have been extensively described in the literature. The major identification algorithms have been implemented in the Interactive System Identification toolbox ISID. We refer to the ISID Part 1 manual for more information about system identification and a list of the major references. ISID is a Graphical User Interface (GUI) built on top of the ISID algorithms. ISID's easy-to-use graphical user interface and extensive on-line helps simplify the system identification process.
ISID contains up to 30 different algorithms for processing, identification, validation and displaying results. Even though some of these algorithms are fairly complicated, ISID makes them user-friendly.
This ISID Part 2 manual complements the extensive on-line help system built into ISID. The information in this manual can also be found in the ISID on-line helps.

1.1.2 Organization of the chapters

The chapters in this manual contain the following material.

Chapter 1

This chapter gives a broad overview of ISID. We describe the main concepts and features. As a novice user you should browse through this chapter to get acquainted with the basic ISID concepts.

Chapter 2

The best way to get started with ISID is to try an example. Chapter 2 guides you through a simple example, illustrating many of ISID features[1].

Chapter 3

A complete and detailed overview of ISID is presented in Chapter 3. This chapter is meant for those who want to know more about ISID. It is not required to get started.

Chapter 4

A complete reference list of the algorithms implemented in ISID can be found in Chapter 4. The reference list mirrors the the on-line help.

Chapter 5

Finally, Chapter 5 contains six case studies of industrial processes. As a novice user, it can be instructive to run through them to get an impression of ISID capabilities.

Novice users of ISID should browse through this chapter to get acquainted with basic ISID concepts, then do the example in Chapter 2. You can investigate more sophisticated features (and algorithms) by looking at one or two case studies in Chapter 5. Chapters 3 and 4 serve as reference chapters.

1.1.3 Prerequisites

To use ISID, it is helpful

- to have a user's understanding of Xmath (enough to load your measured data into a file)
- to know the basics of how to interact with an Xmath GUI application (e.g. using a variable-edit box to type in a value; data-viewing and plot zooming)
- to know some system identification basics.

[1] To run this example interactively, go to the main ISID window and select Help → Examples.

An introduction to Xmath can be found in the Xmath Basics manual. How to load your measured data into Xmath is explained in the Xmath Basics manual and is summarized in Appendix A of this manual. There are several ways you can find out about the basics of interacting with an Xmath GUI application:

- Typing `help GUI` in the Xmath command window gives a good introduction.
- Typing `guidemo` in the Xmath command window allows you to start up and play with several GUI demo applications, allowing you to try out sliders, pushbuttons, scroll bars, data viewing, and so on, as you read about them.

More information about system identification can be found in the ISID Part 1 manual and the references therein.

1.1.4 Starting ISID

To start ISID, type:

```
isid
```

in the Xmath command window. Your window manager may require you to position the window using the left or middle button. After the ISID main window appears, you can use Xmath and ISID simultaneously.
ISID's user interface is designed to be intuitive, i.e., things mostly work the way you guess they should, so you should be able to start using ISID with minimal instruction. We recommend that you read this chapter and complete the tutorial in Chapter 2.
ISID includes a very complete on-line help system. Every ISID window has a help menu in the menu bar. The help messages contain complete descriptions of every feature and function in ISID. You can get a good overview of ISID features by scanning the entries in the menu bars and reading the help messages.

1.2 ISID concepts

In this section we give an overview of ISID concepts. We do not explain *how* to do something; we just give a summary of how ISID works and what it does.
Typical problems encountered when identifying data with a classical, command-driven toolbox are:

- finding out which identification functions are available
- learning function syntax
- finding out how the parameters influence the result
- keeping track of the enormous amount of data and information that is generated.

ISID provides solutions for each of these problems. This results in a user friendly interactive tool that gently guides you through the system identification process. To understand how ISID works, you should get acquainted with the intuitive concepts of ISID, which are:

1. data objects

2. building blocks

3. chain

The following sections describe these concepts in detail.

1.2.1 Data Objects

System identification typically requires a fair amount of data-handling, which is organized in ISID by data objects. Data objects bundle data and related information into one object. The five different data objects in ISID are:

- **Input-Output Records:** contain input-output data in the time domain. In most identification problems the data is measured from the plant. Alternatively, the data could be gathered by simulating a model of the plant in SystemBuild or another numerical simulator. Note that ISID only supports identification of models with inputs *and* outputs[2].
- **Frequency Responses:** contain complex data in the frequency domain representing a frequency response or a spectral density function.
- **Impulse Responses:** contain data in the time domain representing an impulse response or a covariance sequence.
- **Models:** contain models of systems (one or more). The end result of an identification session is typically a model data object.
- **Square Roots:** contain intermediate identification results. A square root contains the same information as the the input-output data sequence it was derived from, but in a condensed form[3]. For more information, refer to the ISID Part 1 manual. Novice users should not worry too much about square root objects.

Apart from the actual data (the matrix containing the numerical values), data objects also contain the following information:

- sampling time and unit

[2]For time series analysis use the ISID function `sst`.

[3]The data is condensed using an RQ factorization of a Hankel matrix of the data. The square root object contains the R-factor of this decomposition.

- channel names and units
- data object history, which consists of a textual description of how the object came about. This eliminates the need to manually track an identification session. For more information about the history concept, see Section 3.2.2.
- data object name. Within ISID the data objects are uniquely named. Objects can be selected and manipulated using their names.

Data objects are typically used for three purposes:

- At the beginning of an identification session, when an Xmath variable is loaded into ISID from Xmath, it is internally converted to a data object. When the new data object is formed you can enter the sampling time (unit) and channel names (units). Section 3.2.1 describes how data is loaded from Xmath in detail.
- Data objects are also used to transfer information between different algorithms. As will be explained below, one (or more) data objects act as inputs and outputs to algorithms.
- Finally, at the end of an identification session data objects can be converted to Xmath variables and saved in the Xmath workspace.

This conceptual description is all you need to know about data objects to be able to work with ISID. The internal structure of the object will never be visible to the user, and thus is of no further importance.

1.2.2 Building Blocks

The second major ISID concept is **building blocks**. On an abstract level, a building block is an operator with inputs and outputs. A building block operates on its inputs to calculate its outputs. Within ISID, building blocks have the following structure (see also Figure 1.1):

- **Inputs:** one or more data objects
- **Operation:** determined by a set of parameters
- **Outputs:** one or more data objects

Each building block has an algorithm window associated with it. This window displays information about the building block and allows for graphical interaction to set some of the parameters.
ISID has two different types of building blocks :

- **Data boxes** with an Xmath variable as input
- **Algorithm building** blocks with ISID data objects as input

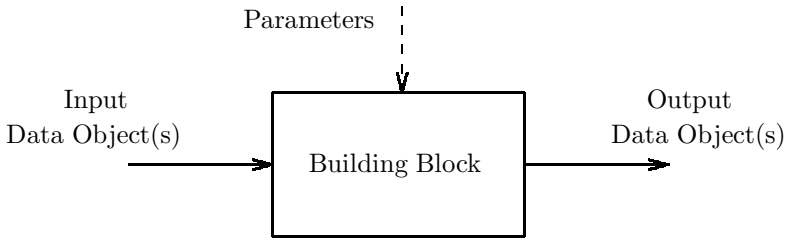


Figure 1.1 An ISID building block. The inputs and outputs are data objects. The operation of the block is determined by a set of parameters.

Data Boxes

Data boxes are always found at the beginning of the chain. Every time an Xmath variable is loaded into ISID, a data box is generated (see Section 1.2.3). Figure 1.2 shows an example of a data box containing an input-output data object.
There are five types of data boxes, each containing one of the ISID data object types, and each requiring a specific type of Xmath variable as input:

- **IO Data Record box:** loads a real matrix or a real parameter dependent matrix (PDM), with minimally 10 elements in the domain. For the PDM, at least one of the sizes must be equal to one.
- **Frequency Response box:** loads a complex PDM with at least 10 elements in the domain.
- **Impulse Response box:** loads a real PDM with at least 10 elements in the domain.
- **Model box:** loads a discrete time Xmath system with at least one state.
- **Square Root box:** loads a Least Squares square root object (see the ISID Part 1 manual).

Algorithm building blocks

There are 4 classes of algorithm building blocks: Processing, Identification, Validation and Display. Each of the algorithm blocks operates on an input data object to create the output object. Input can come from a data box or from another algorithm.
Figure 1.3 shows an example of an identification algorithm building block. The block itself is represented by an icon. The algorithm window associated with the building block is shown beneath.

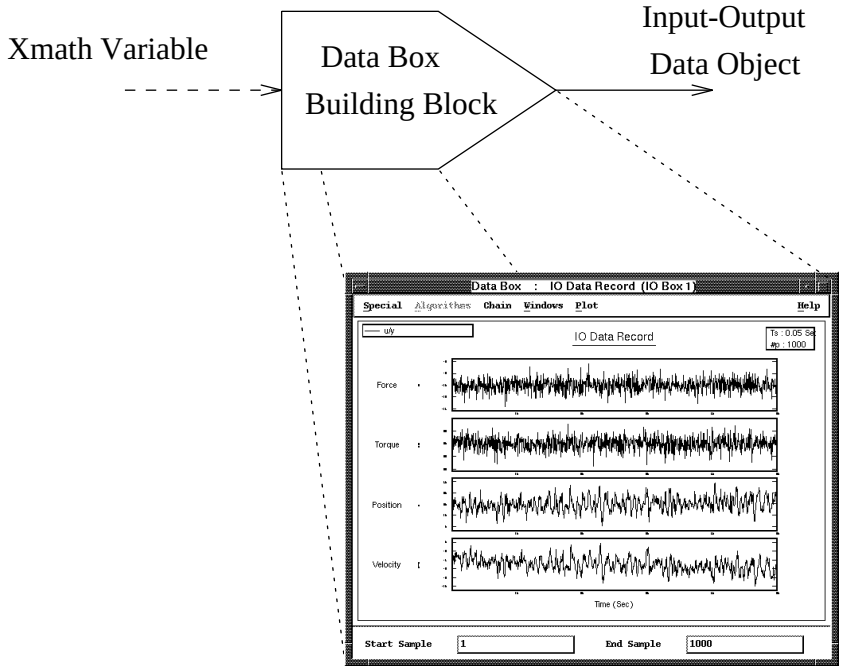


Figure 1.2 An example of a data box. The input of the data box is an Xmath variable, the output is an ISID data object. The associated algorithm window allows graphical interaction with the parameters.

A complete reference list of building blocks (algorithms and data boxes) and their parameters can be found in Chapter 4.

1.2.3 Chain

ISID connects building blocks one to another. The output of one building block is used as the input to another building block. In this way, different building blocks can be connected in series and in parallel. The result is called a **chain** (of building blocks).
Figure 1.4 shows an example of a chain. Each building block is represented by an icon in the Chain Plot. The data box icons are at the beginning of the chain and the algorithm icons follow, connected back to front.
The chain thus consists of a "causal" connection of algorithms in series or parallel (no feedback). Whenever either the input object or the parameters of any of the algorithms in the chain change, these changes are propagated causally (from left to right) through the chain.

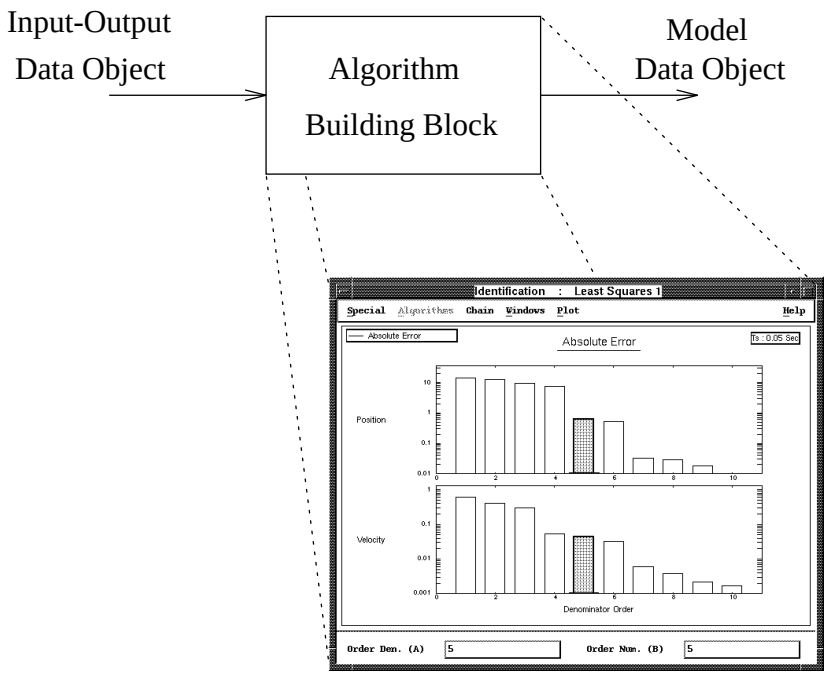


Figure 1.3 An example of an algorithm building block. The block itself is represented by an icon (the rectangular box). The input and output data objects are indicated by the arrows. The associated algorithm window allows graphical interaction with the algorithm parameters.

For instance, when the output of an algorithm (say A) is used as input for another algorithm (say B), then these algorithms are coupled and thus form a part of the chain. Every time a parameter in algorithm A changes, a new output of A is calculated which in turn triggers the recomputation of algorithm B. This procedure is not restricted to two algorithms, and it is thus possible to couple many algorithms back to front, to form a chain.

Typically, the front end of a chain consists of data boxes containing variables loaded from Xmath. The back end consists of validation or display algorithms. The intermediate part of the chain consists of processing algorithms in series and one or more identification algorithms in parallel.

The graphical interaction with the chain and with the parameters of the building blocks allows for a fast inspection of the influence of certain parameters on the quality of the identified model. More details on how to build chains can be found in Section 3.3. The chain execution is described in detail in Section 3.4.

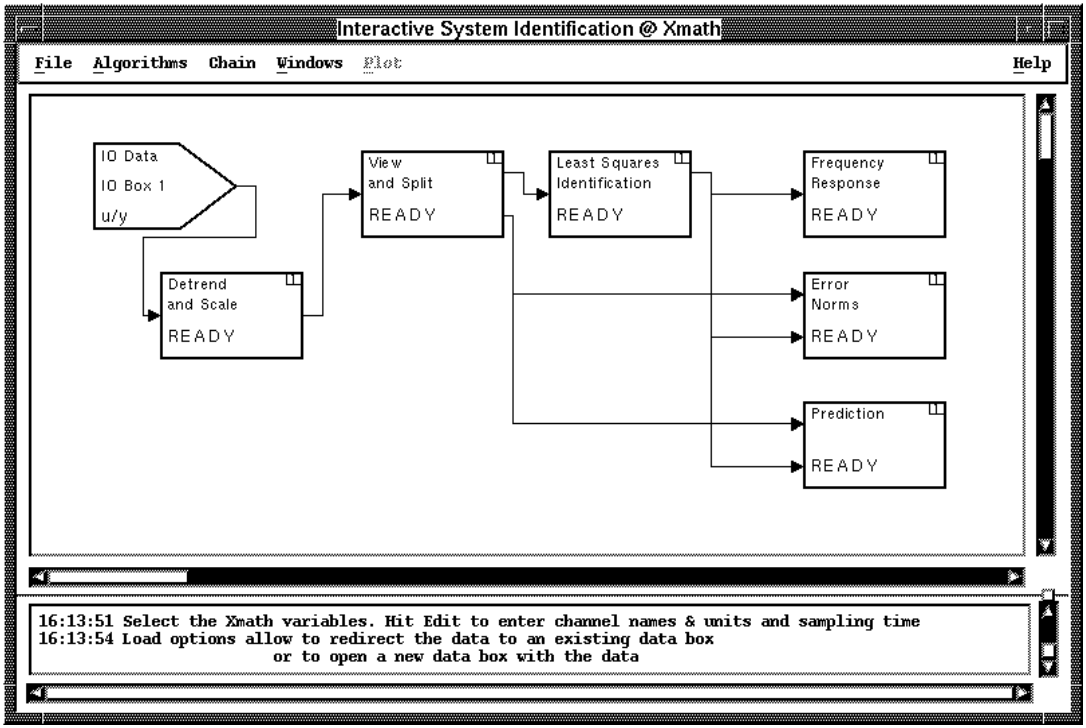


Figure 1.4 An example of a chain. Each icon represents a building block. The data box (arrow shaped) is located at the beginning of the chain. The algorithms are connected back to front.

1.3 ISID Windows

In this section we describe the window anatomy of the typical ISID windows. There are two main types of windows. The first type displays information about the chain and is called the Chain Window. There can be only one Chain Window. The second type is the algorithm window. This is the window that is associated with each of the algorithms. Multiple algorithm windows are possible.

1.3.1 The Chain Window

The Chain Window is displayed in Figure 1.5. From top to bottom, it consists of the following fields:

- **A menu bar.** The menus (File, Algorithms, Chain, Windows, Plot and Help) give access to the ISID functionalities and on-line helps. The menu bar allows you

to:

  - load data in ISID
  - load and save ISID chains
  - open algorithms
  - manipulate the chain execution
  - manipulate the placing of the algorithm windows
  - make a hardcopy of the Chain Plot
  - get extensive on-line help about ISID.

- **The Chain Plot.** This plot displays the chain building blocks as icons. Algorithm icons are rectangular boxes, while data box icons are arrow shaped boxes. Graphical interaction with the Chain Plot allows you to:

  - make and delete connections between building blocks
  - delete building blocks
  - move icons
  - acquire information and on-line help about connections and building blocks
  - disable and enable algorithms
  - display algorithm windows associated with icons
  - manipulate the chain execution
  - manipulate the placing of the algorithm windows.

  Graphical interactions possible with the Chain Plot are listed in Section 3.5.

- **A Message Area.** Hints about what to do next are displayed in the Message Area.

- **Four sliders.** Two horizontal and two vertical sliders allow you to scroll the Chain Plot and the Message Area.

1.3.2 An Algorithm Window

A typical algorithm window is displayed in Figure 1.6. From top to bottom it consists of three areas:

- **A menu bar.** The menus give access to ISID functionalities and on-line help. The menu bar allows you to:

  - save the output of the algorithm to Xmath

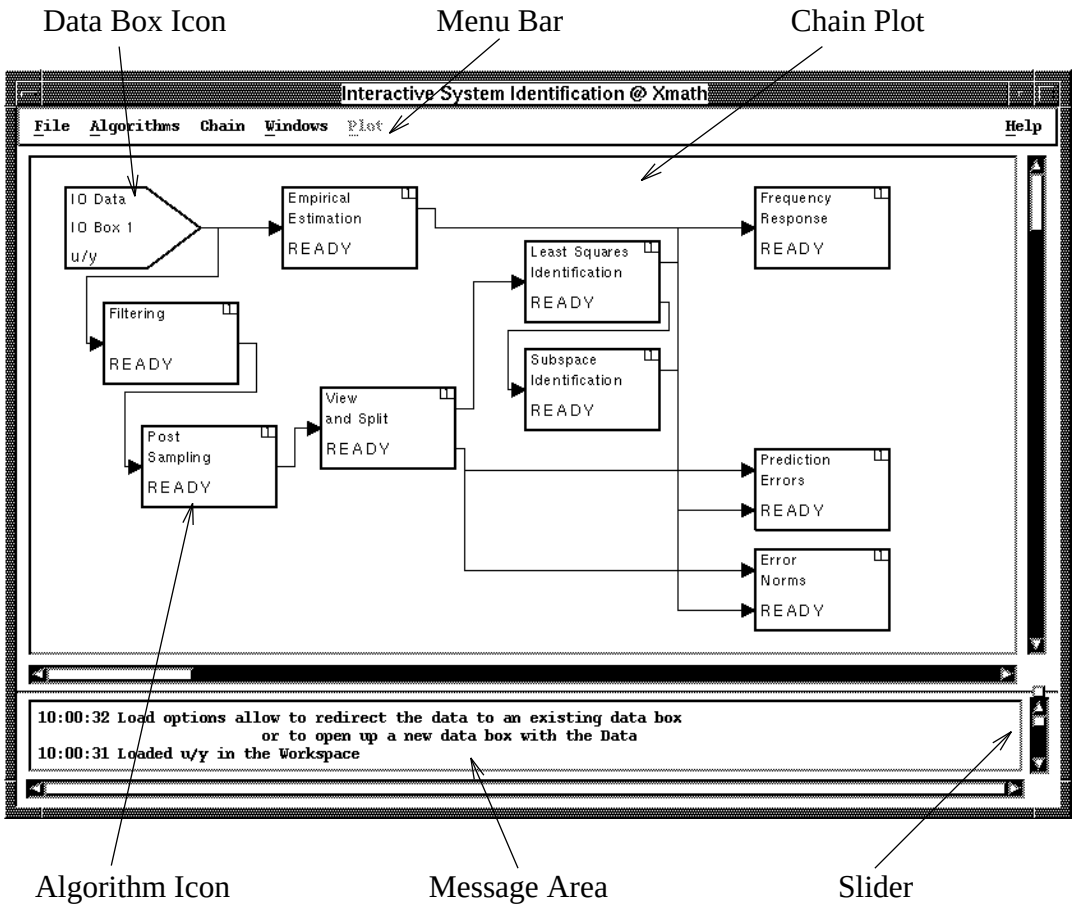


Figure 1.5 The Chain Window and its components: the menu bar, the Chain Plot (with data box icons and algorithm icons), the message area and four sliders.

  - disable and enable the algorithm
  - close the algorithm
  - alter the plot appearance (zoom, scale, ...)
  - make a hardcopy of the plot area
  - alter the default parameters
  - get on-line help about the algorithm
  - manipulate the chain execution
  - manipulate the placing of the algorithms

- **A plot.** The plot area displays information about the algorithm and allows for graphical interaction with the parameters. A detailed description of each algorithm window is given in Chapter 4. The plot is built up from the following parts (see also Figure 1.6):

  - A **plot area** displaying one or more data objects as curves or bar graphs. Graphical interaction with this plot area typically allows you to adjust certain parameters and zoom in on one subplot or on a specific part of the data. The axes of the plot can be set interactively. All classical GUI plot interactions (data viewing, magnifying glass) are available in the plot area. For a complete list of plotting features, see Section 3.7.
  - A **legend** (top left) which displays the names of the data objects with the corresponding line styles. Some algorithms (see Chapter 4) also allow graphical interaction with the legend.
  - A plot **title**.
  - An **information area** (top right) which displays the sampling time (`Ts`) and/or the Nyquist frequency (`fN`) and/or the number of data points (`#p`).
  - Multiple **channel labels** and **axis labels**.

- **A parameter area.** In this area, algorithm parameters are entered numerically (instead of graphically through interaction with the plot).

1.4 Using ISID

An overview of a typical ISID session is depicted in Figure 1.7. ISID can be used in many different ways. As an example you might:

- interactively identify a model from input-output data, frequency response data or impulse response data

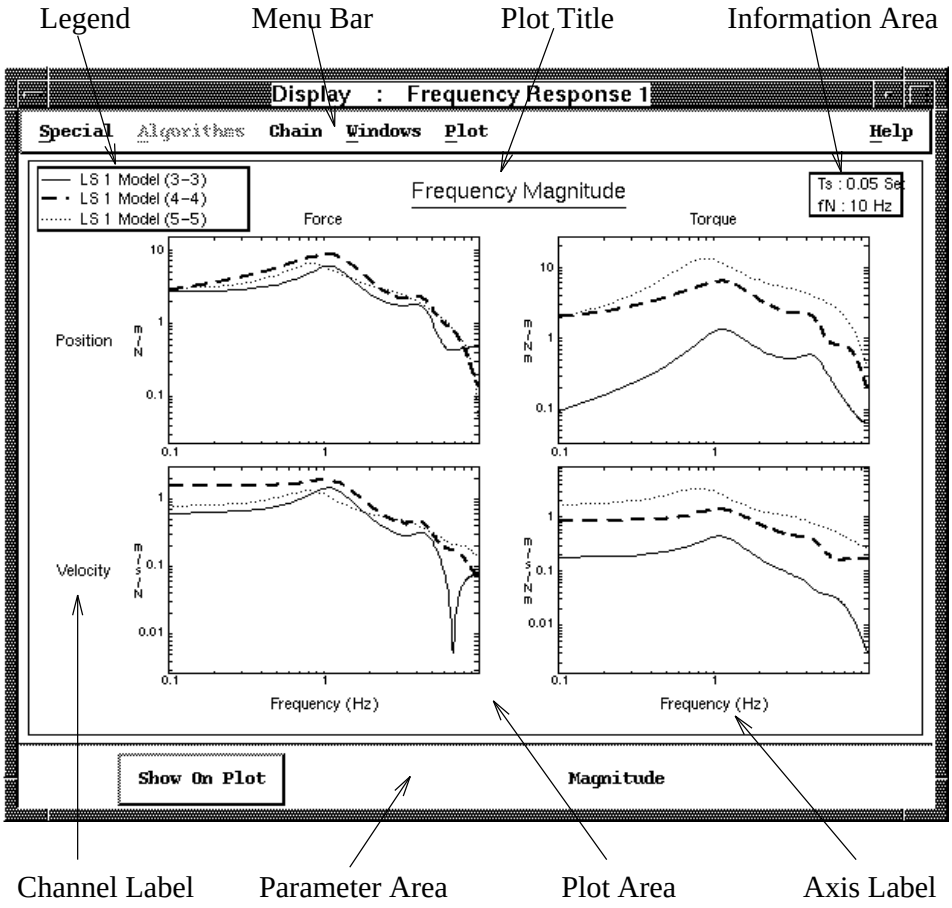


Figure 1.6 An algorithm window and its components: the menu bar, the plot (consisting of the plot area, the legend, the title, the information area, channel and axis labels) and the parameter area.

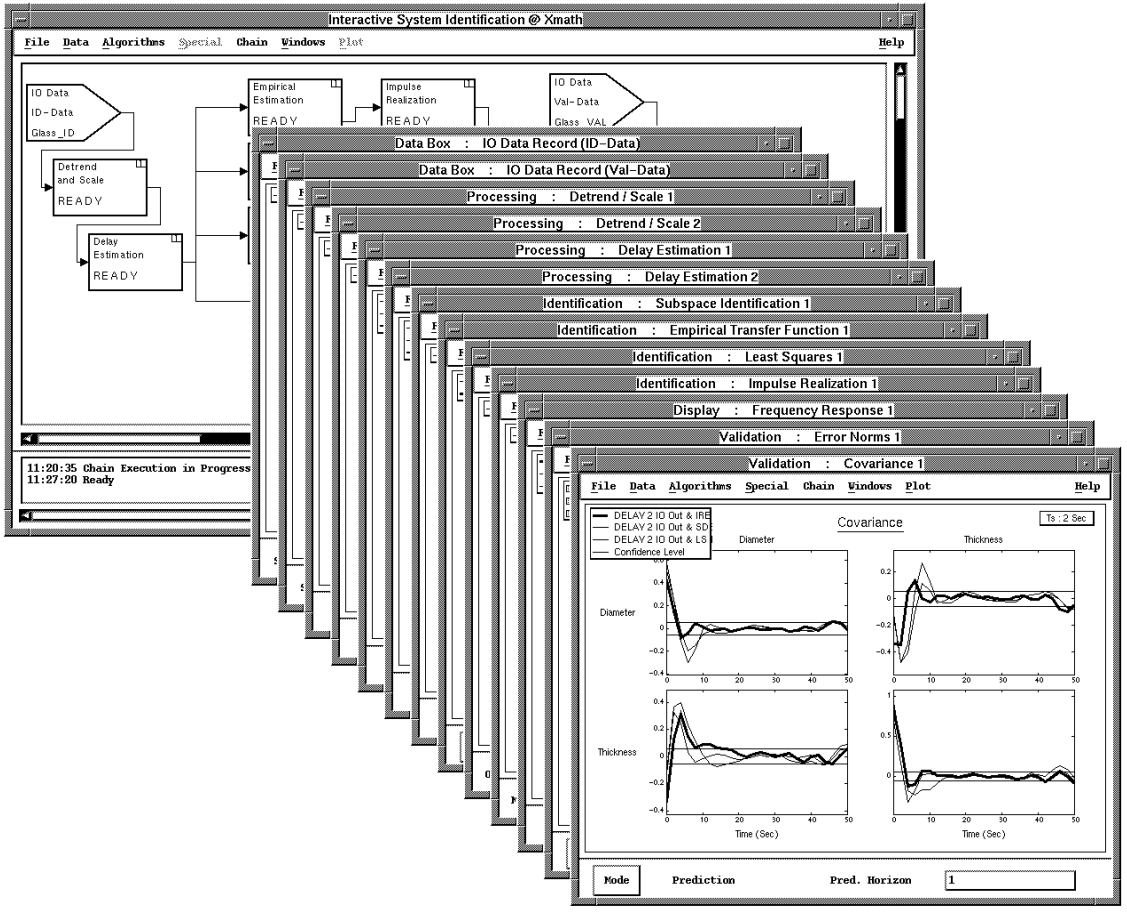


Figure 1.7 An overview of an ISID session. The Chain window is shown in the background. In the foreground, we see the algorithm windows which are automatically ordered as a "card box". There are several utilities to manipulate these windows, details of which can be found in Section 3.7.

- compare the results of different identification techniques (the more these results are alike, the more you can trust your models)
- pre-process data measured from an industrial plant to a form suitable for linear system identification
- cross-validate identified models on validation data
- display several properties of the identified models
- interactively analyze the influence of parameter changes on the resulting model quality
- save the identified model to Xmath and use it (after conversion to a continuous time model) in the Interactive Control Design Module (ICDM)
- linearize your SystemBuild non linear model around a working point.

The most common tasks are: interactively identifying a model from input-output data and interactively analyzing the influence of parameter changes on the resulting model quality. Both novice and advanced users can complete these tasks.

Novice User

ISID allows a novice user, who is not yet familiar with system identification to use a default chain to identify reasonably good models for a great number of practical identification problems. ISID has three built-in default chains. the "Least Squares Identification" chain is automatically loaded at startup. The other default chains can be easily loaded (see Section 3.2.3 for a more detailed description).
The idea behind the default chains is explained in Figure 1.8. The user loads one of the three default chains, loads data and runs the chain. Only one or two major parameters have to be adjusted in the default chains, the order of the identified model(s) in particular. Every time the parameters are changed the model is recalculated. Default chains make it fairly easy to find good models in a very short time span. In Chapter 2, we illustrate this with an example.

Advanced User

Once you have mastered the basic ideas of ISID (which does not take long at all), you are an advanced user. As an advanced user, you will want to open new algorithms to build chains tailored to your own problem. ISID contains up to 30 algorithms you can use with a minimum of background knowledge about system identification. While you are building the chain you can interactively examine the effect of parameter changes on the (intermediate) results. Figure 1.9 illustrates this principle, which will also be illustrated in Chapter 2.

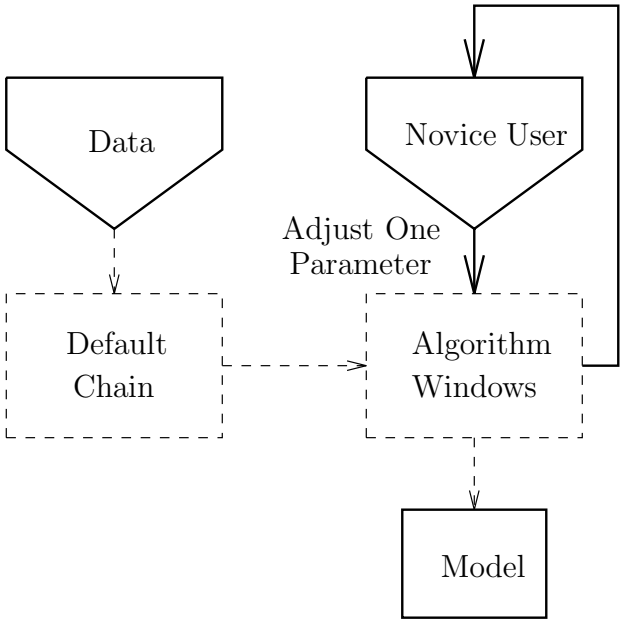


Figure 1.8 The novice user only has to load data and run the default chain. ISID will automatically generate the algorithm windows. By adjusting one (or two) parameter(s) the user is able to obtain a reasonable model (this is represented by the feedback arrow from the algorithm windows to the novice user). The dashed lines indicate tasks performed by ISID, the thick lines indicate user responsibilities, which are, in this case limited to the choice of one parameter.

Note that the default chain, which is automatically loaded at startup, can be easily deleted, so that a new chain can be built (use Chain → Clear). Furthermore you can start ISID without the default chain by typing:

```
isid,[]
```

in the Xmath command area. More details can be found in Section 3.8.
ISID allows you to save and load chains, so that these "graphical programs" can be stored for later use or can be exchanged between different users.

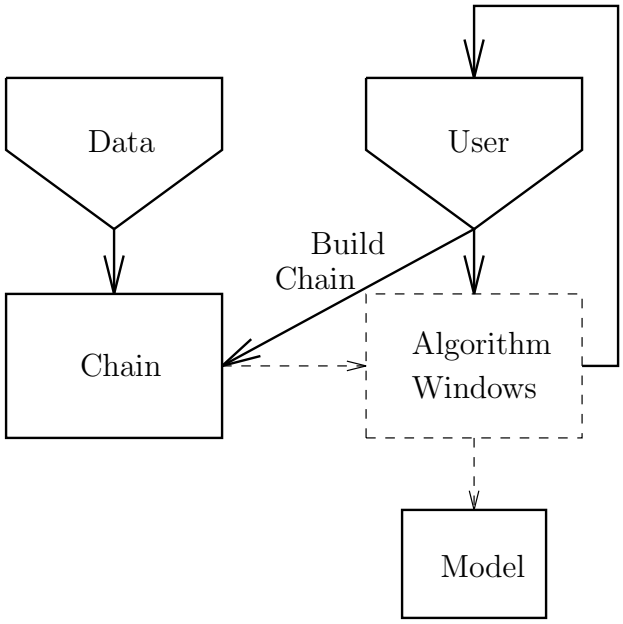


Figure 1.9 The user builds a chain to optimize the quality of the identified model. She has to decide which algorithms to put in the chain and how the algorithms will be connected. ISID opens the algorithm windows automatically. Based on the information in these windows, the user adjusts the parameters of the algorithms to obtain an optimal result.

Chapter 2

Tutorial Example

You have just entered the world of ISID and you want to get started quickly. This chapter guides you step by step through a simple identification example. Although you can do this example with no prior introduction, you will benefit from reading Chapter 1 to get acquainted with the basic concepts and nomenclature of ISID.
Section 2.1 assumes a novice user who is not very familiar with system identification. We use the default startup chain to identify a model from given input-output data. The amount of work and experience required is minimal.
After reading the first section you will be ready to build your own chain. Section 2.2 explains, with the same data, how the results of the first session can be optimized using the more sophisticated ISID features.
Before starting, note that when we say "click with the mouse", you should click once with the *left* mouse button (unless indicated otherwise).

2.1 Using a Default Chain

As a starting point of any identification session, you need to load data into the Xmath workspace. In Appendix A we describe in detail how you can read non-Xmath data into Xmath.

1. To load the tutorial data type:

   ```
   load file="$XMATH/modules/isid/demo_data"
   ```

   Typing `who` in the Xmath command area, reveals that you have loaded two matrices of size 1000×2, which represent the inputs (`u`) and the outputs (`y`). The tutorial system has two inputs and two outputs. For this example, the input names are "Force" (unit N) and "Torque" (unit Nm). The output names are "Position" (unit m) and "Velocity" (unit m/s). The sampling time is 0.05 seconds.

2. To start ISID, type:

```
isid
```

in the Xmath command area. The chain window containing the startup default chain is realized (Figure 2.1)[1]. The Least Squares Identification default chain is one of the simplest and most robust default chains. It leads to satisfactory models for most practical identification problems.

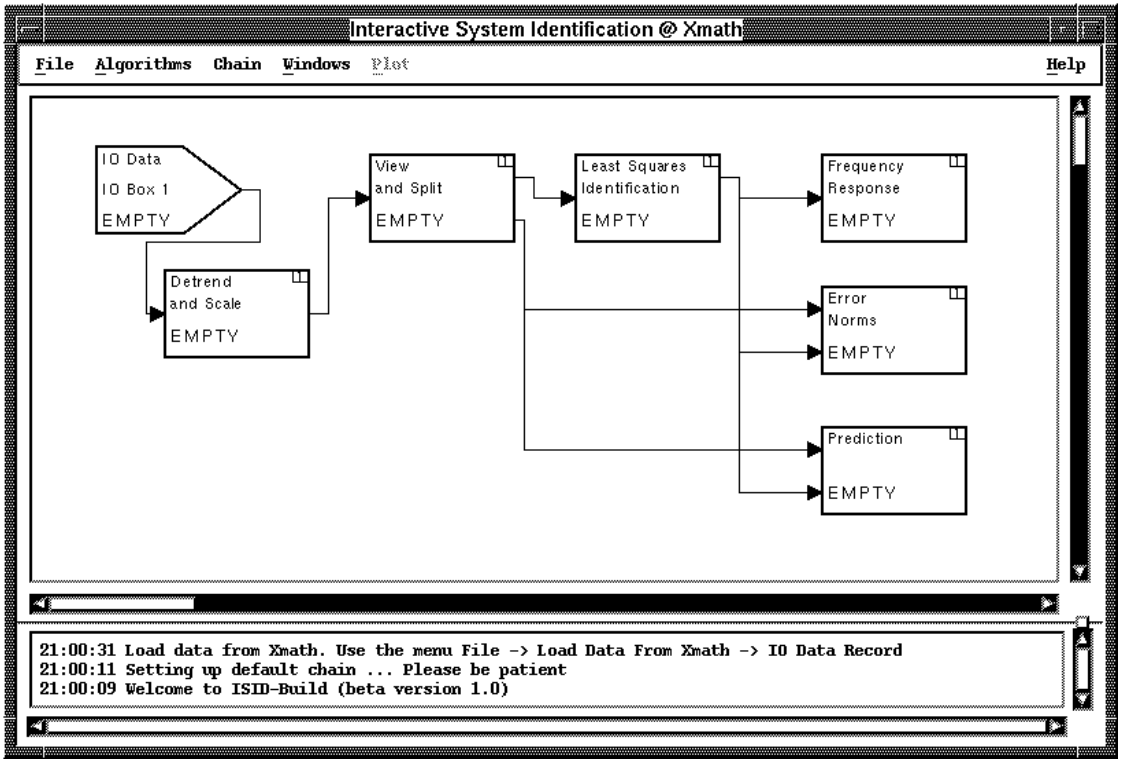


Figure 2.1 The Least Squares Identification default chain.

The top left box in the Chain Plot is the data box. It is arrow-shaped and will soon contain the tutorial input-output data. Its name is `IO Box 1`; the number in the name refers to the fact that there can be several IO Data Boxes, as we will see later. All other rectangular icons in the Chain Plot represent algorithms. There are:

[1]Note that at this point you could load another default chain using the menu entry File → Load Default Chain (see Section 3.2.3). The new chain would clear the current chain.

- two processing algorithms: **Detrend and Scale** and **View and Split**
- one identification algorithm: **Least Squares Identification**
- two validation algorithms: **Error Norms** and **Prediction**
- and one display algorithm: **Frequency Response**

3. As suggested in the message area at the bottom of the Chain window, select File → Load Data From Xmath → IO Data Record to load the tutorial data. Selecting this menu pops up the "LOAD VARIABLE" window in Figure 2.2.

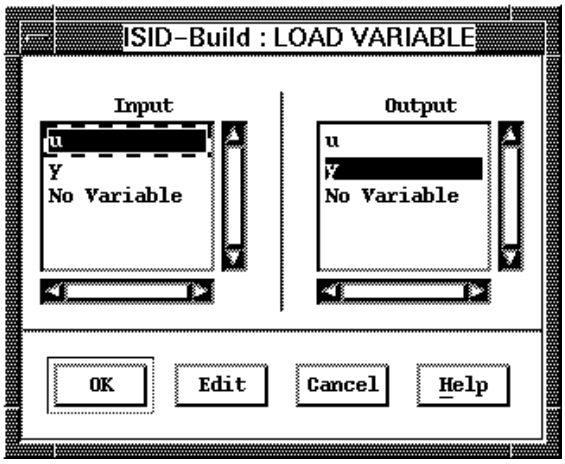


Figure 2.2 The "LOAD VARIABLE" window allows you to select the Xmath variables that serve as input and output.

The "LOAD VARIABLE" window allows you to select the Xmath variables that serve as input and as output. Take `u` as the input and `y` as the output. Click on the variable names to choose. Make sure the selected variables correspond to the selections in Figure 2.2, but do not press OK yet.

4. Now you will enter some information about the inputs and outputs. Hit the Edit button. This pops up the "LOAD NAMES & UNITS" window that allows you to enter the channel names and units and the sampling time and unit. Eventually the window should look like Figure 2.3. To enter text, click on one of the text areas, then enter your own text. After you have entered the text you must *always* hit `return`, before moving to another text area. When the window matches Figure 2.3, press the OK button[2].

[2]Note that these names and units are automatically default to sensible values. It is possible to skip the previous step. The names are than set to their default values.

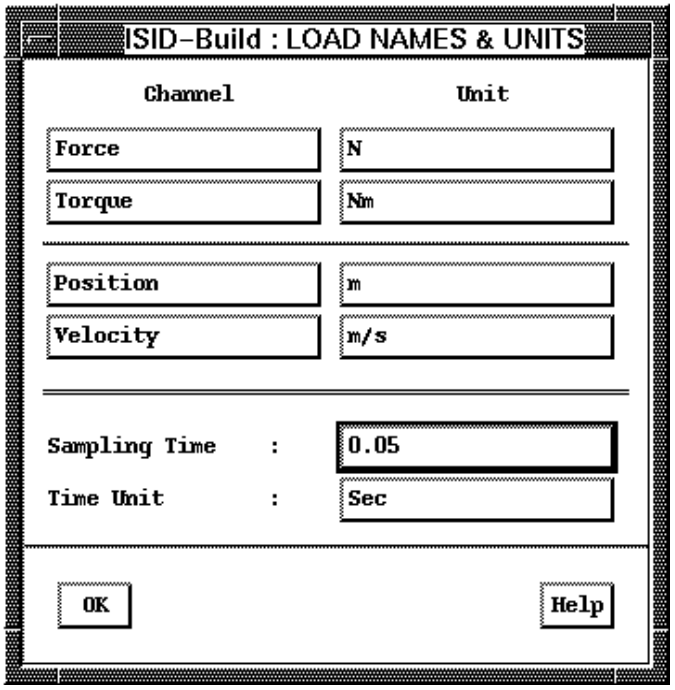


Figure 2.3 The "LOAD NAMES & UNITS" window allows you to enter the channel names and units and the sampling time and unit of the data. These names and units are automatically defaulted to sensible values.

Now you have entered all information about the object you want to load (Xmath variables, names and units). Hitting the OK button of the "LOAD VARIABLE" window (Figure 2.2) loads the Xmath variables into a data object. The program now has to decide what to do with this newly created data object.

5. The "LOAD OPTIONS" window pops up (Figure 2.4). This window allows you to redirect the loaded data to an existing data box (`IO Box 1` in this case) or to create a new data box. Since it is our intention to put the data in the default chain, we will redirect the loaded data to the existing data box. Leave the default selection as it is and hit the OK button in the "LOAD OPTIONS" window.

   After you have redirected the loaded data to `IO Box 1` in Figure 2.1, the chain starts executing. This means that the algorithms in the chain are executed one by one. While the chain is calculating, sit back and watch the "execution-wave" ripple through the chain. You can see the text in the icons change from "EMPTY" to "WORKING" to "READY". If you have a color monitor, you can also see the

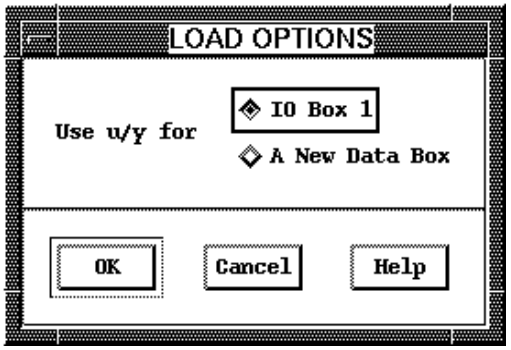


Figure 2.4 The "LOAD OPTIONS" window allows you to determine the destination of the loaded data.

algorithms that are executing change color. Every time an algorithm is done, its associated algorithm window pops up, displaying specific results for the algorithm.

When the chain has finished execution, your screen should look like Figure 2.5. Double clicking on one of the icons in the Chain Plot, raises the corresponding window.

6. Inspect the window corresponding to `IO Box 1`. Double click on the IO icon; the window containing the data pops up. Let us examine some of the plotting features (all algorithm plots share the same features). To look at one channel of the data, hold down the `<Ctrl>` key and click in the background of the subplot you want to see. Doing this for the first output results in Figure 2.6. To go back to the original plot, hold down the `<Ctrl>` key and click in the background. You can toggle back and forth between the two plots by holding down the `<Ctrl>` key and clicking on the title of the plot ("IO Data Record" for this plot).

   To zoom in on part of the data, hold down the `<Ctrl>` key and click and drag the left mouse button in one of the subplots. A rectangular box appears. When you release the mouse button, the zoomed part of the data becomes visible. To go back to the original plot, hold down the `<Ctrl>` key and click in the background of the zoomed plot or on the title.

   See Section 3.7 for a complete overview of zooming capabilities.

7. Let's return to the chain and the identification process. Going over the chain from left to right, you see that the loaded data is first detrended and then split into two parts: an identification part that is redirected to the least squares identification algorithm, and a validation part used in the validation algorithms. Finally, the frequency response of the identified model is displayed.

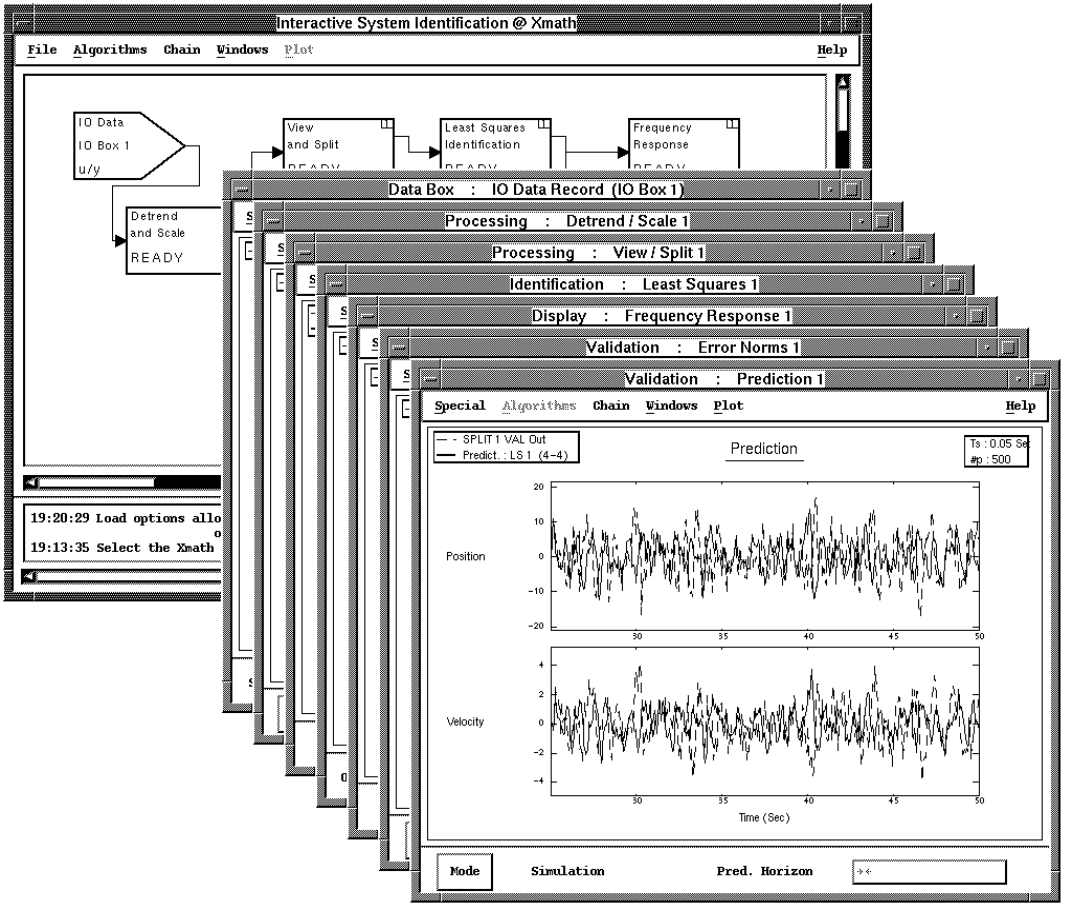


Figure 2.5 The Least Squares default chain after it is run with the default data.

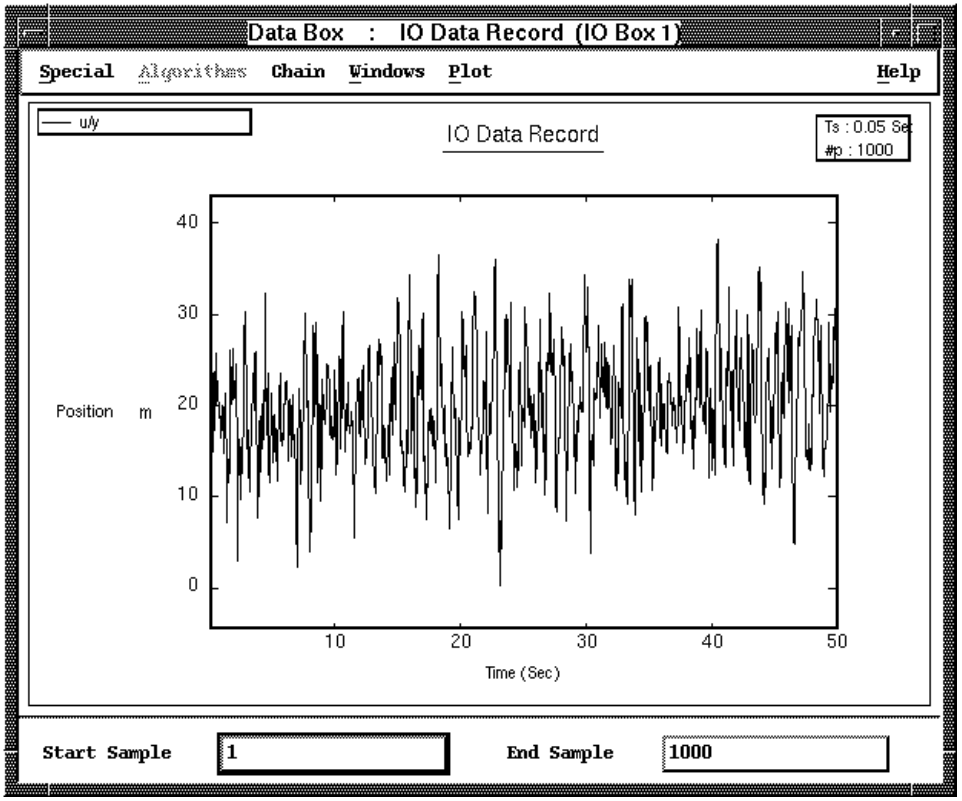


Figure 2.6 The window corresponding to the Input Output data box, with only the first output plotted.

Double click on the **Prediction** icon. A window displaying the validation data ("SPLIT 1 VAL Out") and the simulated data using the least squares model ("Predict.: LS 1 (4-4)") pops up (see Figure 2.5). Clearly the original and predicted data are not very close. This is mainly because the order of the identified model is too low (the default is 4). For default chains, the model order typically needs to be adjusted to obtain a good model.

8. Double click on the **Least Squares Identification** icon to raise the identification window. The order of the system is set to 4 (the highlighted bar and the number at the bottom of the window). You can easily change this order by either entering a new order in the parameter area at the bottom, or by clicking on the bar corresponding to the desired order. More than one order can also be selected by entering a vector of numbers (`[3:5]` for instance), or by clicking and dragging

with the left mouse in the plot.

Assume you want to inspect all orders from three to five. In either plot, click with the left mouse on the bar corresponding to order three, and drag the mouse until the pointer is located above the bar corresponding to order five, then release the button. Since you have changed one of the parameters of an algorithm in the chain, the chain automatically re-executes. All algorithms after the **Least Squares Identification** algorithm recalculate and update their results.

9. The most interesting algorithm (in this case) is the **Prediction** algorithm. Raise its window by double clicking on the corresponding icon. Figure 2.7 shows the **Prediction** algorithm window. This window still displays two curves, but now there are 4 entries in the legend of the plot (top left corner). The first legend entry is the original data. The other entries correspond to the three models that were identified using the least squares algorithm. If you click with the left mouse on the text of one of the legend entries (for instance click on the text "Predict. LS 1 (3-3)"), you see another predicted signal show up. In this way, you can easily compare the predictions and thus compare the quality of the different models.

   Click on each of the three legend entries (one by one) and you will find out that the fifth order model gives the best results. The fifth order model is thus a reasonable model for the data. Now that you have found this model, save it for later use. The output of any ISID algorithm can be saved by using the menu entry Special → Save Output to Xmath. Since you want to save the least squares model, you should turn your attention to the algorithm that has this model as its output: **Least Squares Identification**.

10. Raise the **Least Squares Identification** window and select the menu entry Special → Save Output to Xmath. The "SAVE" window in Figure 2.8 pops up. You should first select the output to be saved. Since you found the fifth order model to be the best, click on the diamond to the left of the text "LS 1 Model (5-5)".

    When you hit the button to the right of the text "Comment to Save" (the button that reads "Push To Edit"), the "HISTORY" window pops up (see Figure 2.9). This window displays a textual description of all algorithms and parameters that were active in producing the output. This description is saved to Xmath as a comment of the Xmath variable. You can edit the comment by clicking with the left mouse button in the "HISTORY" window and typing some new text (see also Figure 2.9).

    Hit the OK button in the "HISTORY" window. The only thing left to do is to specify the name of the Xmath variable the least squares model is saved in. Use the text area to the right of the text "Xmath Name" to do so. Enter the name `mod1` as shown in Figure 2.8 and press return. Hit the OK button in the "SAVE"

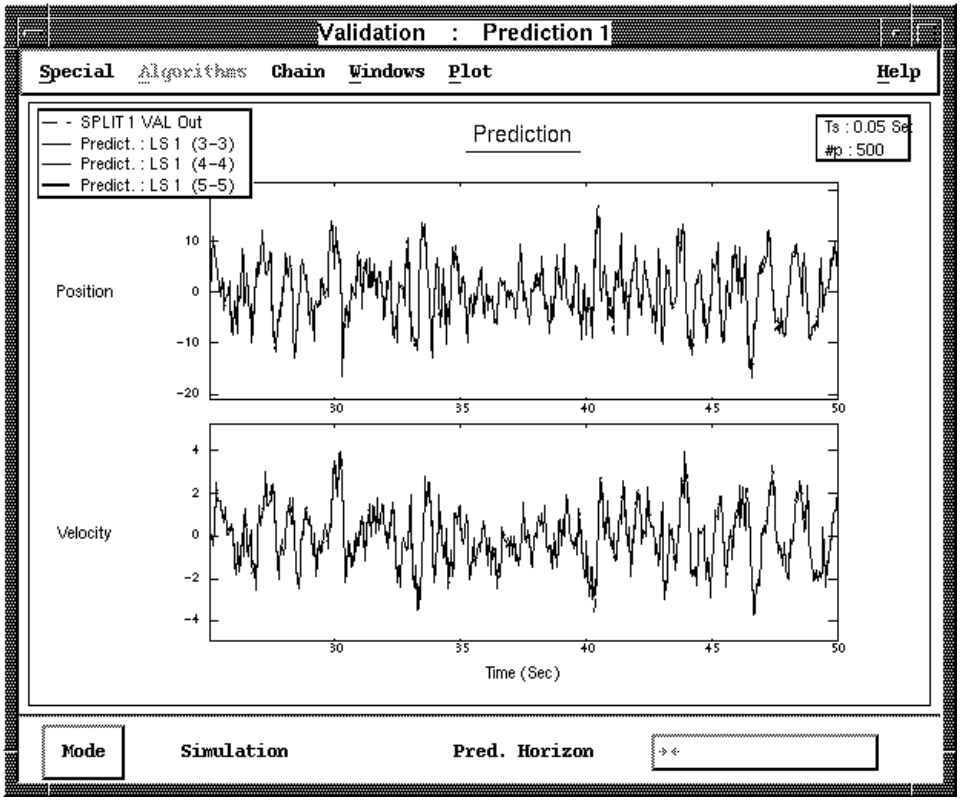


Figure 2.7 The **Prediction** algorithm window. Clicking on a legend entry displays the corresponding predicted output.

window. This saves the ISID object to the Xmath workspace. A dialog box pops up (Figure 2.10) telling where the *input-output model* is saved (the input-output model describes the transfer from inputs to outputs) and where the *innovations model* is saved (the innovations model also includes the noise model). For more information about input-output models and innovation models refer to the ISID Part 1 manual.

When you type `who` in the Xmath command area, you see that there are two new variables: `mod1` containing the input-output model and `mod1_inn` containing the innovations model. To retrieve a comment, use the Xmath `commentof` command. For example, typing:

```
commentof(mod1)
```

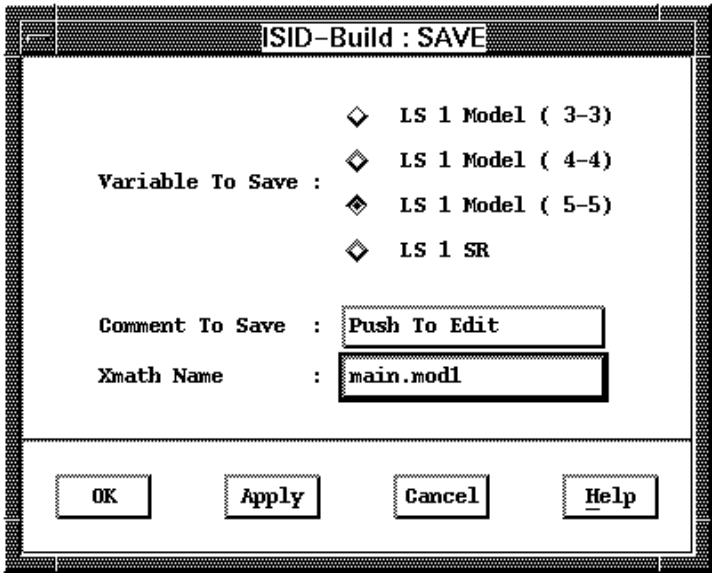


Figure 2.8 The "SAVE" window allows you to save the identified model in two Xmath variables.

displays the history of `mod1`.

These systems can be re-used further. Note that the systems are *not* saved in a file yet. Type:

```
save mod1 "file1"
```

to save the model in a file named `file1.xmd`.

This completes the process of identifying a model from input-output data. You can put your own data into the default chain or use one of the other two default chains; use the menu entry File → Load Default Chain. When you load one of the other default chains, you will see that their setup is very similar, and that there is only one (or two) parameter(s) you need to alter to get reasonable results. This parameter is typically the model order of the identification algorithms.
Eventually, you may need more specialized chains to do the identification. The following section explains how to build your own chain.

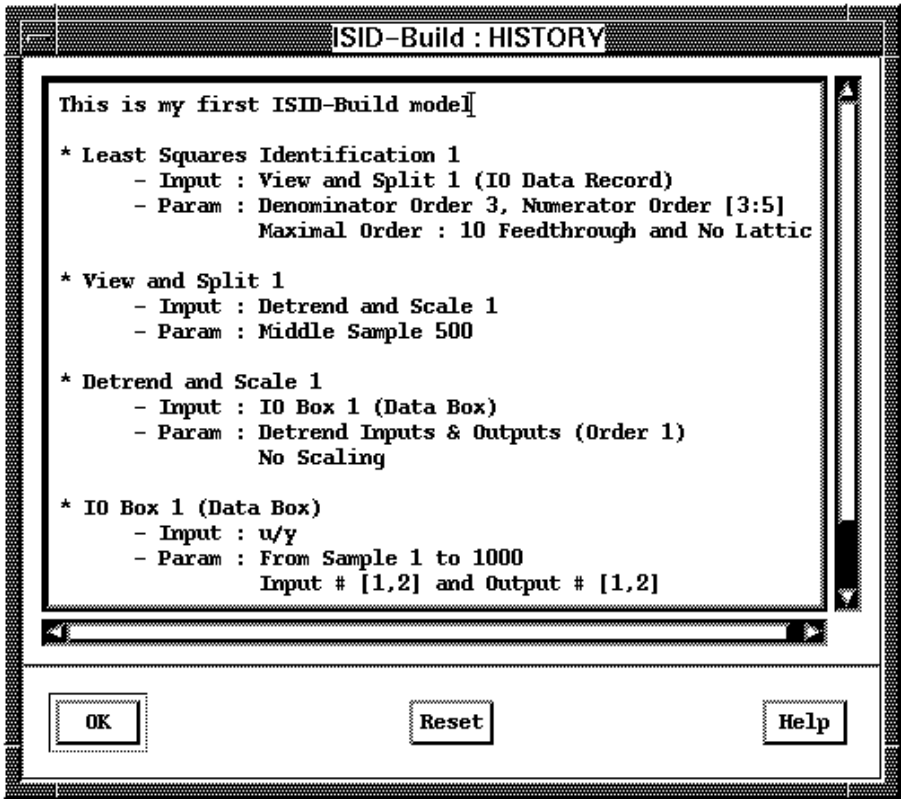


Figure 2.9 The "HISTORY" window displays the comment saved with the Xmath variable. The sentence "This is my first ISID-Build model" has been added.

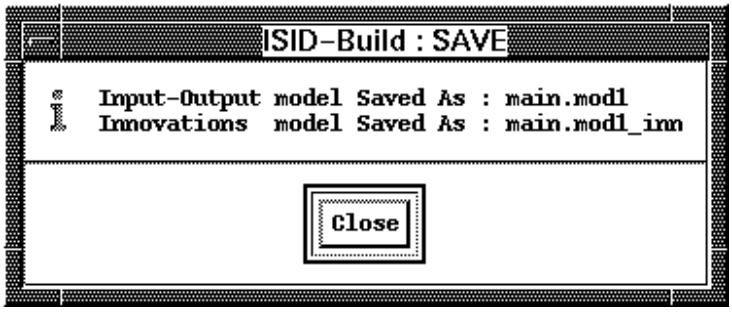


Figure 2.10 The SAVE window informs you that the input-output model is saved in `mod1` and the innovations model is saved in `mod1_inn`.

2.2 Building your own Chain

This section uses the same data as the previous section. You will now build a chain to obtain a more optimal model. Since the explanations in this section are a little more concise, read Section 2.1 first. You should run through this example on your computer since not all intermediate results are shown here.

1. First clear the chain and the Chain Plot with the menu entry Chain → Clear. Alternatively, you can start ISID without a default chain:

   ```
   isid,[]
   ```

2. After you have cleared the chain, load the demo data of the file `demo_data` by selecting File → Load Data From Xmath → IO Data Record. Edit the channel names and units and the sampling time and unit as in Section 2.1 (Figure 2.3). After hitting the OK button in the "LOAD VARIABLE" window (Figure 2.2), a data box is created automatically. There is no "LOAD OPTIONS" window (as in the previous section) because the data can only be redirected to one place, namely a new data box.

3. To position the data box icon in the Chain Plot click on the icon and drag the mouse pointer to the desired location. All icons in the Chain Plot can be positioned in a similar way.

4. You will build up the chain algorithm by algorithm. To know what the final result looks like, see Figure 2.11.

   First raise the algorithm window associated with the data box you just opened (double click on the icon). It is clear from the plot that the average values of the data differ from zero. Looking at a close up of the second output also reveals a trend in the data. You should remove this zeroth and first order trends before proceeding.

5. Select the menu entry Algorithms → Processing → Detrend/Scale. Selecting this entry changes the cursor from an arrow into a cross and displays a rectangular box that follows the cursor. Click to place the **Detrend/Scale** algorithm on the Chain Plot. This pops up the "OPEN" window in Figure 2.12, which allows you to determine the inputs connecting to the algorithm. You can select the connections either by selecting a data object name in the list or by selecting an icon in the Chain Plot with the left mouse button. We refer the reader to Section 3.3.1 for more details. Select `IO Box 1` as the input to the **Detrend/Scale** algorithm.

   Note that optionally you could change some of the default parameters. Hitting the Default button of the "OPEN" window pops up a window containing the

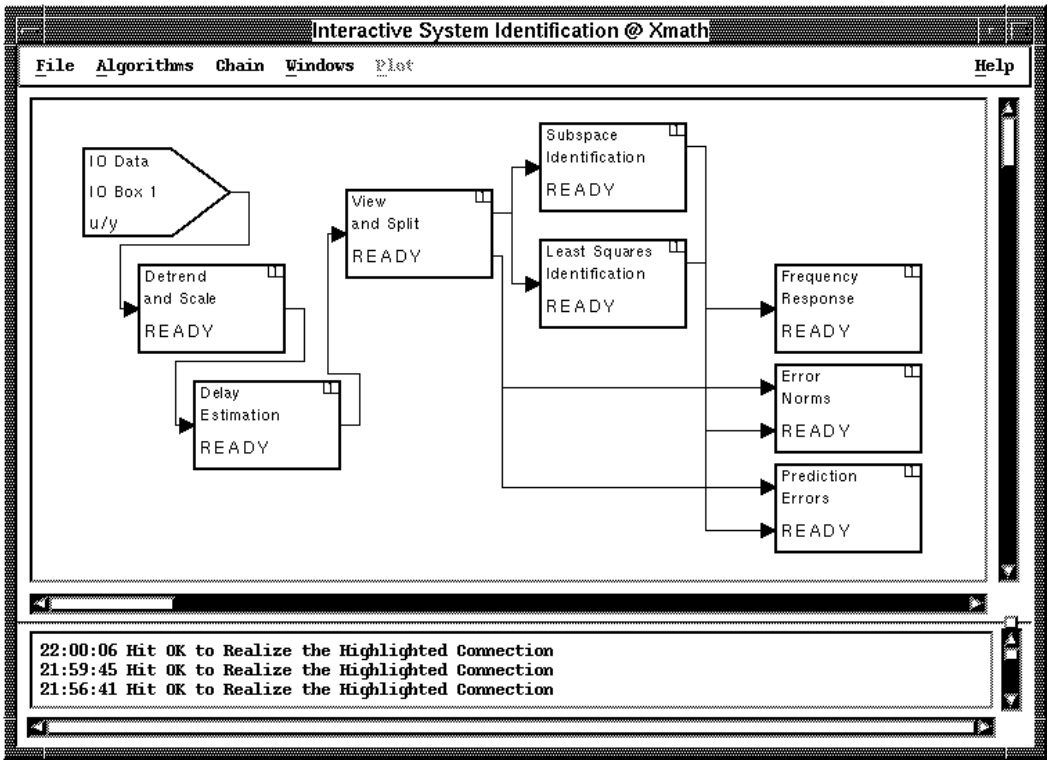


Figure 2.11 Final chain obtained at the end of this section.

parameters. You can alter these parameters as you see fit. Section 3.3.1 gives more details on this subject.

Hit the OK button in the "OPEN" window. The icon changes its state from "OPEN" to "WORKING" to "READY". You have opened your first ISID algorithm and have built the first part of your own ISID chain. If you are not pleased with the result, you can change the parameters of the **Detrend/Scale** algorithm in the parameter area. For the default parameters use the menu entry Special → Defaults.

6. It is time to open your second algorithm. Assume you know that there is a fair amount of delay present in the process. You can compensate this delay with the **Delay Estimation** algorithm (Algorithms → Processing → Delay Estimation). The procedure to place the algorithm is similar to step 5. When the "OPEN" window pops up, define the inputs to the algorithm and (optionally) adjust the default parameters. From Figure 2.11 it should be clear which connection to make

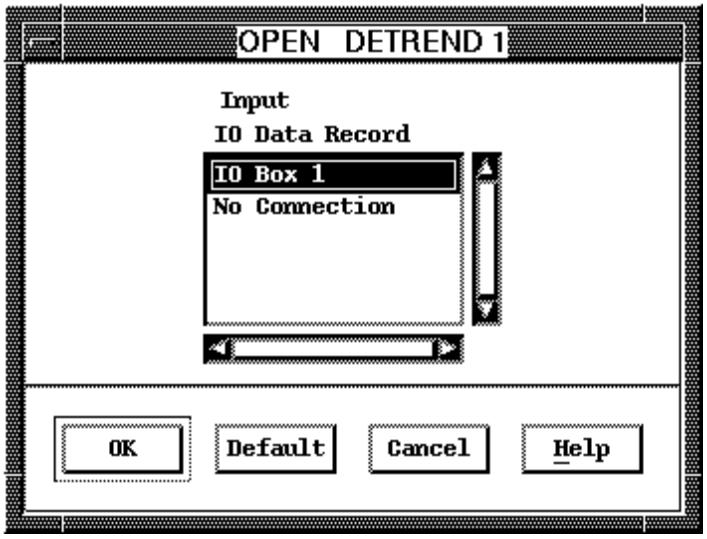


Figure 2.12 The "OPEN" window allows you to connect an open algorithm.

("DETREND 1 Output"). Select the right input data object, then hit the OK button.

After the **Delay Estimation** algorithm has finished execution, the algorithm window shown in Figure 2.13 pops up. The help entry in the menu bar of this window (top right corner) gives you more information about:

- what is displayed on the plot
- how the algorithm works
- the meaning of the parameters
- the graphical interaction possibilities.

The **Delay Estimation** algorithm plots a non-parametric estimate of the impulse response. When there is a delay, the impulse response is zero up to a certain time instance. Since the data sample we have at our disposal is finite, the impulse response is never exactly equal to zero (due to noise effects). The horizontal (blue) lines indicate the confidence level, below which the impulse response is virtually zero.

The vertical (red) lines indicate the estimated delay. Clicking near one of these lines and dragging it from left to right allows you to adjust the estimated delay. When you release the left mouse button, the "DELAY" window (Figure 2.14) pops up. Note that hitting the push button in the parameter area (labelled "Push

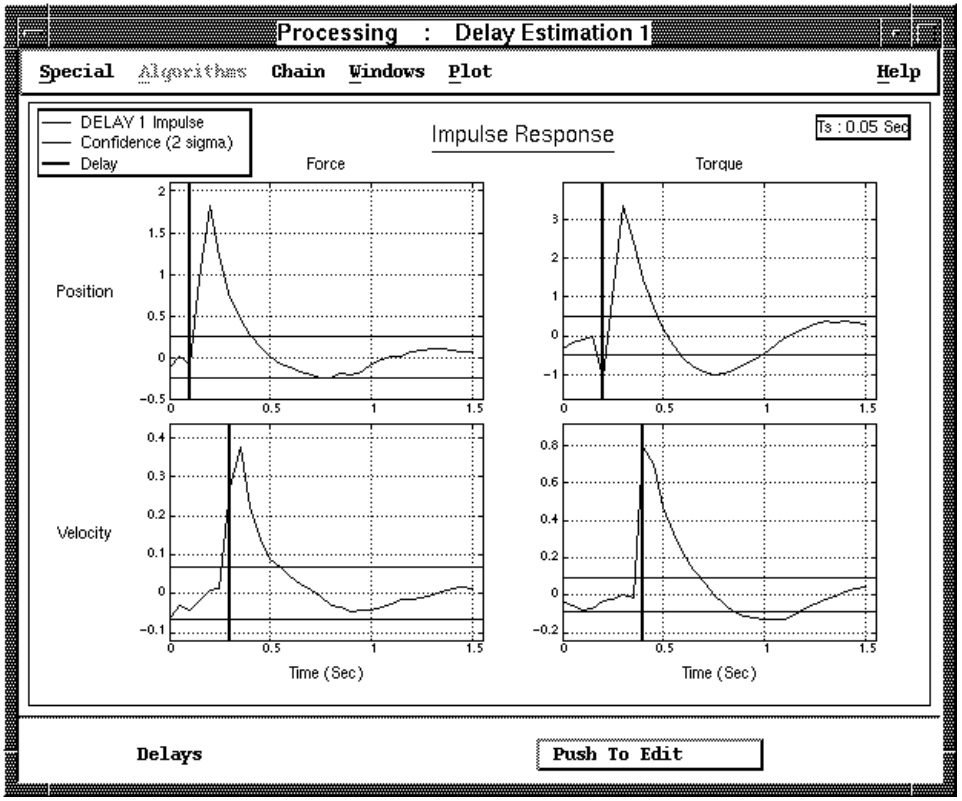


Figure 2.13 The **Delay Estimation** algorithm window.

To Edit") would also have resulted in the "DELAY" window. Enter all delays (graphically or numerically) then hit the OK button.

It is always better to underestimate delays then to overestimate them, since over estimation can lead to a non-causal system. Set the delays equal to `[0.1,0.2; 0.25,0.35]` and hit the OK button. The inputs and outputs are shifted to compensate for the delays. The shifted signals are available at the output of the **Delay Estimation** algorithm.

7. In this step, you will split the input-output data record into two at the **Delay Estimation** output. Use the menu entry Algorithms → Processing → View/Split to place and open the algorithm with the correct input connections (see Figure 2.11). The intermediate Chain Plot is depicted in Figure 2.15.

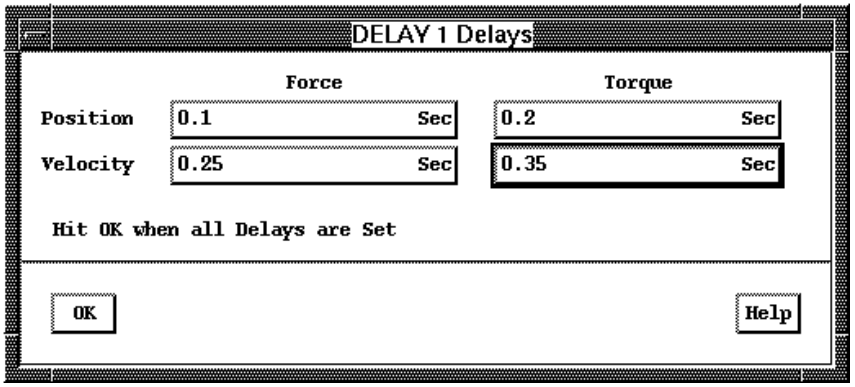


Figure 2.14 The "DELAY" window allows you to enter the delays numerically.

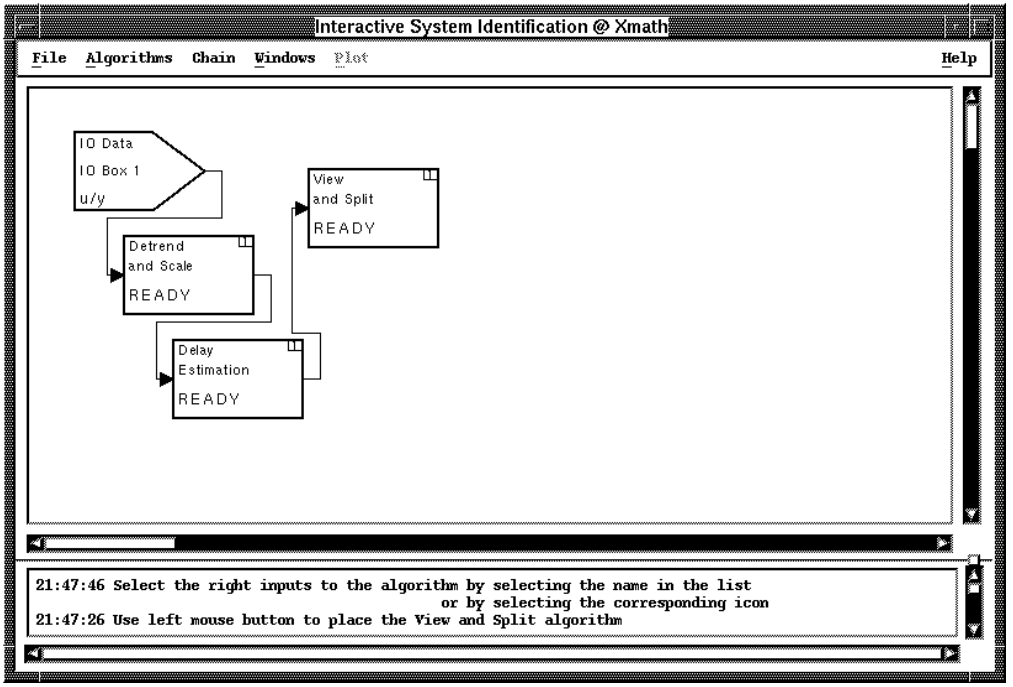


Figure 2.15 Intermediate Chain Plot after processing of the input-output data.

8. It is time to open your first identification algorithm. Use the menu entry Algorithms → Identification → Subspace Identification to start the algorithm. Make sure you connect the input of the identification algorithm to the data object "SPLIT 1 ID Out" (and not "SPLIT 1 VAL Out") since you want the identification output to be redirected to the identification algorithm. Connect the inputs, then hit OK in the "OPEN" window. This pops up the **Subspace Identification** window (Figure 2.16).

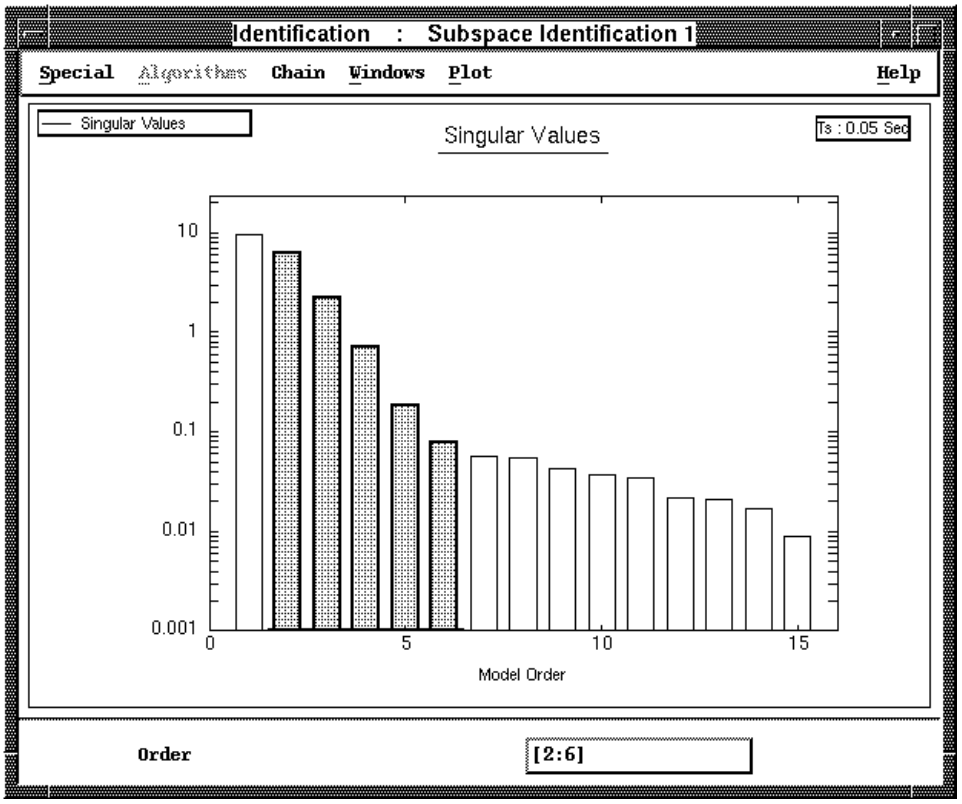


Figure 2.16 The **Subspace Identification** algorithm window.

9. The next step is to determine the order of the system by looking at the last significant singular value on the plot. The order seems to be around five, but this is not very clear from the plot. An easy way to determine a reasonable order is as follows. Open the **Error Norms** window from the menu entry Algorithms → Validation → Error Norms. Connect the validation data from the **View/Split** algorithm ("SPLIT 1 VAL Out") and the model output of the **Subspace Identification**

algorithm ("SDS 1 Model") to the **Error Norms** algorithm. After opening the **Error Norms** algorithm, return to the **Subspace Identification** window and select model order 2 to 6 (click and drag with the left mouse). When the chain returns from execution, the error norms plot is created (Figure 2.17).

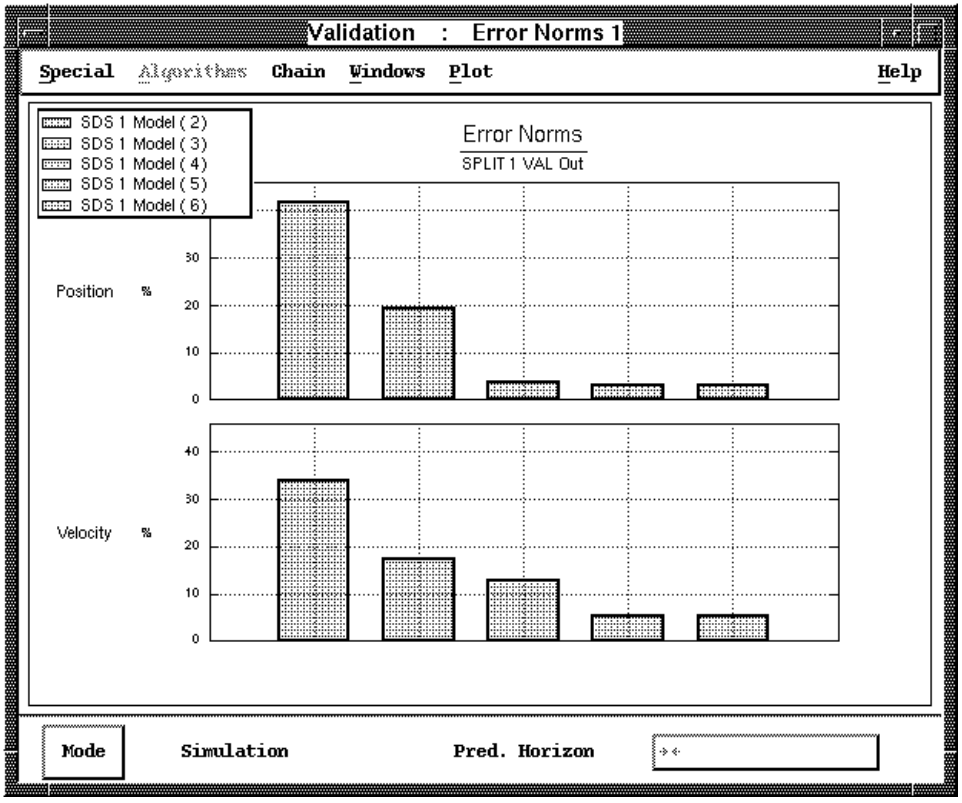


Figure 2.17 The **Error Norms** algorithm window.

It is clear that a fifth order model is the best. This is because from model order 6 on, the validation error is increasing again, which indicates over-parameterization. You can use data viewing to inspect the value of the error norm bars: Click with the right mouse button on one of the bars to display the height of the bar. See Section 3.7.2 for more details. Note that the above strategy (looking at the error norms on the validation data set) can be used for other identification algorithms as well.

10. Now that you have established that the fifth order model is the best, select order 5 in the **Subspace Identification** algorithm window. Inspect the frequency

response (Algorithms → Display → Frequency Response) and the prediction error (Algorithms → Validation → Prediction Errors). The connections can be derived form Figure 2.11. Remember that you can always move the algorithm icons in the Chain Plot by clicking and dragging with the left mouse button.

11. Finally open a second identification algorithm by selecting the menu entry Algorithms → Identification → Least Squares. Connect the **Least Square Identification** algorithm to the identification output of the **View/Split** algorithm. Select the fifth through ninth orders, or type `[5:9]` in the order input box. It is very useful to inspect the error norms of the least squares model on the same plot as the error norms of the subspace identification model. This can be easily done as follows (see also Section 3.3.3 on making and deleting connections):

    - Select the **Least Squares** icon. The algorithm icon is highlighted.
    - Click on the **Error Norms** icon.
    - The "CONNECT" window in Figure 2.18 pops up.

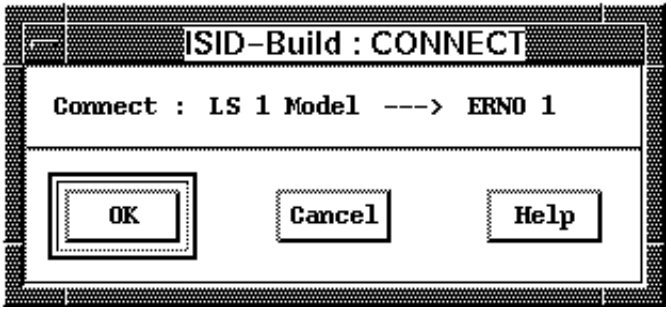


Figure 2.18 The "CONNECT" window allows you to make new connections.

    - Hit the OK button of the "CONNECT" window to realize the highlighted connection and to plot the error norms of the identified least squares models on top of the error norm of the identified subspace model. The resulting **Error Norms** algorithm window is shown in Figure 2.19.

From studying the error norms (Figure 2.19) and examining the models with data viewing, two things become clear:

- The seventh order model is the best least squares model.
- This model is about as accurate as the previously identified fifth order subspace identification model.

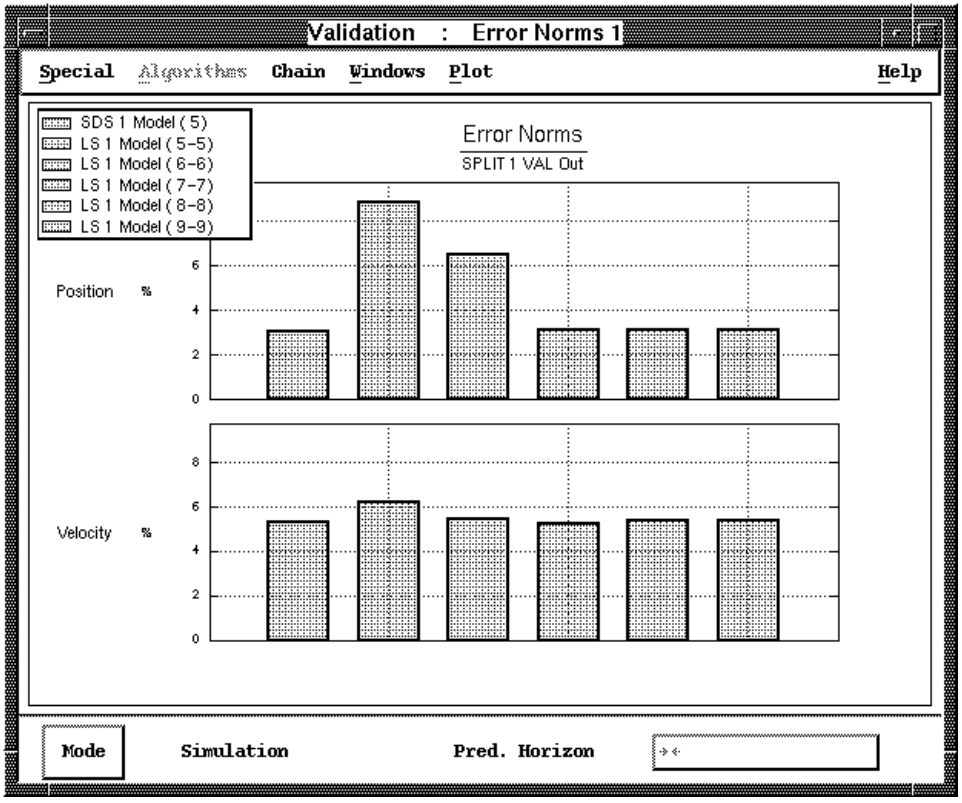


Figure 2.19 The algorithm window associated with the **Error Norms** algorithm. Both the error norm of the fifth order subspace model and the error norms of the least squares models are plotted.

12. Return to the **Least Squares** algorithm window and select model order 7. As a last step, connect the output of the **Least Squares** icon to the **Frequency Response** icon and the **Prediction Error** icon. This results in the chain shown in Figure 2.11.

    It is worth noting that the menu entries Windows → Iconify All (`<Ctrl> - a`) and Windows → Reorder All (`<Ctrl> - r`) can be very useful in keeping track of all the windows. See also Section 3.8 for more information.

13. Now that you have built this chain, it is instructive to check the effect of the estimated delay on the final result. This process illustrates some chain execution features.

One way to check the effect of a different delay is to go back to the **Delay Estimation** algorithm window and adjust the estimated delay. If you do this, all algorithms after the **Delay Estimation** algorithm are automatically re-executed. The execution can be stopped at any time by hitting `<Ctrl> - c` in any ISID windows, or selecting Chain → Stop. Stopping the chain execution is very useful if you accidentally start a chain and you don't want to wait until the chain has finished calculating.

Selecting Chain → Automatic → Off is a more controlled manner of executing the chain (see also section 3.4.1). Parameter changes are not automatically executed when the automatic execution is turned off. You must start the chain manually using Chain → Start (`<Ctrl> - e`) or Chain → Step (`<Ctrl> - s`). This allows you to change different parameters before re-executing the chain. Chain → Step also gives you control over the order in which the algorithms are executed: the selected algorithm is executed first (see section 3.4.1). After selecting Chain → Automatic → Off, you can change the time delays (for instance to `[0.1,0.2;0.3,0.4]`) and step through the chain (use `<Ctrl> - s`) to see the effect of the changes. This experiment is up to you.

14. You can save any of the outputs of the algorithms (the models or the frequency response for instance) by selecting File → Save Output To Xmath. This works exactly as explained in Section 2.1.

15. To save the whole chain for later use, select File → Save Chain To Xmath. The chain can be re-loaded with File → Load Chain From Xmath. More details about saving and loading chains can be found in section 3.2.3 and 3.2.4.

ISID has many more options and possibilities than described in this example. Only the basic options have been explained. Additional options can be discovered by experimenting, browsing through the menu entries, or reading Chapter 3, which also contains a quick reference section.

Chapter 3

Detailed Description

This chapter contains a detailed description of all ISID functionalities. If you read it as a whole you will be acquainted with everything ISID can do. Alternatively, you can use this chapter as a reference when you want to know something more about a specific part or functionality of ISID.
All information contained in this chapter can also be found in the on-line helps. The organization of the sections in this chapter roughly corresponds to the organization of the help entries in the Main window. Section 3.1 is a quick reference section for acquiring more details about a specific menu entry, accelerator or graphical interaction. Section 3.2 specifies the details of loading and saving data objects and chains. The following sections describe how to build a chain (Section 3.3), how to execute a chain (Section 3.4) and how to interact with the Chain Plot (Section 3.5). Section 3.6 describes how to manipulate algorithm parameters. Section 3.7 covers the general plotting features of the algorithm windows. ISID sometimes creates many windows. The window handling problem is dealt with in Section 3.8. Finally, the possibilities of customizing your ISID session are explained in Section 3.9.

3.1 Quick Reference

This section provides a fast, easy way to access ISID. All menu entries, accelerators and graphical interaction methods are listed. The section numbers and pages where more information about a feature can be obtained are also listed.

Menu Entries

File	Section	Page
→ Load Data From Xmath	3.2.1	44
→ Load Default Data	3.2.1	44
→ Load Chain From Xmath	3.2.3	50
→ Save Chain To Xmath	3.2.4	50
→ Load Default Chain	3.2.3	50
→ Hardcopy Chain Plot	3.7.4	68
→ Exit	-	-

Special	Section	Page
→ Defaults	3.6.3	64
→ Disable	3.5.1	61
→ Save Output to Xmath	3.2.2	47
→ Hardcopy Plot	3.7.4	68
→ Close Window	3.3.2	57

Chain	Section	Page
→ Automatic	3.4.1	59
→ Start	3.4.1	59
→ Step	3.4.1	59
→ Stop	3.4.1	59
→ Clear	3.4.1	59

Plot	Section	Page
→ X-Axis	3.7.3	68
→ Y-Axis	3.7.3	68
→ Bar Graph	3.7.3	68
→ Grid	3.7.3	68
→ Scale	3.7.1	65
→ Text	3.7.3	68

Algorithms	Section	Page
→ Processing	3.3.1	51
→ Identification	3.3.1	51
→ Validation	3.3.1	51
→ Display	3.3.1	51

Windows	Section	Page
→ Chain	3.8.3	69
→ Iconify All	3.8.1	69
→ Reorder All	3.8.2	69

Graphical Interaction

Chain Plot interaction	Section 3.5 on page 61
Left click on icon	select/deselect/connect
Left double click on icon Left <Shift> click on icon	deiconify/raise
Left <Ctrl> click on icon	close
Left <Alt> click on icon	disable/enable
Left drag on icon	move icon
Middle click on icon	deiconify/raise disable/enable/close help/information
Left <Ctrl> click on connection	delete connection
Middle click on connection	delete connection information on connection
Left click background	deselect all/place algorithm
Left <Ctrl> click background	cancel placing
Middle click background	automatic execution on/off start/step/stop chain
Right click background	iconify/reorder/deselect all

Algorithm Plot interaction	Section 3.6.2 on page 64
Left click on data/background/legend	algorithm specific
Left <Ctrl> click in subplot	zoom subplot/show original data
Left <Ctrl> drag in subplot	lasso zoom
Left <Ctrl> click/drag on axis	re-range axis (inward)
Left <Ctrl> click outside axis	re-range axis (outward)
Plot → Scale → Automatic	re-range axis automatic
Left <Ctrl> click on title	toggle original data/zoomed data
Middle click/drag	hourglass
Middle <Ctrl> click/drag	bigger hourglass
Middle <Shift> click/drag	hourglass/larger magnification
Middle <Ctrl> and <Shift> click/drag	bigger hourglass/larger magnification
Right click/drag on data	data viewing

Accelerators

<Ctrl> - e	Start chain execution
<Ctrl> - s	Step chain execution
<Ctrl> - c	Stop chain execution
<Ctrl> - w	Show chain window
<Ctrl> - a	Iconify all windows
<Ctrl> - r	Reorder all windows

3.2 Loading and Saving

ISID and Xmath interact when loading and saving data objects and chains. Typically, the measured (or simulated) data is available in a file before you start an identification session. The first step is to load data from a file into Xmath using the Xmath `load` command or `read()` function. Appendix A describes in detail how to load your data from a file. Once the data is available as an Xmath variable, it can be loaded into ISID. This is explained in Section 3.2.1. At the end of an identification session, an identified model can be saved as an Xmath variable. Section 3.2.2 describes how to do this. Moreover, whole chains can be saved to Xmath and loaded back into ISID. This is very useful when you want to complete work on a certain chain another time or when you want to save the chain for later use (when new measurements become available, for instance) or to exchange information between users. Loading and saving chains is discussed in Sections 3.2.3 and 3.2.4.

3.2.1 Loading Data Objects

Once the data exists as an Xmath variable, it can be loaded into ISID using File → Load Data From Xmath. Since there are five different types of data objects (see Section 1.2.1), there are five possible types of data that can be loaded. The Xmath variables that can be loaded as one of these data objects are:

- **IO Data Record:** a real matrix or a real parameter dependent matrix (PDM), with minimally 10 elements in the domain. For the PDM, at least one of the sizes must be equal to one. For more information about PDMs, refer to the Xmath Basics manual.
- **Frequency Response:** a complex PDM with at least 10 elements in the domain.
- **Impulse Response:** a real PDM with at least 10 elements in the domain.
- **Model:** a discrete time Xmath system with at least one state.

- **Square Root:** a Least Squares square root object (see the ISID Part 1 manual).

For example selecting File → Load Data From Xmath → IO Data Record pops up the "LOAD VARIABLE" window in Figure 3.1. You can now select the Xmath variable you want to use as input and output by selecting the variable name in one of the lists[1]. The Edit button at the bottom of the screen allows you to edit the channel names and units and the sampling time and unit. When the selected Xmath variable is a PDM, this information is automatically extracted from its column names, domain name and domain.
It is also possible to select "No Variable". This implies you will be opening an EMPTY data box with no variable connected to it (see below). Note that when no appropriate variables are found in Xmath, the only choice in the list is the "No Variable" entry.

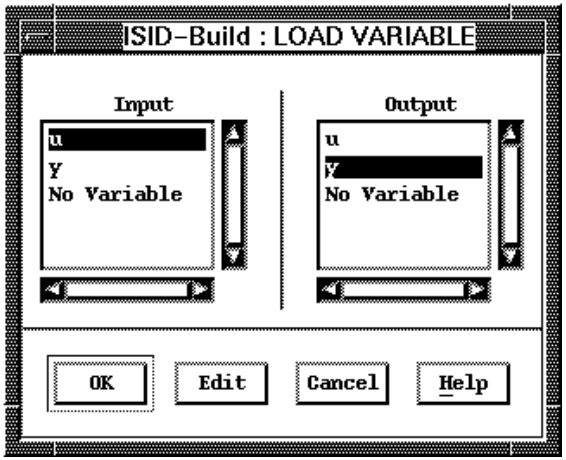


Figure 3.1 The "LOAD VARIABLE" window. You can select any of the Xmath variables to form an input-output data object. The Edit button allows you to edit the channel names and units and the sample time and unit.

After you have selected the right variables to be loaded and have filled out the correct channel names (units) and sampling time (unit), hit the OK button in the "LOAD VARIABLE" window. At that moment, the Xmath variables become an ISID data object. After hitting OK, one of the following two things happens:

- When there is no data box of the type you loaded open yet, a data box is generated that contains the newly loaded data object. This means that a data box icon is plotted in the Chain Plot window and that a corresponding window displaying

[1] Only the variables that reside in the `main` partition are displayed.

the loaded data is opened. When "No Variable" is selected, an EMPTY data box icon is created.

- When there is already a data box of the type you are loading present in the chain, the "LOAD OPTIONS" window pops up (see Figure 3.2). This window allows you to either redirect the loaded data to one of the existing data boxes or open a new data box.

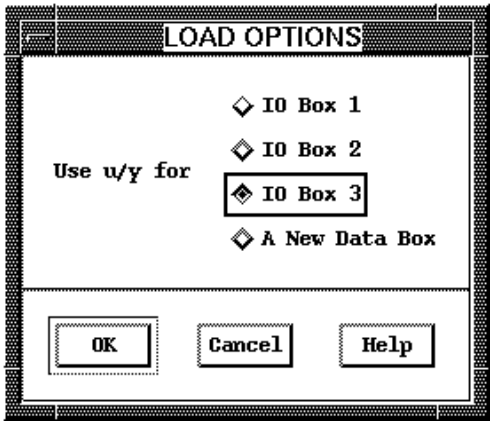


Figure 3.2 The "LOAD OPTIONS" window. You can either redirect the loaded variable to an existing data box or open a new data box with the variable.

The menu entry File → Load Default Data enables you to load any of the default data sets. These data sets are used for the case studies described in Chapter 5.

Remarks:

- Empty data boxes can be very useful when you want to construct an empty chain. Building empty chains is typically done when one knows in advance which algorithms to put in the chain and how to connect them. The chain can then be built very fast, avoiding repeated chain execution every time a new algorithm is opened. After building the chain, you can load an Xmath variable and redirect it to the empty data box at the start of the chain. This is the way the default chains work (see Section 3.2.3).

- Loading the other four types of data objects is very similar. The only difference is that there is only one list with Xmath variables in the "LOAD VARIABLE" window.

- For a further description of the parameters, plots, and data box options see Chapter 4.

3.2.2 Saving Data Objects to Xmath

Just as the identification process begins with loading data, it ends with saving. The output(s) of each of the building blocks (algorithms and data boxes) can be saved back to Xmath. When you save an ISID data object, it is transformed into an Xmath variable. This variable can then be further used in Xmath. For a definition of the outputs of each of the separate building blocks, we refer to Chapter 4, where all building blocks are described in detail (inputs, outputs and parameters).
To save the output of a building block, use the menu entry Special → Save Output to Xmath of the corresponding building block. The "SAVE" window pops up on top of the algorithm window (Figure 3.3). The top part of this window consists of three parts:

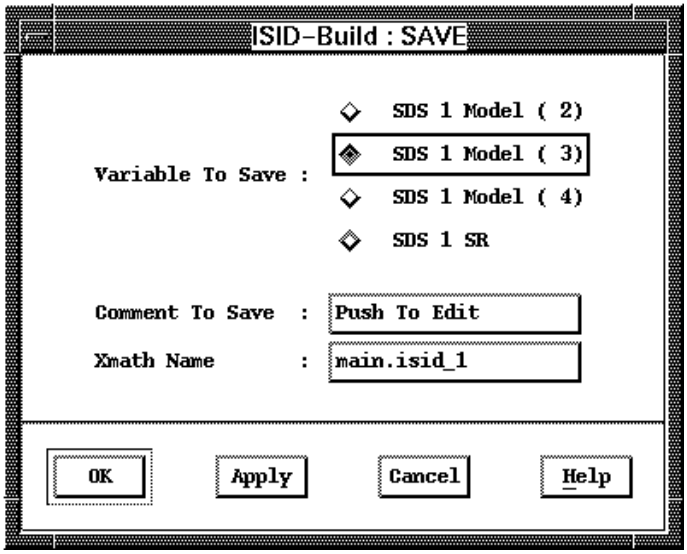


Figure 3.3 The "SAVE" window allows you to save ISID data objects as Xmath variables. You can select the data object to save, the comment to save with it and specify the Xmath variable name.

- The output data object you wish to save as an Xmath variable. When there is more than one data object that could be saved, you must select one by clicking a diamond shaped box.

- The comment to save. This allows you to save a comment with the variable. This comment can be retrieved from the Xmath variable with the function `commentof`. For instance, when you save the object in an Xmath variable `isid_1`, you can retrieve the comment by typing:

  ```
  commentof(isid_1)
  ```

  in the Xmath command area. This is a very simple but useful utility that allows you to keep track of how a certain model was obtained. You can enter your comment by hitting the button that reads: "Push To Edit".

  When you push this button, the "HISTORY" window pops up (Figure 3.4). The data object's history becomes the default comment saved with the variable. The history is a textual description of how the data object came about (a description of the chain). It describes all building blocks that were relevant when generating the data object. This allows you to find the exact settings of the chain at the moment you have saved the data object. This ability is very useful for bookkeeping.

  You can edit the text in the "HISTORY" window by clicking with the left mouse in the text area and typing in the required text. In this way you can add extra comments to the default history (or you can replace it totally if so desired). You can always go back to the original history by hitting the Reset button in the "HISTORY" window.

- The third part of the "SAVE" window is the Xmath variable name that will contain the object.

After entering a comment and an Xmath variable name hit the Apply or the OK button in the "SAVE" window (Figure 3.3). The only difference between the two buttons is that the Apply button does not destroy the "SAVE" window and thus enables you to save another variable. This is useful when you want to save more than one output.

Remarks:

- Most of the identification algorithms allow you to select more than one order. The model data object at the output of these algorithms thus contains more than one model. You can not save all models at once though. The Variable To Save field in the "SAVE" window requires you to select a single model to save.

- IO Data records and models are saved in the following way:

  - **Data Record:** When the Xmath Name is `var`, then the input data is saved in the variable `var_u` and the output variable is saved in `var_y`.

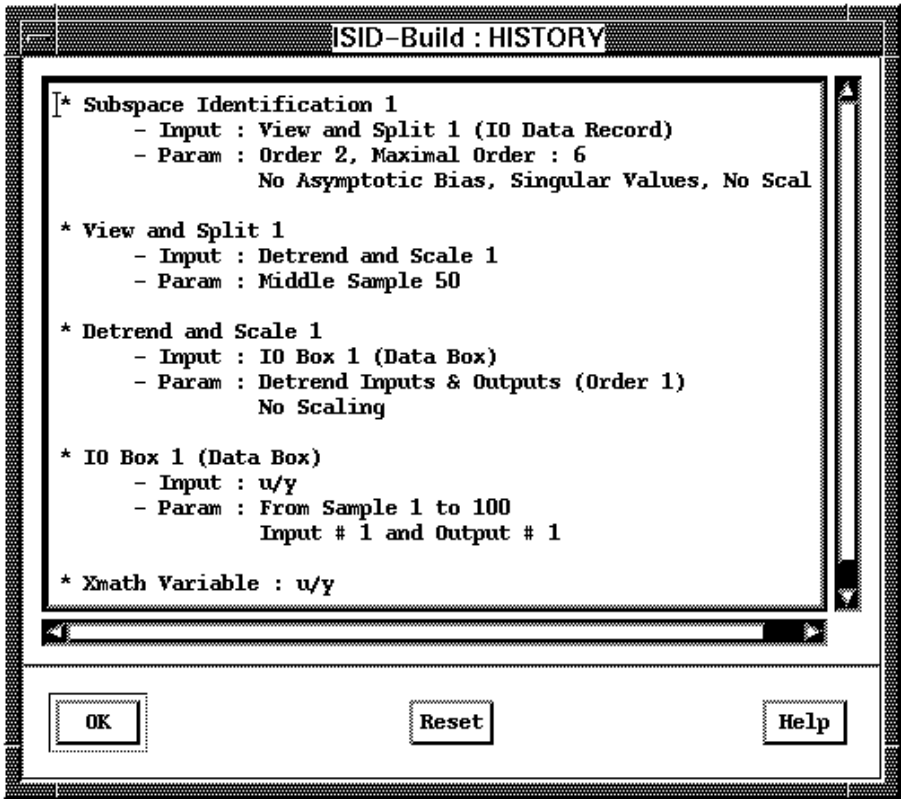


Figure 3.4 The "HISTORY" window contains the history of the object that will be saved. It also allows you to enter your own comment.

– **Model:** When the Xmath Name is `var`, then the input-output part of the model is saved in the variable `var` and the innovations model is saved in the variable `var_inn`. This is similar to the `inn` keyword that can be used in many of the ISID functions. For more information about input-output and innovations models refer to the ISID Part 1 manual.

3.2.3 Loading Chains

ISID allows you to load chains. There are two cases:

1. A chain you have saved yourself (see Section 3.2.4) can be loaded with the menu entry Load → Load Chain From Xmath. A dialog pop up will ask you for the name of the Xmath variable you saved the chain in. This allows you to continue working with chains that you made before, or to exchange chains between different people. Typically after you have loaded a chain from Xmath, you should load an Xmath variable in ISID (use File → Load Data From Xmath). This variable can then be easily redirected to the data box at the beginning of the empty chain.

2. The default chains are very useful when you want to get fast results. In principle they return reasonable models in 90% of the identification problems. Figure 3.5 shows the "DEFAULT CHAIN" pop up resulting from File → Load Default Chain. The "DEFAULT CHAIN" window consists of two areas:

   - The top three chains are simple default chains that can be used to get fast results.
   - The bottom chains are more complicated chains that are used in the case studies in Chapter 5.

   The following three Figures review the first three default chains.

   - **Least Squares Identification:** Figure 3.6.
   - **Empirical Identification:** Figure 3.7.
   - **Subspace Identification:** Figure 3.8.

   After loading any of these default chains you should also load data from Xmath (use File → Load Data From Xmath). This data can then be easily redirected to the data box at the beginning of the chain.

Beware that loading a new chain will destroy the existing chain. That is why ISID asks for confirmation to delete the chain before loading any of the default chains. If you want to save your work, first use File → Save Chain To Xmath.

3.2.4 Saving Chains to Xmath

To save the chain you have build in ISID, use the menu entry File → Save Chain To Xmath. A dialog prompt will ask you for the name of the variable you want to save the chain in. The whole ISID setup (connections, parameters and icon positions) will be saved in one Xmath variable. This Xmath variable can then be saved in a file. The chain can be re-loaded into ISID using File → Load Chain From Xmath (see Section 3.2.3).

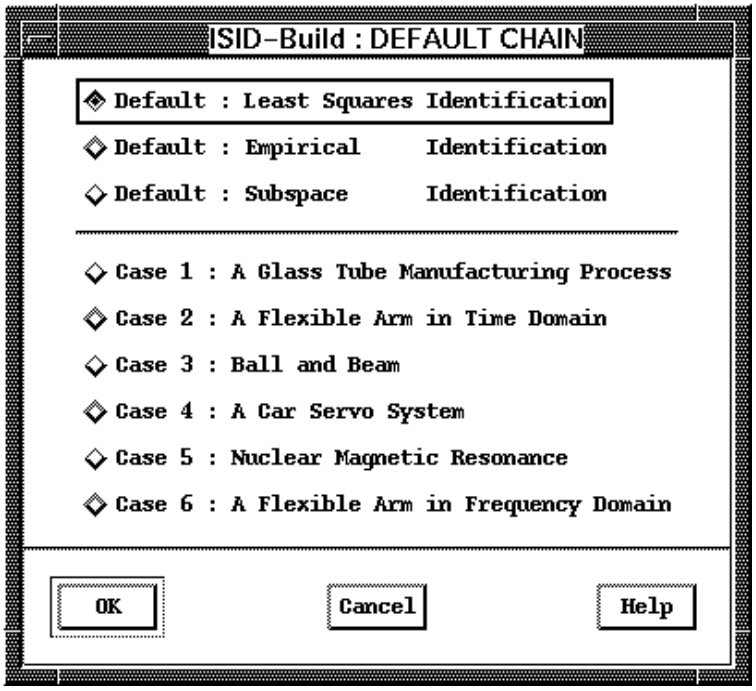


Figure 3.5 The "DEFAULT CHAIN" window. The top three chains are the chains that are preprogrammed to deliver good results in 90% of the identification problems. The bottom chains are used for the case studies in Chapter 5.

3.3 Building a Chain

Building chains is done through opening building blocks one by one. Data boxes are automatically opened when data is read into ISID with File → Load Data From Xmath. In this section we discuss how algorithms can be opened and closed. Finally we discuss how to make and delete connections.

3.3.1 Opening an Algorithm Building Block

There are 4 types of algorithm building blocks: Processing, Identification, Validation and Display. More information about the exact inputs, outputs and parameters of these blocks can be found in Chapter 4. Here we review how to open any of these algorithms. All algorithm building blocks can be found under one of the following menu entries:

- **Algorithms → Processing:** process input-output data records.

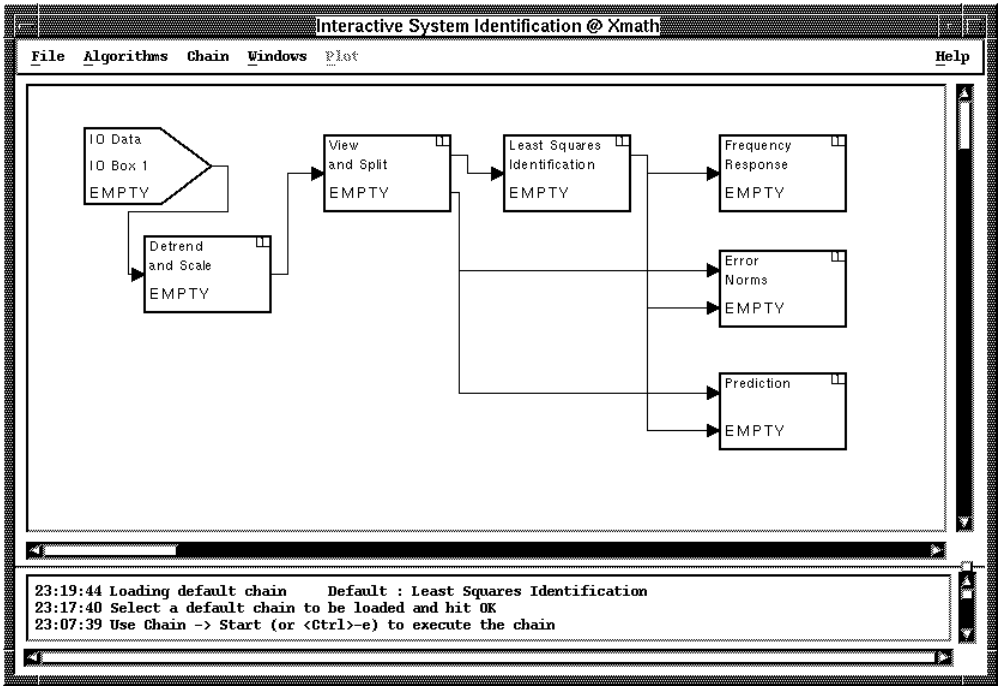


Figure 3.6 The Least Squares Identification default chain identifies a least squares model from one input-output data record. The data is detrended and then split into an identification and a validation data set. A least squares model is identified from the identification data set. The frequency response and least squares error norms of the identified model are plotted. From the error norm plot, the best model order can be determined. The prediction using the least squares model is also plotted.

- **Algorithms → Identification:** identify models.
- **Algorithms → Validation:** validate models.
- **Algorithms → Display:** display model properties.

Select one of the algorithm building blocks from the Algorithms menu. The cursor changes from an arrow to a cross. A dotted rectangular box appears in the Chain Plot (the center of the box is the cross). To place the algorithm icon click somewhere in the Chain Plot. The algorithm icon will be placed exactly at that place. If you make a mistake (you selected the wrong algorithm for instance), you can cancel by holding down the `<Ctrl>` key and clicking in the background of the Chain Plot.
After you have positioned the algorithm icon, the "OPEN" window will pop up (Figure 3.9). This window has different functions:

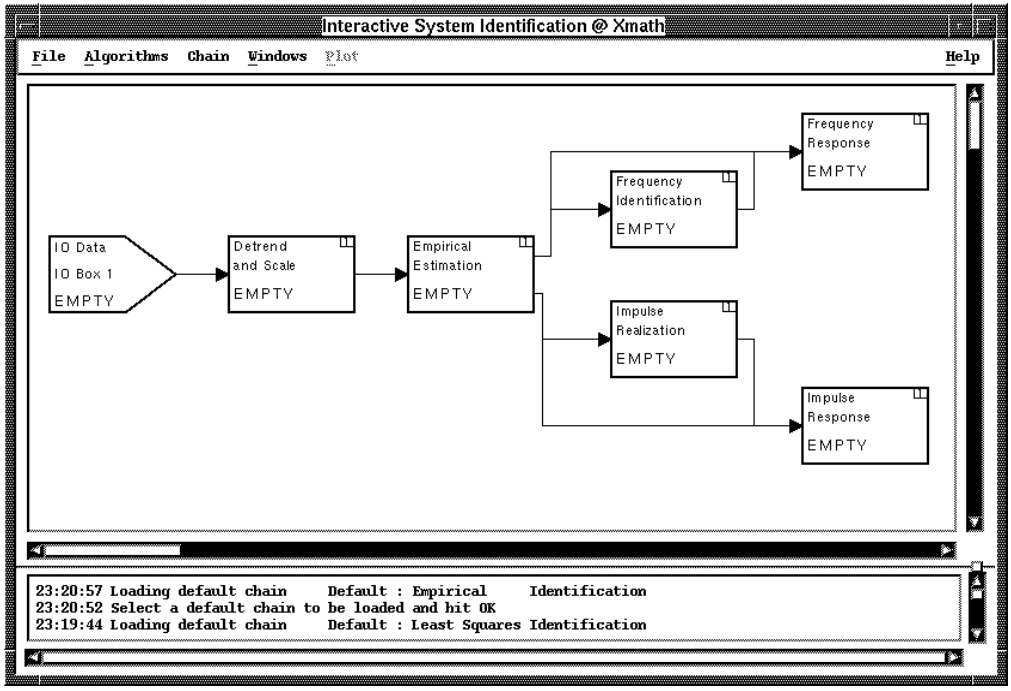


Figure 3.7 The Empirical Identification default chain identifies a non-parametric impulse and frequency response from one detrended input-output data record. The resulting frequency response is modeled in the frequency domain, while the impulse response is modeled in the time domain.

- Determine the data objects at the input of the algorithm. The list(s) in the "OPEN" window display all possible data objects that can be connected to the input(s) of the algorithm that is opened. Click to select one (or click and drag to select more, if the algorithm allows for multiple inputs) of these objects. The Chain Plot displays the connections you have selected.

  Apart from selecting a data object name from the list(s) to establish connections, it is also possible to select an icon whose output will serve as input to the algorithm. If the output of the selected algorithm is compatible with any of the inputs of the algorithm you are opening, this output will be connected to the input of the algorithm.

  Finally, it is possible to select the "No Connection" entry in the list (alternatively, click in the background) to open an algorithm without connections. The connections can be established later as described in Section 3.3.3.

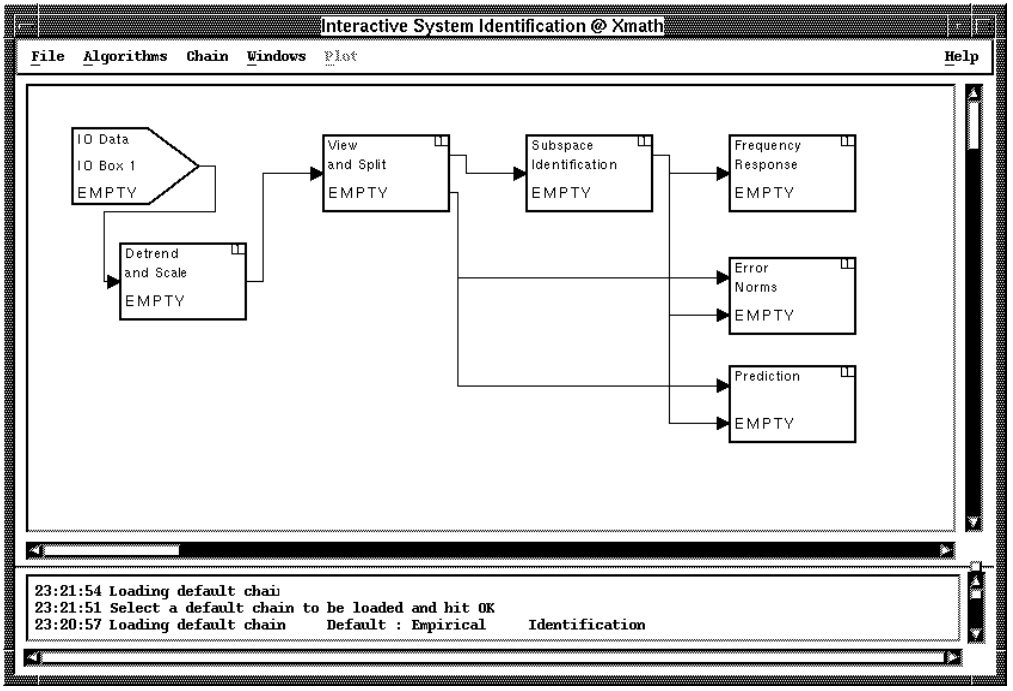


Figure 3.8 The Subspace Identification default chain identifies an innovation model from one input-output data record. The data is detrended and then split into an identification and a validation data set. A subspace identification model is identified from the identification data set. The frequency response and simulation error norm of the model are plotted. The best order can be determined by selecting different model orders and looking at the error norm plot. The prediction using the subspace model is also plotted.

- The "OPEN" window also allows you to set the parameters of the algorithm you are opening. All default parameters are chosen sensibly, and, in general should not be changed. However, if you do want to change them, hit the Default button in the "OPEN" window. This pops up the "DEFAULT" window that contains all parameters of the algorithm (Figure 3.10). The parameters can now be adjusted (see also Section 3.6).

After all inputs are defined (and possibly some parameters are altered), hit the OK button in the "OPEN" window. This will start calculations and open up the corresponding algorithm window (unless the opened algorithm is empty).

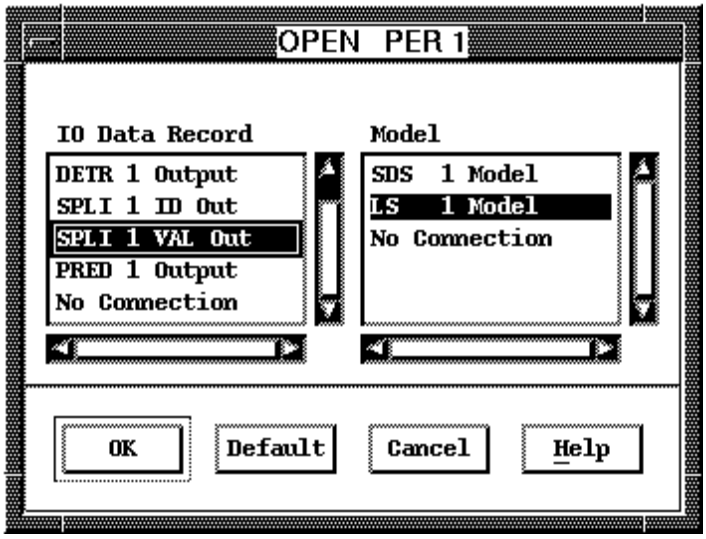


Figure 3.9 The "OPEN" window allows you to select the inputs to the algorithm and adjust the algorithm parameters.

Remarks:

- Some algorithms allow for more than one data record at one of the inputs (for instance the algorithm **Error Norms** allows for one or more IO data records and one or more models). These multiple data objects can easily be selected from the "OPEN" window list by clicking and dragging with the left mouse in the list. If you would want to select two data objects that are not next to each you should select the first one, and then select the second one while holding down the `<Ctrl>` key. Note also that it is not possible to select more than one input by clicking on more than one icon in the Chain Plot.
- When opening an algorithm, there are always some default inputs selected. These defaults are determined as follows (it is useful to know this, since it allows you to influence the default selection):
  - If the last algorithm selected before the algorithm was opened has an output compatible with the input of the opened algorithm, then the selected algorithm is taken as the default input.
  - Else if the last algorithm opened has a compatible output then it becomes the default.
  - Else if the algorithm that is closest to the icon of the newly opened algorithm has a compatible output then this one is taken.

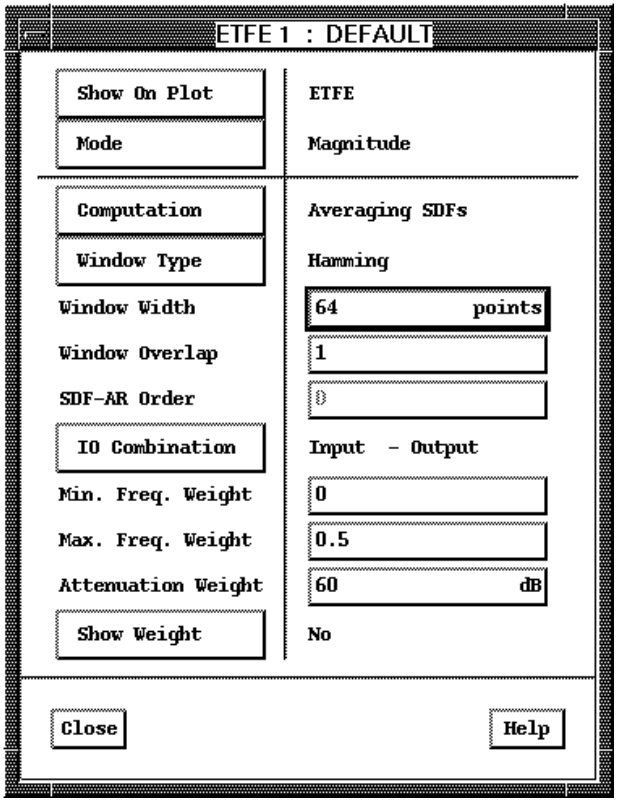


Figure 3.10 The "DEFAULT" window enables you to set the parameters before opening the algorithm. As can be seen from this **Empirical Estimation** "DEFAULT" window, it would be an extensive job finding all the parameters yourself.

  - Else ISID selects a compatible output as default input.

- You can easily open more than one instance of the same algorithm (the number of identical algorithms is restricted to 9 per algorithm). The instance number (displayed in the top right corner of the icon) uniquely determines the algorithm. The names of the outputs of an algorithm also contain the instance number of the algorithm. In that way the naming of objects is unique, and there is no confusion possible. See also Chapter 4 for the naming conventions of the outputs of an algorithm.

3.3.2 Closing a Building Block

All building blocks can be closed (removed from the chain) very easily by selecting Special → Close Window. When the building block is removed, all related data objects will also be deleted. This implies that the outputs of the block are destroyed, and if there were other algorithms that used these outputs as input objects, these dependent algorithms will automatically become empty.
Finally note that an algorithm can also be closed by hitting `<Ctrl>` left mouse on the corresponding icon or by using the middle mouse pop up menu (see Section 3.5.1). This is easier when closing more than one algorithm.

3.3.3 Making/Deleting Connections

Even though the inputs to an algorithm are generally determined while opening it (through the "OPEN" window of Figure 3.9), it is often useful to make and delete connections after an algorithm was opened. This is done as follows:

- To make a new connection between (say) algorithm A and B, select algorithm A and then select algorithm B (note that you can deselect the selected algorithm by clicking in the background of the Chain Plot). After you have selected algorithm B the "CONNECT" window will pop up (Figure 3.11). This window describes the connection you have just selected. The selected connection is also highlighted in the Chain Plot for clarity.

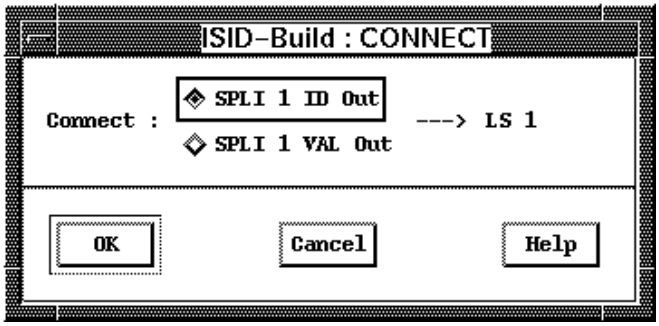


Figure 3.11 The ""CONNECT" window enables you to make connections after you have opened an algorithm.

Hitting the OK button in the "CONNECT" window will realize the highlighted connection. However, it should be noted that an existing input connection to algorithm B will be deleted if algorithm B can only take *one* data object as an input (which is the case for all processing and identification algorithms). Thus, realizing new connections can automatically destroy existing connections. When

algorithm B can take more than one input (as the validation and display algorithms do), then the new connection is added to the existing connection(s).

- Deleting a connection is very similar to making a new one. Say you want to delete the connection between algorithm A and B. First select algorithm A, then select algorithm B. This will pop up the "DELETE" window (Figure 3.12).

  Hitting the OK button in the "DELETE" window will delete the highlighted connection. Note that the state of algorithm B can become "EMPTY" when you delete a connection (see also Section 3.4.2).

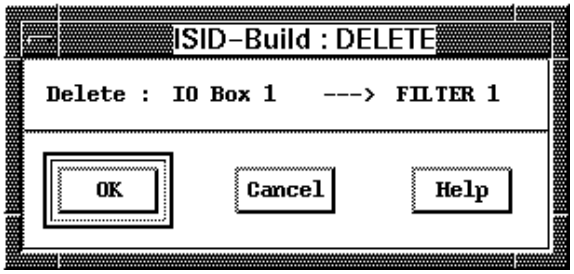


Figure 3.12 The "DELETE" window enables you to delete existing connections.

Remark that connections can also be deleted by hitting `<Ctrl>` left mouse on the connection in the Chain Plot, or by using the middle mouse button pop up (See Section 3.5.1).

3.3.4 Deleting a Chain

When you want to clear the Chain Plot and build a new chain, select Chain → Clear. After confirmation, the whole chain and all corresponding windows will be destroyed. It is not possible to recover the deleted information, unless you have saved the chain before deleting it (use File → Save Chain To Xmath).

3.4 Executing a Chain

The chain consists of algorithms that are connected back to front. When an input or a parameter of an algorithm changes, the algorithm is executed to calculate its new output. When this output serves as an input to a second algorithm, the connected algorithm will be executed too. Thus, changing a parameter in an algorithm, triggers different algorithms in the chain to be executed one after another. This is called "Automatic Execution".

At startup, the chain execution is set to automatic. In some cases, this is not the desired behavior. For instance, when you want to change more than one parameter and *then* study the effect of these parameter changes on the chain. It is possible to alter the execution mode of ISID so that you get more control over the time of execution and the order in which the algorithms are executed.

3.4.1 Manipulating Execution

Selecting Chain → Automatic → Off, turns off automatic chain execution. The effect is as follows:

- After an algorithm input or parameter changed, the chain does not re-execute automatically. Instead, the algorithm that has changed will alter its state from "READY" to "CHANGED" (see also Section 3.4.2). This implies that the output object of the algorithm does not correspond to the current input objects and parameter settings.
- To start the execution of the chain, use the menu entry Chain → Start (or alternatively hit `<Ctrl> - e`). At this moment, all algorithms with changed inputs or parameters will execute one by one (as in regular automatic execution mode). This enables you to change different parameters without having the chain re-executing every time you change one.
- You can also step through the chain with Chain → Step (or alternatively `<Ctrl> - s`). This will execute the "first" algorithm in the chain that has changed inputs or parameters. "First" roughly means "the algorithm that is the closest to the start of the chain (to the data boxes)". It is also possible to manipulate which algorithm executes first. Just select the specific algorithm (single left click on the icon), then select Chain → Step (or `<Ctrl> - s`). This will execute the selected algorithm.

Automatic chain execution can be reactivated by selecting Chain → Automatic → On. At this moment all remaining changed algorithms will be executed automatically.

Disabling specific algorithms is another way to manipulate chain execution. An algorithm that is disabled will not react to input or parameter changes. However, its output will stay intact. This is useful (for instance) when the output of an identification algorithm is connected to an error norm algorithm and a frequency response algorithm. For example you may want to see the effect of different model orders on the error norms, without having the frequency responses being redisplayed every time. You can disable the frequency response algorithm using Special → Disable. At that moment, the frequency response algorithm will not react to input (or parameter) changes. After you have selected the right model order, you can enable the frequency response algorithm

again with the same menu entry (which has become Special → Enable). The algorithm will execute if the chain is set to automatic, or its state will change to "CHANGED", when the chain is set to non-automatic.

It is often useful to interrupt the execution of the chain, especially when the execution is set to automatic and you have started the chain. To stop the chain, select Chain → Stop. Note that this menu entry is only sensitive when the chain is executing. Alternatively, you can hit `<Ctrl> - c` at any moment to stop the execution of the chain. Execution can be resumed by selecting Chain → Start (or `<Ctrl> - e`).

Remarks:

- When you open a new algorithm (see Section 3.3.1), it will always be executed, independent of the chain execution mode. Empty or partially connected algorithms are not executed.
- A data box can not be disabled because it is positioned at the beginning of a chain.
- Empty algorithms can not be disabled.
- If you change a parameter that does not alter the output of the algorithm (displaying magnitude/phase in the frequency response algorithm for instance), the algorithm will always be executed immediately, independent of the execution mode.

3.4.2 States of a Building Block

As seen in the previous section, a building block can change its state depending on its current inputs and parameters and the execution settings of the chain. In this section we list the possible states of an algorithm. An algorithm's state is indicated by the bottom text entry in the algorithm icon. On color monitors, the state is also indicated by the color of the icon and the associated algorithm window.

- **EMPTY (normal color):** a data box is empty because you selected "No Variable" when loading the data (see Section 3.2.1) or because you have loaded a chain. An algorithm is empty when not all inputs are connected, or when the inputs themselves come from an empty algorithm. For instance, when loading a chain from Xmath, all data boxes and algorithms are empty.
- **CHANGED (green):** a building block is changed when either an input or parameter has changed, and Chain → Automatic → Off is selected.
- **DISABLED (lavender):** an algorithm is disabled when Special → Disable is selected. A disabled algorithm does not react to input or parameter changes. It can be enabled with Special → Enable. See Section 3.4.1.

- **OPEN (red):** an algorithm is in the "OPEN" state at the moment the connections (and default parameters) are defined. See Section 3.3.1.
- **WORKING (purple):** a building block is working when the underlying functions are executing to generate a new output data object.
- **READY (normal color):** a building block is ready when it is none of the above.

3.5 Interacting with the Chain Plot

There are many possible graphical interactions with the Chain Plot. In this section, we present an exhaustive list of these interactions. Note that, unlike all other plots, the middle and right mouse buttons do not perform zooming and data viewing in the Chain Plot. Note also that in the following sections "`<Shift>` Click" indicates that you should hold down the shift key while pressing the mouse button. In section 3.1 you can find a table containing the Chain Plot interactions. This table serves as a quick reference.

3.5.1 Left Mouse Button

You can use the left mouse button on an icon, on a connection or in the background.

Using the left mouse button on an icon

- **Single Click:** single clicking on an icon selects it. The selected icon is highlighted (by a red line on a color monitor). Selecting a highlighted icon de-selects it.

  Algorithms are selected to make connections at icon opening (Section 3.3.1), to make and delete connections (Section 3.3.3), and to alter the execution order when stepping through a chain (Section 3.4.1).
- **Click Drag:** when you hold down the left mouse button after you have clicked on an icon, you can move the icon around the Chain Plot. This enables you to reorder the icons: move the pointer over an icon and hold down the left mouse button; without releasing the button, move the mouse until the icon is properly positioned; then release the button.
- **Double Click or `<Shift>` Click:** double clicking (or hitting `<Shift>` Click) on an icon de-iconifies/raises the corresponding algorithm window.
- **`<Ctrl>` Click:** close the corresponding building block (after confirmation).
- **`<Alt>` Click:** disable/enable the corresponding algorithm.

Using the left mouse button on a connection

- `<Ctrl>` **Click:** delete the connection (after confirmation).

Using the left mouse button in the background

- **Single Click:** deselect all selected algorithms. Alternatively, when you are opening an algorithm, a single click with the left mouse in the background determines the position of the algorithm icon.
- `<Ctrl>` **Click:** cancel the opening of an algorithm.

3.5.2 Middle Mouse Button

Selecting an object in the Chain Plot with the middle mouse button will realize a pop up menu. This popup allows you to show, enable/disable or close the corresponding building block. To select one of the menu entries, you must use the **right** mouse button. Moreover, you can also view the on-line help for the building block. Finally, the "Information" menu item displays information about the inputs and outputs of the building block (see Figure 3.13). This is very useful when there are many building blocks in the Chain Plot and the inputs and outputs are not clear.

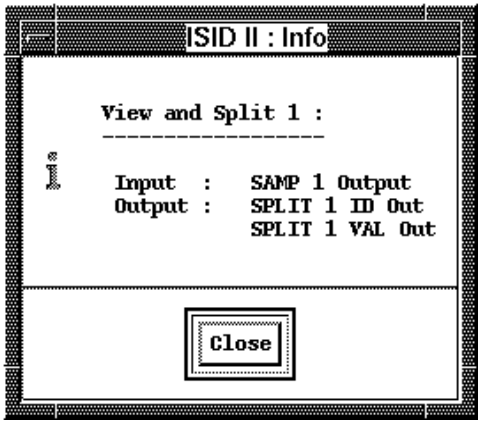


Figure 3.13 Information about the inputs and outputs of a building block.

Clicking the middle mouse button on a connection allows you to delete the connection or ask for information about the connection. Figure 3.14 shows a typical connection information window. It lists the origin and destination of the connection. It also lists the variable name of the data object going through the connection.

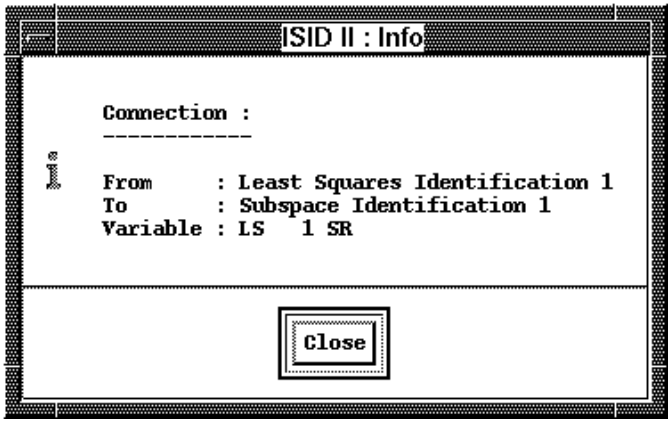


Figure 3.14 Information about a connection.

Clicking the middle mouse in the background pops up a menu that is very similar to the Chain menu. It allows you to set the chain execution mode and to start, step through and stop the chain.

3.5.3 Right Mouse Button

Clicking the right mouse button anywhere in the Chain Plot pops up a menu (very similar to the Windows menu) that enables you to iconify and reorder all windows (see Section 3.8). You can also deselect all selected algorithms. To select a menu item from this popup you must use the **right** mouse button.

3.6 Manipulating Parameters

In this section we discuss the two types of parameters and how they can be altered:

- **Parameters:** the main parameters of a building block.
- **Defaults:** the auxilary parameters of a building block.

The distinction between the two types of parameters is only made in this section and in the description of the building blocks (Chapter 4). When we refer to parameters in other parts of the ISID Part 2 manual, both parameters *and* defaults apply. From an algorithmic point of view, there is no real difference between the two types. The difference is from a users point of view, because the *parameters* are readily available to the user, while the *defaults* are more hidden.

3.6.1 Setting Parameters and Defaults at Opening

It is often useful to set the parameters and the defaults when an algorithm is opened. All parameters and defaults are initially set to sensible default values. However, once you know more about ISID and you get more acquainted with the algorithms and the syntax, you may want to change the defaults. As described in Section 3.3.1, it is very easy to alter the value of the parameters by hitting the Default button in the "OPEN" window (Figure 3.9). There are two different ways of entering parameter values:

- **By Value:** some parameters are displayed in a VarEdit Box. To enter a new value for a parameter, click on the box, enter the parameter, and hit `return`.

  When an illegal value is entered, an error message pops up on top of the "OPEN" window and the parameter is reset to its old value. "Window Width" (Figure 3.10) is an example of this type of parameter.

- **By List:** other parameters only have a few discrete settings (for instance "Computation" in Figure 3.10). These parameters can be altered by clicking on the parameter name ("Computation" in the case of Figure 3.10). A menu with the possible choices pops up and enables you to select the right parameter value.

3.6.2 Graphical Manipulation of Parameters

Once the algorithm is opened, the parameters can be interactively altered or the parameters at the bottom of the associated algorithm window can be changed. Graphical interaction with the algorithm windows involves clicking and dragging with the left mouse button. Algorithm-specific information about which parameters that accept interaction can be found in Chapter 4.

3.6.3 Manipulating the Defaults

To change the defaults select Special → Defaults. A window similar to Figure 3.10 pops up. This window only contains the defaults (and not the parameters, since these are displayed at the bottom of the algorithm window). The procedure for making changes is the same as described in Section 3.6.1.

3.7 General Plotting Features

All algorithm windows have the same plotting features. These features include data viewing, zooming and scaling. Moreover it is possible to put a grid on a plot, to alter the title and to change the axis from logarithmic to linear and back. Graphical interactions with the algorithm plot are tabulated in the quick reference Section 3.1.

3.7.1 Zooming

Zooming functions involve holding down the `<Ctrl>` key and clicking the left mouse. Conceptually there are two plots in all ISID algorithm windows:

- The original plot containing the original data.
- The zoomed plot containing the zoomed data.

Only one of the two plots can be visible, so it seems there is only one. To switch between the original plot and the zoomed plot hold down the `<Ctrl>` key and click on the plot title.

- **Display one subplot:** Hold down the `<Ctrl>` key and click in the background of the subplot you want to see. The zoomed plot will become visible and it will display the selected subplot. To return to the original plot, hold down the `<Ctrl>` key and click on the title or in the background of the zoomed plot. The original plot will appear again.

- **Zoom in on a plot:** Zooming involves using a selection box, or "lasso" to select the area you want zoomed. To lasso some data, place the cursor near the data. Hold down the `<Ctrl>` key, click and drag the left mouse button until the data is surrounded by the box. Release the mouse and the data in the box will appear in the zoomed plot. To go back to the original view follow the same procedure as when displaying a subplot. Figure 3.15 illustrates the principle of data zooming with a lasso.

- **Adjust the axis:** An axis in the original and zoomed plot can be adjusted graphically. Hold down the `<Ctrl>` key, click on an axis, and drag towards the middle of the plot. When you release, the axis will be updated. Note that this is different from zooming, since in this case the axis of both plots - original and zoomed - are updated.

  To expand an axis, hold down the `<Ctrl>` key and click outside the axis you want to enlarge. The axis will be zoomed out at a factor of two, until it reaches the minimum or maximum data range.

  To set all axes back to normal (their maximum value), select Plot → Scale → Automatic.

  Plot → Scale → Manual allows you to set the axes manually by typing in the minimum and maximum range. This menu entry pops up the "MANUAL SCALE" window (see Figure 3.16). You can change any of the four entries in this window to adjust the axes of the plot. The plot will automatically be updated every time you hit `return`.

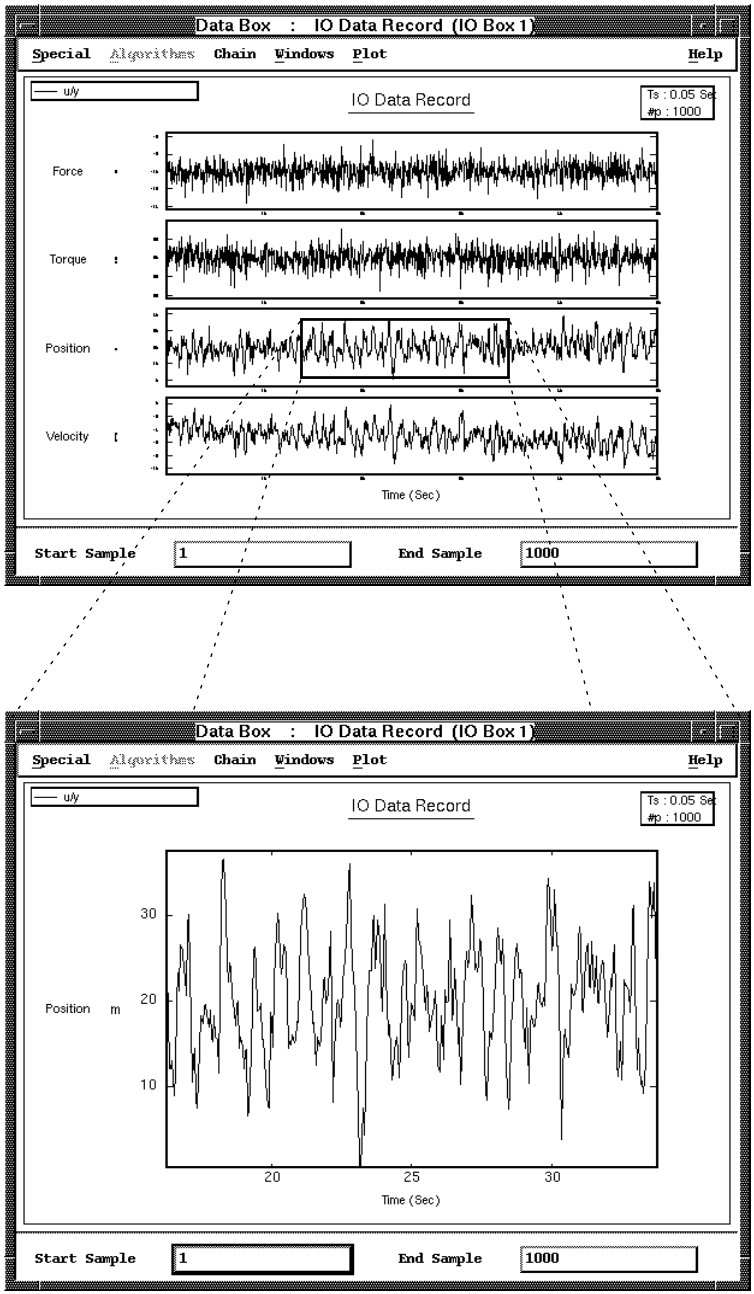


Figure 3.15 Original plot with lasso and zoomed plot.

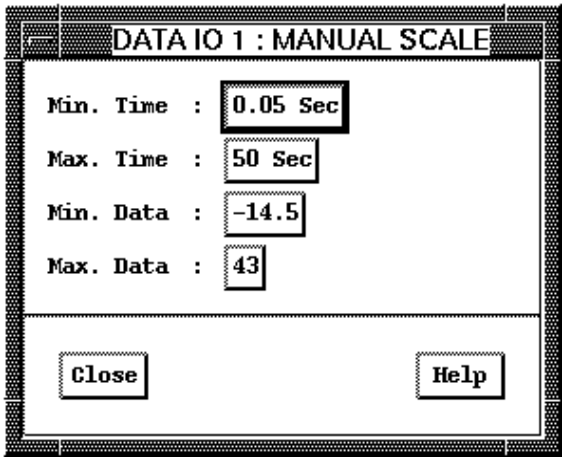


Figure 3.16 The "MANUAL SCALE" window allows you to adjust the plot axes manually.

Apart from the zooming features specific to ISID, there is also a built-in magnifying glass. Pressing the middle mouse button anywhere in the plot magnifies a square area centered at the cursor. To change the focus hold down the middle mouse button and drag across the plot. The center of the magnification corresponds to the tip of the cursor.
Pressing `<Ctrl>` with the middle mouse button increases the size of the magnified box. Pressing `<Shift>` with the middle mouse button increases the zooming factor. Pressing `<Shift>` and `<Ctrl>` together, along with the middle mouse button, yields a large zoom box with a large magnification factor.

3.7.2 Data Viewing

By pointing at or near an object in a plot and pressing the **right** mouse button, a small window appears that identifies the object (by the object name) and gives the coordinates of the nearest data value. If the right mouse button is pressed and dragged, the selected object will be tracked, even if another object comes close.
Pressing `<Shift>` along with the right mouse button allows the user to get values on a piecewise linear curve that interpolates the data values. Data viewing is very helpful when you want to inspect the value of an outlier, the error percentage in an error norm plot, etc ...

3.7.3 Plotting Modes

The Plot menu supplies several entries that enable you to change the plot:

- **Plot → X-Axis:** linear or logarithmic X-axis. When the X-axis contains negative data, it is not possible to set it to logarithmic.
- **Plot → Y-Axis:** linear or logarithmic Y-axis. When the Y-axis contains negative data, it is not possible to set it to logarithmic.
- **Plot → Bar Graph:** plot the data as a curve or as a bar graph. Typically error norms and singular values are plotted as bar graphs.
- **Plot → Scale:** adjust the range of the plot. See Section 3.7.1.
- **Plot → Text:** change the title of the plot. A dialog box containing the current title pops up. You can enter a new title.

3.7.4 Making Hardcopies

It is easy to make a hardcopy of an ISID window. Select Hardcopy to write the contents of a window to a file in your current Xmath working directory. You can also specify a pathname as part of the file name.

- To make a hardcopy of the Chain Plot select File → Hardcopy Chain Plot. A pop up will ask you for the name of the file you want to dump the Chain Plot in. After hitting OK, the cursor changes to a cross. Move the cursor over the Chain Plot and click the left mouse button. The Chain Plot is then dumped into the specified file.

  Remarks:

  - This only works for X Window based implementations of Xmath.
  - Make sure no other windows overlap the Chain Plot. Because this is an X Window dump, any images in front of the Chain Plot will be included in the Postscript file.
  - Postscript is the only output format available for Chain Plot hardcopies.

- To make a hardcopy of an algorithm plot, select Special → Hardcopy Plot. A pop up will ask you for the name of the file you want to dump the Chain Plot in. The extension of the name determines the file format the hardcopy is written to:

  - **No extension or extension .ps:** Adobe Postscript
  - **extension .eps:** Adobe Encapsulated Postscript
  - **extension .hpgl:** Hewlett-Packard Graphic Language
  - **extension .pict:** Apple Macintosh QuickDraw/PICT format

3.8 Manipulating Windows

When working with ISID, you will soon find your screen full of windows. This section discusses special utilities to manipulate ISID windows.

3.8.1 Iconify All

Select Windows → Iconify All, to lower all algorithm windows at once. Only the Chain Plot window will remain. This enables you to get an overview of the session you are currently running. You can re-open the algorithms you are interested in by double clicking on the icon (or by using `<Shift>` click). Alternatively, hitting `<Ctrl> - a` in any of the ISID windows iconifies all windows.

3.8.2 Reorder All

Select Windows → Reorder All to reposition all windows in the original order. The window corresponding to the first opened icon is used as a reference to justify the windows, so moving the first window opened and selecting Windows → Reorder All will move the whole "card box" of windows. In other words, the first window (typically a data box) enables you to determine the place of the ordered windows. Alternatively, hitting `<Ctrl> - r` in any of the ISID windows reorders all windows.

3.8.3 Show Chain Plot

When the Chain Plot is hidden by many algorithm windows, the menu entry Windows → Chain will raise (and/or deiconify) the Chain Plot window. This is useful when many windows are obscuring the identification session. Alternatively, hitting `<Ctrl> - w` in any of the ISID windows pops up the Chain Plot.

3.9 Customizing ISID

You can customize your ISID session with options to the `isid` command. Instead of typing these options every time you start ISID, you can also create a startup file that defines the values of the options once and for all.

3.9.1 At Startup

`isid` keywords are as follows (alternatively, type `help isid` in the Xmath command area):

- **destroy:** this will destroy the current ISID session. It has the same effect as the menu entry File → Exit. Note that typing `isid` in the Xmath command area,

while ISID is running will not destroy the ISID session. You should only type `isid` without keywords when:

- there is no ISID session running
- you are trying to recover from a crash. After you have aborted the debugger, type `isid` to recover. This will restart the execution of the chain and reset some variables. In most cases, this enables you to (at least) save your results.

- **startup chain:** typing

  ```
  isid <chain_name>
  ```

  will start ISID and will load the specified chain automatically at startup.

3.9.2 With a Startup File

You can customize your own ISID session by copying the file

`$XMATH/modules/isid/startup_isid.ms`

to your `~/xmath` directory. This file enables you to define the default chain at startup, the execution mode at startup, and enable a beep to sound whenever an error dialog pops up. Read the comments in the startup file for more information.

Chapter 4

Building Blocks

This chapter describes the ISID building blocks in detail. For each building block, you can find:

- the input types
- the output types and names
- the parameters and defaults
- the plot and the graphical interaction with the plot
- the algorithm and the related Xmath functions.

Each of the algorithms has one or more output(s). The names of these output(s) are composed of:

- the *short name* of the algorithm which can be found in the tables on the next page. The short name of the algorithm typically corresponds to the Xmath function name. For instance `LS` for least squares identification.
- the *instance number* of the algorithm which can be found in the top right corner of the algorithm icon.
- a *name* specific to the output which can be found in the reference pages and in the on-line helps.

For instance, the model output of the second instance of the subspace identification algorithm is named:

```
SDS 2 Model
```

where `SDS` is the *short name*, `2` is the *instance number* and `Model` is the *name*. The output of the first instance of the frequency response algorithm is named:

```
FREQ 1 Output
```

In this chapter and in the on-line help entries, the instance number is represented by a "Pound" sign (`#`). So the outputs of the subspace identification algorithm are:

```
SDS # Model
SDS # SR
```

In the following tables we list all ISID building blocks. The type of the input and output data object is indicated with the following abbreviations:

- **IO:** input-output record
- **Freq:** frequency response
- **Imp:** impulse response
- **Model:** model
- **SR:** square root

If more than one input type is possible one of the following entries is used:

- **IO** & **Model:** both input-output data record *and* model should be present at the input.
- **IO** | **SR:** *either* an input-output data record *or* a square root should be present at the input.
- **Model(s)** + **Freq(s):** one or more models *and/or* one or more frequency responses should be present at the input.

Data Boxes	Short	Input	Output	Section	Page
Input Output Data Box	IO Box	Xmath Var.	IO	4.1	74
Frequency Data Box	Freq Box	Xmath Var.	Freq	4.1	77
Impulse Data Box	Imp Box	Xmath Var.	Imp	4.1	79
Model Data Box	Mod Box	Xmath Var.	Model	4.1	81
Square Root Data Box	SR Box	Xmath Var.	SR	4.1	83

Processing	Short	Input	Output	Section	Page
Detrend and Scale	DETREND	IO	IO	4.2	94
Peak Shaving	PEAK	IO	IO	4.2	85
Delay Estimation	DELAY	IO	IO	4.2	88
Filtering	FILT	IO	IO	4.2	91
Post Sampling	SAMPLING	IO	IO	4.2	97
View and Split	SPLIT	IO	IO (2)	4.2	100

Identification	Short	Input	Output	Section	Page
Least Squares	LS	IO | SR	Model & SR	4.3	102
Empirical Estimation	ETFE	IO	Freq & Imp	4.3	105
Subspace Identification	SDS	IO | SR	Model & SR	4.3	109
Instrumental Variables	GIV	IO	Model	4.3	112
Prediction Error Method	PEM	IO & Model	Model	4.3	114
Frequency Least Squares	FWLS	Freq	Model	4.3	117
Impulse Realization	IREA	Imp	Model	4.3	120

Validation	Short	Input	Output	Section	Page
Error Norms	ERNO	Model(s) & IO(s)	-	4.4	123
Prediction	PRED	Model(s) & IO	IO	4.4	126
Prediction Errors	PER	Model(s) & IO	IO	4.4	128
Covariance	COVAR-PE	Model(s) & IO	Imp	4.4	130
Spectral Density	SDF-PE	Model(s) & IO	Freq	4.4	133
Cross Correlation	CROSS-PE	Model(s) & IO	Imp	4.4	136

Display	Short	Input	Output	Section	Page
Frequency Response	FREQ	Model(s) + Freq(s)	Freq	4.5	139
Impulse Response	IMP	Model(s) + Imp(s)	Imp	4.5	142
Spectral Density	SDF	Model(s) + Freq(s)	Freq	4.5	144
Covariance	COV	Model(s) + Imp(s)	Imp	4.5	146
Poles & Zeros	PZ	Model(s)	-	4.5	148

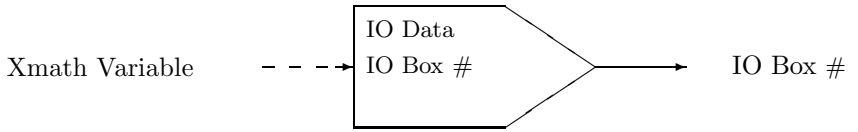
4.1 Data Boxes

DATA BOX : IO DATA RECORD

Description

The input-output data record box is one of five possible data boxes that can start a chain. The box can be empty or contain an input-output data record that was loaded from the Xmath workspace.

The time axis of the data record at the box output can be altered, so that data points at the beginning and end of the record can be removed. One or more channels can be selected (by default all channels are selected). The data record at the box output is equal to the selected channels over the selected time axis.



Inputs

Xmath Variable: a real matrix or a real parameter dependent matrix (PDM), with minimally 10 elements in the domain. For the PDM, at least one of the sizes must be equal to one. For more information about PDMs, refer to the Xmath Basics manual.

Outputs

IO Data Record : IO Box #

Parameters

Start Sample	Sample number indicating the start of the time axis of the data record at the box output.
End Sample	Sample number indicating the end of the time axis of the data record at the box output.

Defaults

Active Inputs	Selected active input channel numbers in the data record at the box output.

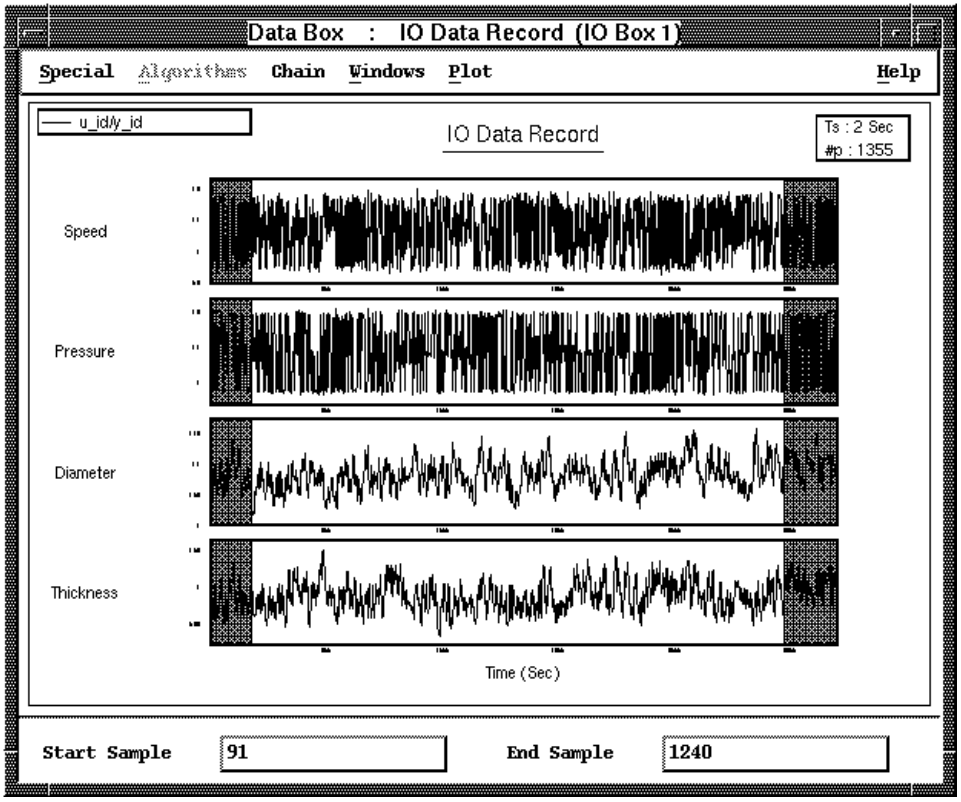
Active Outputs	Selected active output channel numbers in the data record at the box output.

Plot

The plot shows the data record that was loaded from the Xmath workspace.

To alter the start and end sample interactively, click on the data near either the beginning or end of the plot, and drag. While dragging, a vertical line indicates the current sample position, and the sample value at the bottom of the window is updated.

The selected channels are indicated by a (red) highlighted axis-box. The rejected data points (not in the new time axis) are grayed out.



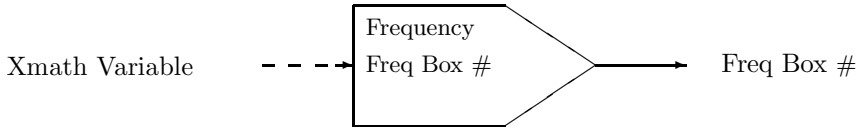
The IO Data Record Box Window.

Data Box : Frequency Response

Description

The frequency response box is one of five possible data boxes that can start a chain. The box is either empty or contains a complex frequency response that was loaded from the Xmath workspace.

The frequency axis of the response at the box output can be altered. One or more channels can be selected (by default all channels are selected). The frequency record at the box output is equal to the selected channels over the selected frequency axis.



Inputs

Xmath Variable: a complex PDM with at least 10 elements in the domain.

Outputs

Frequency Response : Freq Box #

Parameters

Start Sample	Sample number indicating the start of the frequency response axis at the box output.
End Sample	Sample number indicating the end of the frequency response axis at the box output.

Defaults

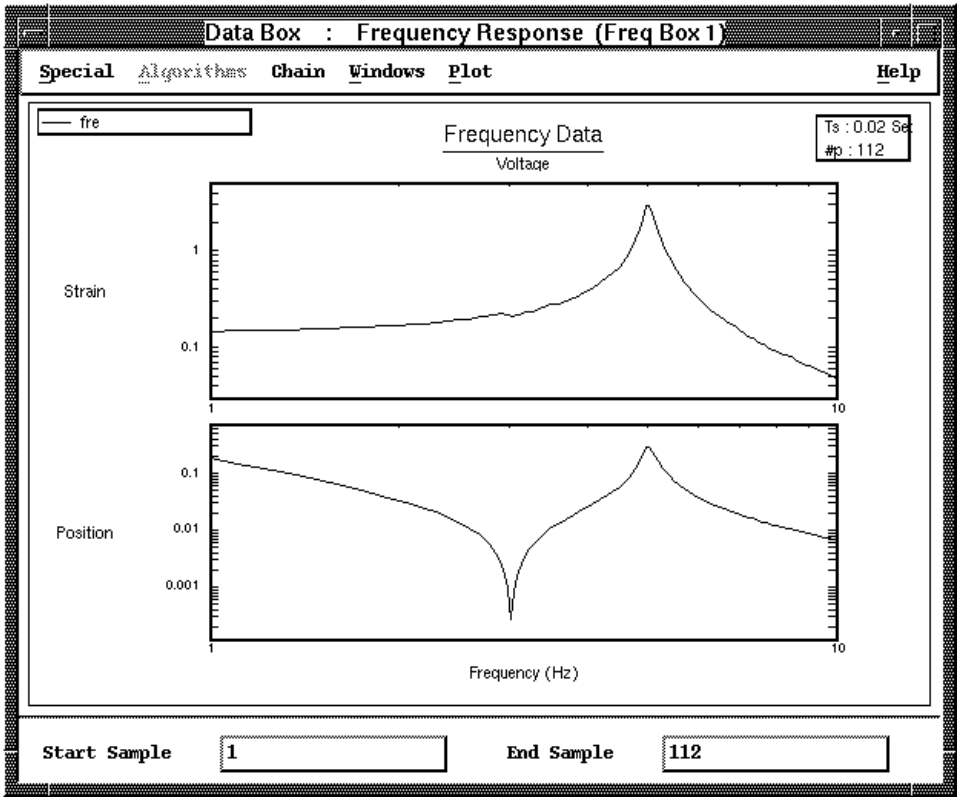
Active Inputs	Selected active input channels of the response at the box output.
Active Outputs	Selected active output channels of the response at the box output.

Plot

The plot shows the magnitude of a frequency response that was loaded from the Xmath workspace.

To alter the start and end sample interactively, click on the data near either the beginning or end of the plot, and drag. While dragging, a vertical line indicates the current sample position, and the sample value at the bottom of the window is updated.

The selected channels are indicated by a (red) highlighted axis-box. The rejected data points (not in the new frequency axis) are grayed out.



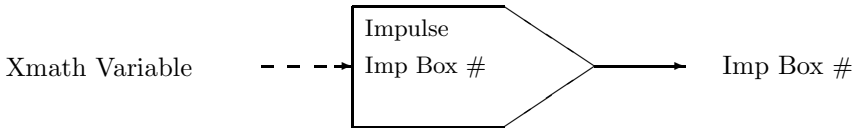
The Frequency Response Box Window.

Data Box : Impulse Response

Description

The impulse response box is one of five possible data boxes that can start a chain. The box is either empty or contains an impulse response that was loaded from the Xmath workspace.

The time axis of the response at the box output can be altered. One or more channels can be selected (by default all channels are selected). The response at the box output is equal to the selected channels over the selected time axis.



Inputs

Xmath Variable: a real PDM with at least 10 elements in the domain.

Outputs

Impulse Response : Imp Box #

Parameters

End Sample	Sample number indicating the end of the time response axis at the box output.

Defaults

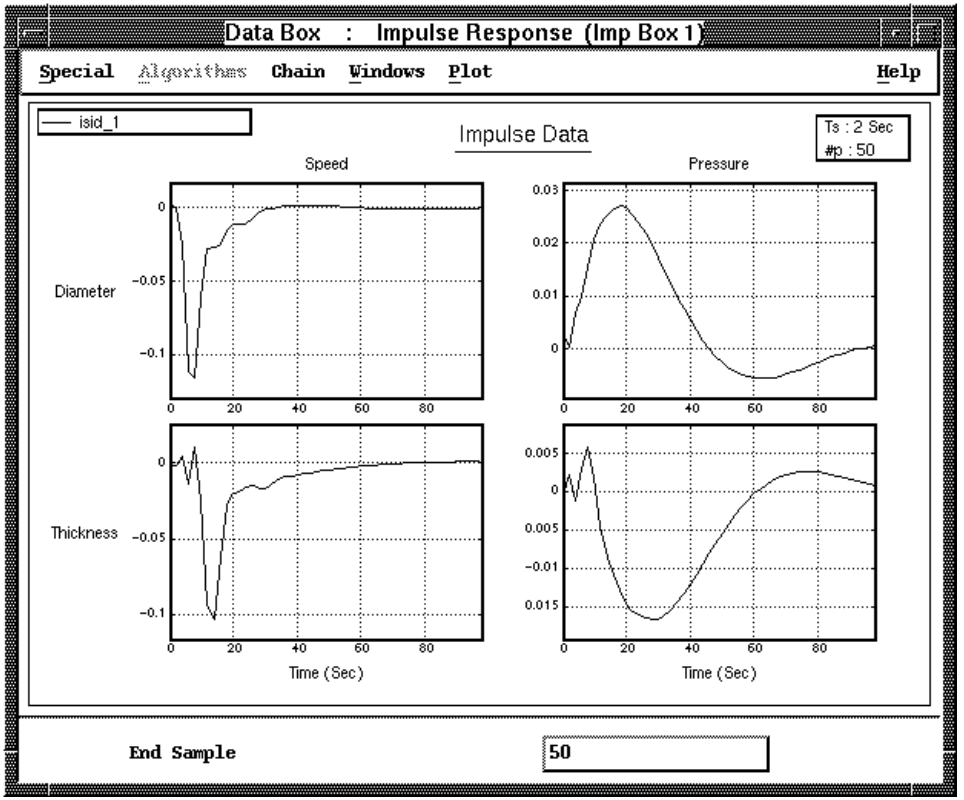
Active Inputs	Selected active input channels of the response at the box output.
Active Outputs	Selected active output channels of the response at the box output.

Plot

The plot shows an impulse response that was loaded from the Xmath workspace.

To alter the end sample interactively, click on the data near the end of the plot, and drag. While dragging, a vertical line indicates the current sample position, and the sample value at the bottom of the window is updated.

The selected channels are indicated by a (red) highlighted axis-box. The rejected data points (not in the new time axis) are grayed out.



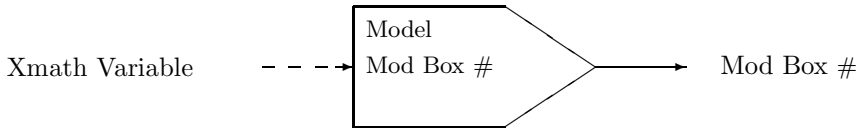
The Impulse Response Box Window.

DATA BOX : MODEL

Description

The model box is one of five possible data boxes that can start a chain. The box is either empty or contains a model that was loaded from the Xmath workspace.

The model at the box output is equal to the selected channels of the loaded model.



Inputs

Xmath Variable: a discrete time Xmath system with at least one state.

Outputs

Model :	Mod Box #

Parameters

Show On Plot	You can view the frequency response, impulse response, spectral density or covariance sequence of the loaded model. Default: Frequency Response.

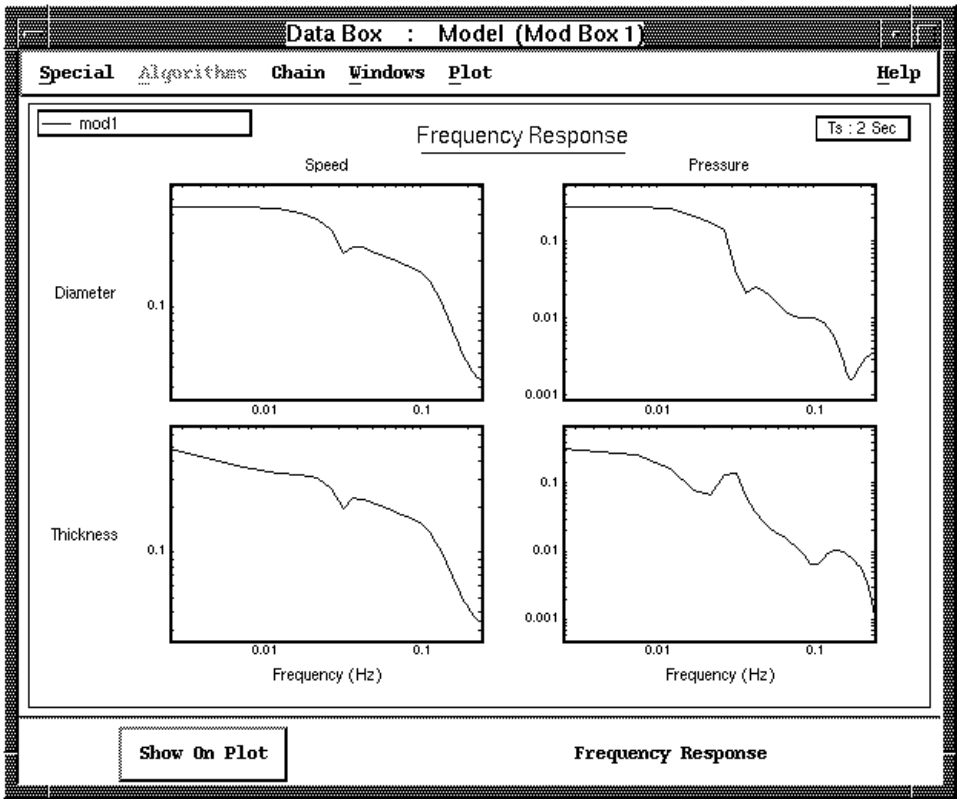
Defaults

Active Inputs	Selected active input channels of the model at the box output.
Active Outputs	Selected active output channels of the model at the box output.

Plot

The plot shows the frequency response (magnitude), impulse response, spectral density (magnitude) or covariance sequence of the loaded model. The selected channels are indicated by a (red) highlighted axis-box.

There is no algorithm-specific interaction with the plot (only regular zooming).



The Model Box Window.

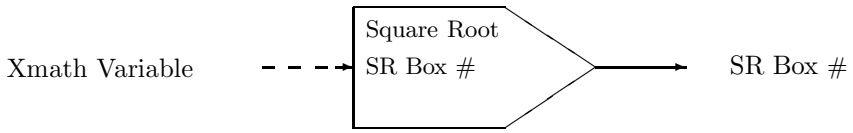
Data Box : Square Root

Description

The square root box is one of five possible data boxes that can start a chain. The box is either empty or contains a square root that was loaded from the Xmath workspace.

The ISID Part 1 manual contains more information about the square root object. Conceptually, it is an intermediate R factor of a QR decomposition of the data.

The square root at the box output is equal to the selected channels of the loaded square root.



Inputs

Xmath Variable: a Least Squares square root object (see the ISID manual).

Outputs

Square Root : SR Box #

Parameters

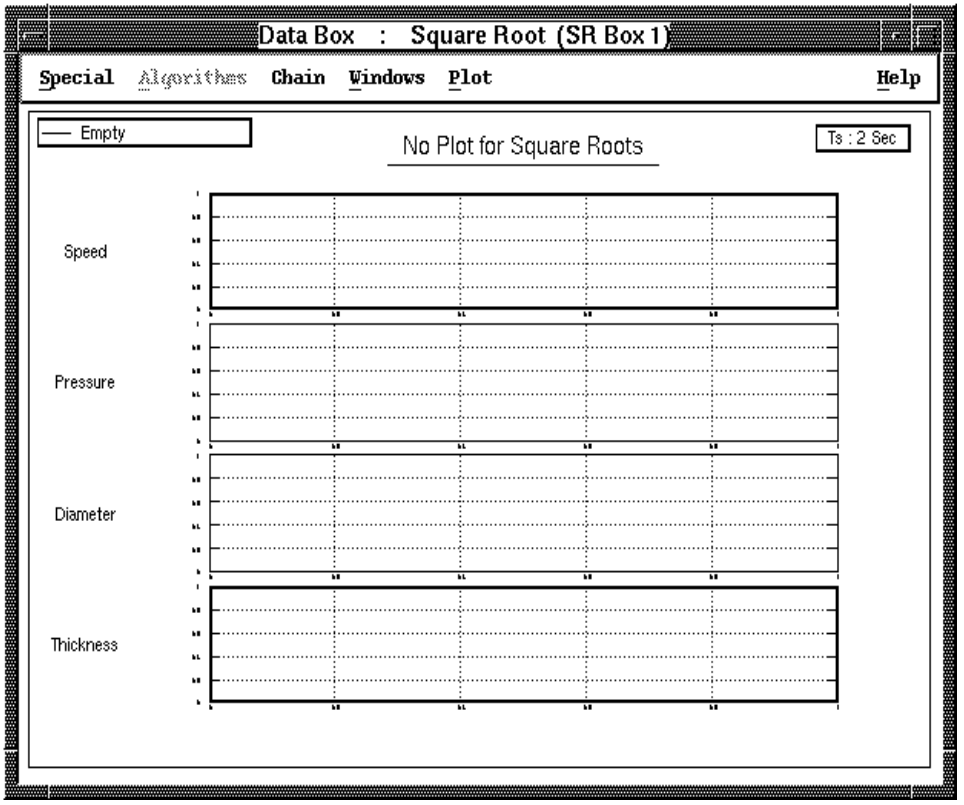
None

Defaults

Active Inputs	Selected active input channels of the square root at the box output.
Active Outputs	Selected active output channels of the square root at the box output.

Plot

Since a square root does not have anything specific to display, the plot only displays the input and output labels. The selected channels are indicated by a (red) highlighted axis-box.



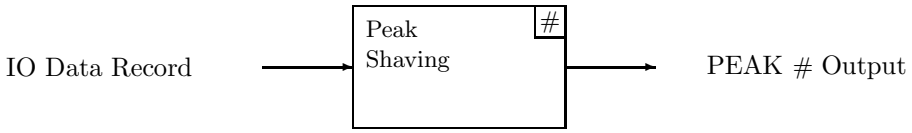
The Square Root Box Window.

4.2 Processing

Peak Shaving (PEAK)

Description

Removes outliers due to sensor failure or interference. An upper and lower limit can be entered graphically or manually. A statistical procedure is then applied to remove the outliers (see algorithm below), and replace them with linearly interpolated values.



Inputs

IO Data Record

Outputs

IO Data Record :	PEAK # Output

Parameters

Show On Plot	Show either the Peak Shaved or the Original data on the plot. Default: Original Data.
Levels	Hitting this push button will pop up a table with the upper and lower levels used to determine the outliers. These values correspond to the upper and lower levels shown on the plot.

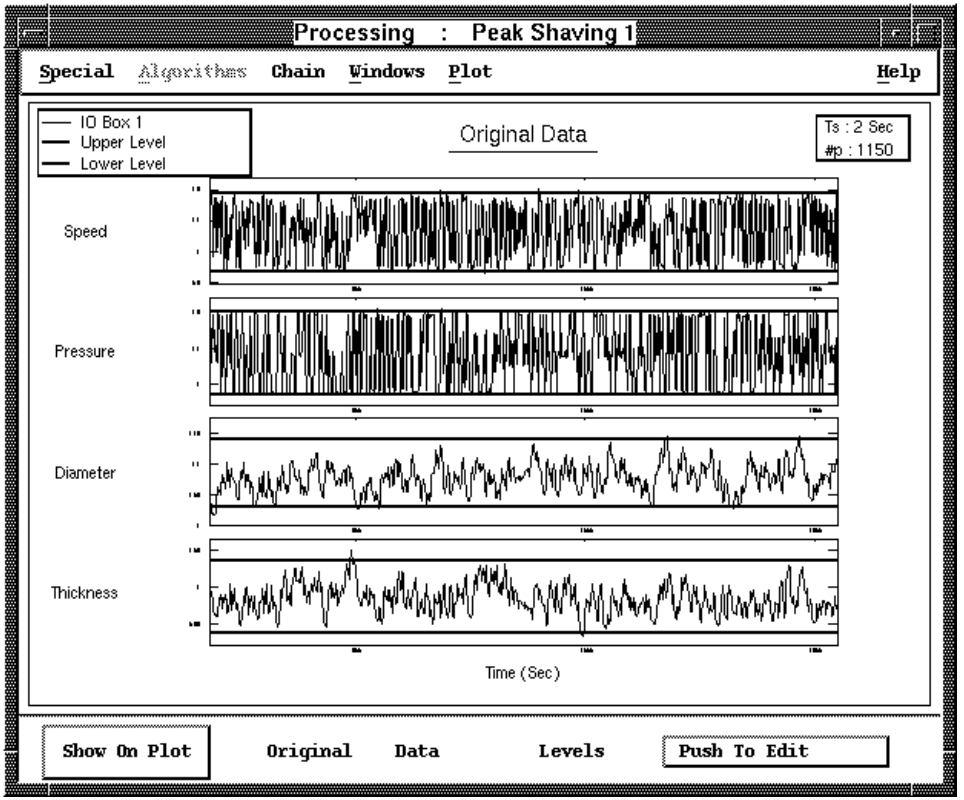
Defaults

Sigma Factor	See algorithm below.

Plot

The plot shows either the original or the peak shaved data (with the upper and lower level).

The upper and/or lower level can be adjusted by selecting the horizontal lines (blue and red) and dragging them up and down. Upon release, a dialog box containing the levels pops up. All channels can be adjusted at the same time. No calculations are performed until OK is selected.



The Peak Shaving Window.

Algorithm

The outliers are determined as follows:

The standard deviations of the signals lying between the upper and lower level are calculated. All signal points lying outside the band determined by twice the standard deviation (Sigma Factor) are treated as outliers.

Outliers are replaced by linearly interpolated values. The upper and lower level defaults are chosen such that 95 out of 100 samples lie between the levels.

Xmath Functions

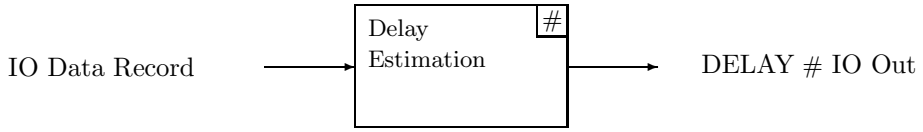
peak_shave(), sort(), interpolate()

DELAY ESTIMATION (DELAY)

Description

Estimates and extracts delays, which are estimated through the construction of a non-parametric impulse response estimate.

The input and output signals are shifted back and forth to compensate for the estimated delays. This algorithm is useful when there are large delays in the process. When large time delays are not compensated for, the model order of the identified model becomes too high.



Inputs

IO Data Record

Outputs

IO Data Record : DELAY # IO Out

Parameters

Delays Hitting this push button will pop up a dialog box containing the estimated delays. The values in the table correspond to the time delays indicated by the vertical lines on the plot.

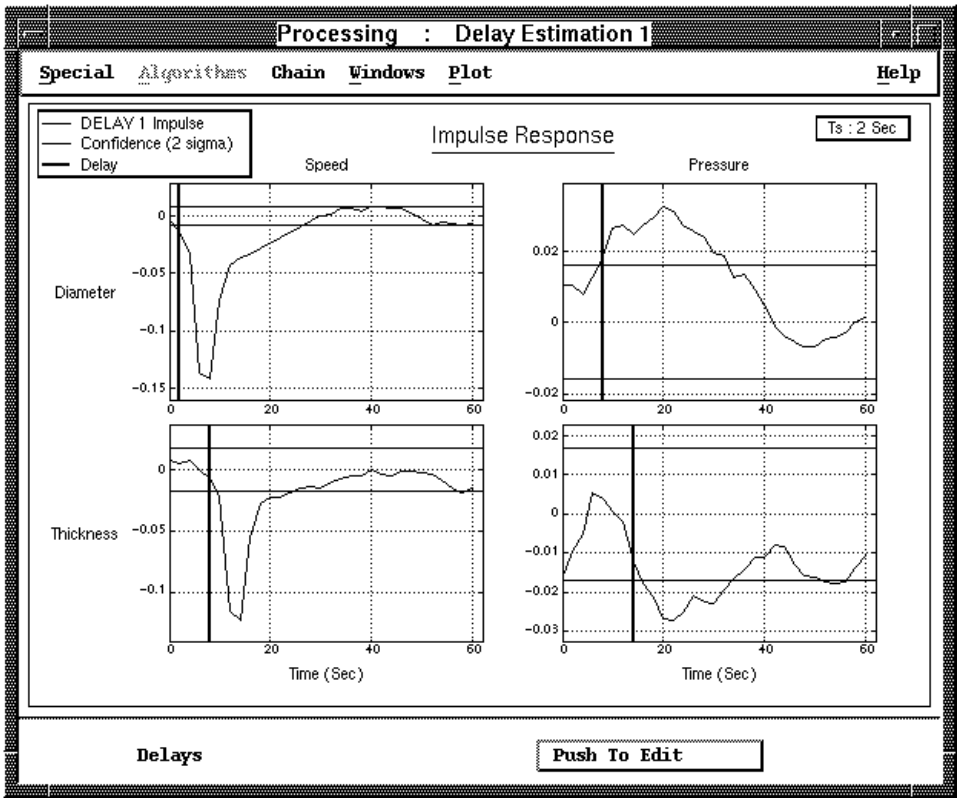
Defaults

#Impulse Lags Number of samples of the estimated non-parametric impulse response.

Plot

The plot shows the estimated non-parametric impulse response (see algorithm below). The confidence levels are drawn at twice the standard deviation of the negative impulse response estimate (see below).

To adjust time delays, select the vertical lines and drag them left and right. Upon release, a dialog box with the time delays pops up (unless there is only one input and one output). Time delays for all channels can be adjusted at the same time. No calculations are performed until the OK button is selected.



The Delay Estimation Window.

Algorithm

The non-parametric impulse response is estimated using a correlation technique in the time domain.

The confidence levels are determined from the standard deviation of the impulse response for negative time lags. When the system is causal, non-zero negative correlations can only be due to noise. The confidence levels are drawn at twice the estimated standard deviation. The impulse response is only significantly different from zero when it lies outside the confidence levels.

After the delays are determined the signals are shifted back and forth, to compensate for the delays. The number of degrees of freedom in performing this action (nu+ny-1) is smaller than the number of delays that need to be compensated for (nu*ny). The requested delays are matched as closely as possible (through a linear programming problem). This implies that ISID can correct delays defined by the user.

Xmath Functions

delay_imp(), get_corr(), delay_adjust()

FILTERING (FILTER)

Description

Filters an input-output data record with a low pass, high pass or band pass filter using Butterworth, Chebychev or Elliptic methods.

Filtering can be used to eliminate high or low frequency disturbances from the data.

Inputs

IO Data Record

Outputs

IO Data Record : FILTER # Output

Parameters

Min. Freq.	The smallest 3 dB cutoff frequency of the filter (disabled when low pass filtering).
Max. Freq.	The largest 3 dB cutoff frequency of the filter (disabled when high pass filtering).

Defaults

Order	The order of the filter.
Type	Choice of Low Pass, High Pass, Band Pass and Band Stop filtering. Default: Low Pass.
Method	Choice of Butterworth, Chebychev and Elliptic filtering. Default: Butterworth.
Magn. Pass.	The passband will have a magnitude between 0 dB and minus Magn. Pass. (in dB). This is the maximal ripple on the pass band.

Magn. Stop.	The stopband will always be lower than minus Magn. Stop. (in dB). This is the attenuation of the stop band.
Show On Plot	Choice between Magnitude or Phase. Default: Magnitude.

Plot

The plot shows either the magnitude or the phase of the filter (depending on Show On Plot).

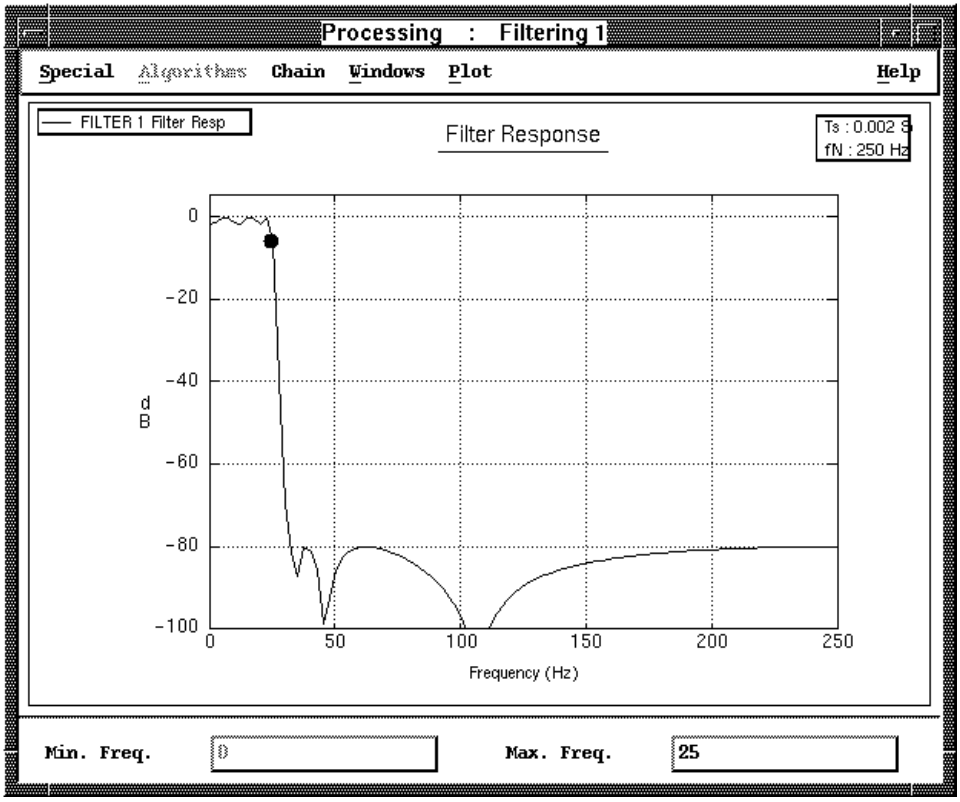
To alter the 3 dB cut off frequencies, select and drag the (red) dot(s). The numerical values at the bottom of the window are adjusted automatically.

Algorithm

Starting from the minimal and maximal frequency, the Filters are calculated using buttconstr(), chebconstr() or ellipconstr(). This allows for a simple specification of the filter characteristics.

Xmath Functions

buttconstr(), chebconstr(), ellipconstr()



The Filtering Window.

DETREND AND SCALE (DETREND)

Description

Removes trends from input-output data records with orthogonal polynomial fitting.

The input-output data can be further conditioned by Equal Energy data scaling. This leads to unit energy input and output signals.

Inputs

IO Data Record

Outputs

IO Data Record : DETREND # Output

Parameters

Show On Plot — Display either the detrended data or the trend that was removed from the data. Default: Detrended Data.

Scaling — When set to Equal Energy Scaling, the inputs and outputs will be scaled to unit energy. This processing should be performed before applying a scaling sensitive identification algorithm. Default: No Scaling.

Defaults

Detrend — Select which channels to detrend: Inputs and Outputs, Only inputs or Only outputs. Default: Inputs and Outputs.

Order — Enter the order of the polynomial fit (see algorithm below). Should be below 20.

Plot

The plot shows either the detrended data or the trend in the data (depending on the Show On Plot setting). The order of the fitting polynomial can be determined by inspection.

There is no algorithm specific interaction with the plot (only regular zooming).

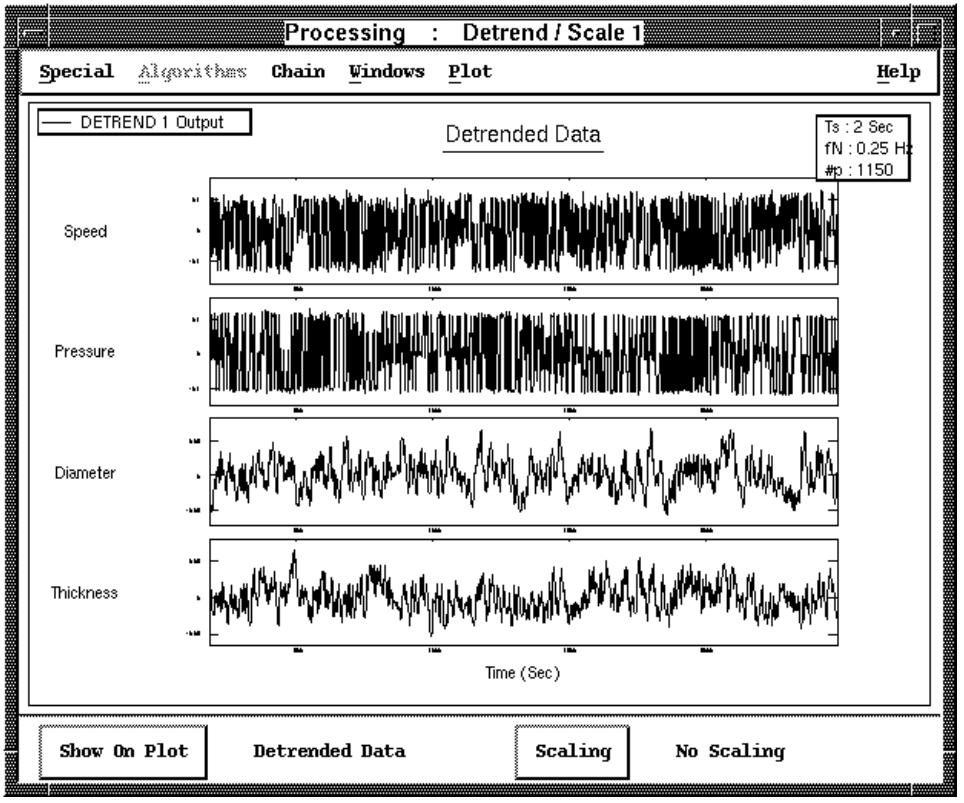
Algorithm

The data is detrended by fitting 20 or less orthogonal polynomials (of order 20 or less) through the data. The Polynomial Order defaults to 1, which implies that the mean and linear trends are removed from the data. This suffices in most practical cases.

The data is scaled by dividing each selected channel by its standard deviation.

Xmath Functions

poltrend(), covariance()



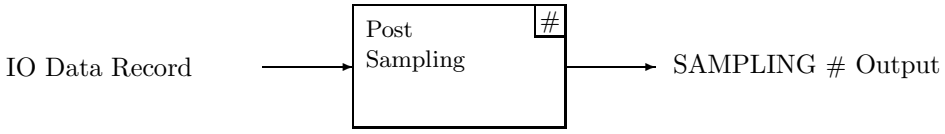
The Detrend and Scale Window.

Post Sampling (SAMPLING)

Description

Post sampling of an input-output data record. After (possible) anti-aliasing filtering, the input-output signal is resampled every N samples.

The frequency content is displayed (FFT or spectral density). This guides you in the decision of the post sampling factor.



Inputs

IO Data Record

Outputs

IO Data Record :	SAMPLING # Output

Parameters

Show On Plot	View the Fast Fourier Transform or the Spectral Density of the input-output signal, or post sampled input-output data. Default: Fourier Transform.
Post Sampling Factor	The signal will be resampled every Post Sampling Factor sample.

Defaults

Aliasing Filter	When set to Automatic, the 3 dB cut off frequency of the aliasing filter is automatically set to 90 percent of the new Nyquist frequency (after post sampling). The aliasing filter can also be set to manual or turned off. Default: Automatic.
Filter Frequency	Manual Aliasing Filter frequency.

Plot

The plot shows either the fast Fourier transform or the spectral density of the input-output signal. The new Nyquist frequency (after post sampling) is indicated by a vertical line (not visible when the Post Sampling Factor is equal to one).

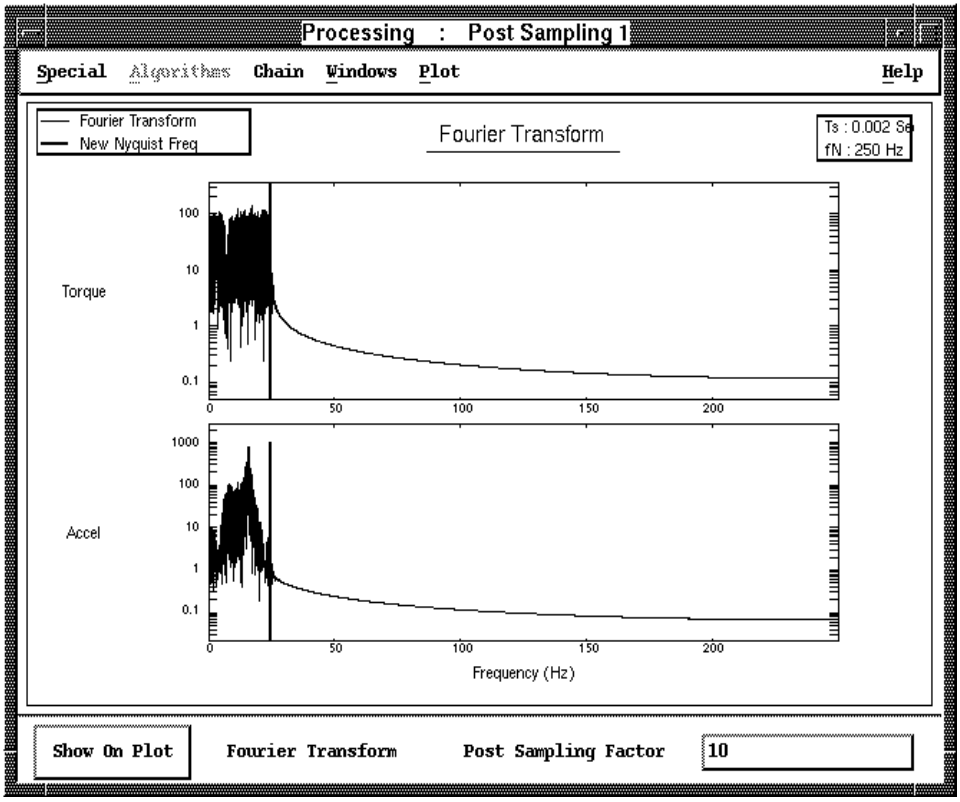
The Post Sampling Factor can also be adjusted by selecting and dragging the new Nyquist frequency (vertical line).

Algorithm

The input-output data is filtered by an anti-aliasing filter (if a filter is selected), then it is post sampled. Post sampling is useful when the sampling period of the original signal was chosen too small. This is generally indicated by a drop in the energy of the output signal from a certain frequency on (while there is still energy in the input signal).

Xmath Functions

fft(), sdf()



The Post Sampling Window.

VIEW AND SPLIT (SPLIT)

Description

Displays an input-output data record and splits it into two parts: identification and validation.

This function is typically used after a data set is processed by a chain of processing algorithms. The first output of the View/Split algorithm is then redirected to the identification algorithms, the second output to the validation algorithms.

Inputs

IO Data Record

Outputs

IO Data Record :	SPLIT # ID Out
IO Data Record :	SPLIT # VAL Out

Parameters

Splitting Sample	Splitting edge between identification and validation output.

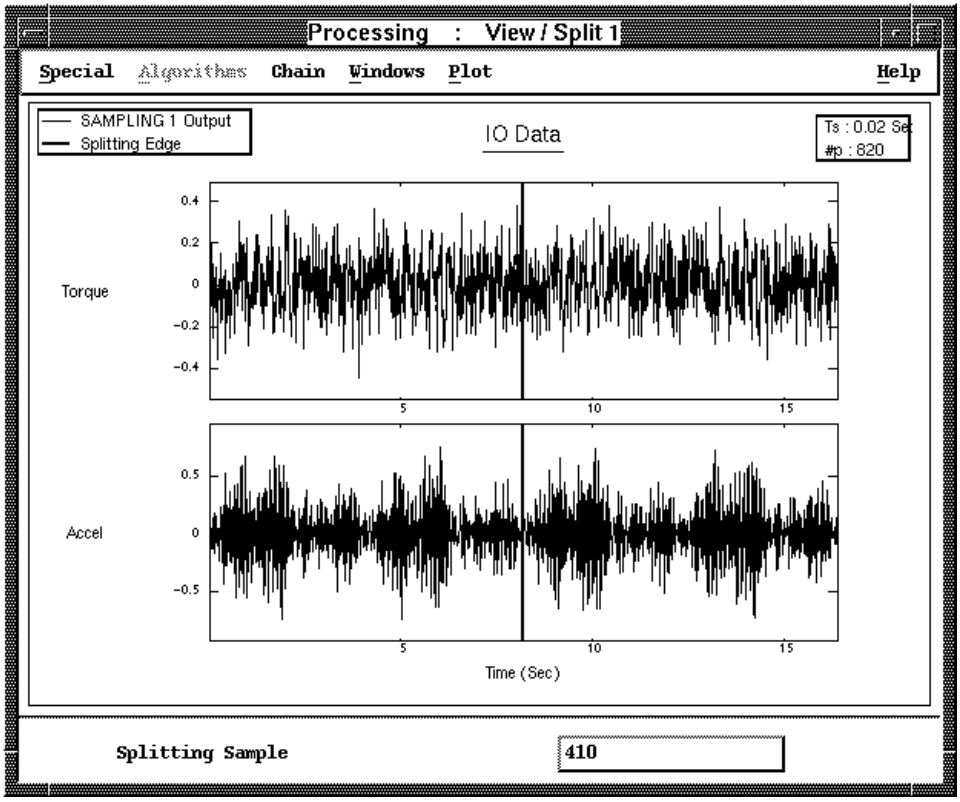
Defaults

None

Plot

The plot displays the input-output data and a vertical line at the splitting edge dividing the identification and validation output.

The splitting sample can be altered by selecting and dragging the vertical line.



The View and Split Window.

Algorithm

This algorithm is useful to split a pre-processed input-output data record into an identification and a validation part.

Xmath Functions

None

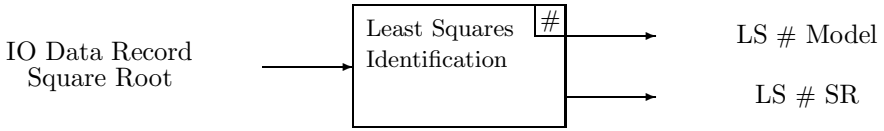
4.3 Identification

LEAST SQUARES IDENTIFICATION (LS)

Description

Identifies a least squares (ARX) state space innovations model from raw input-output data or from an intermediate square root.

The identified model and calculated square root are available at the output.



Inputs

IO Data Record or Square Root

Outputs

Model :	LS # Model
Square Root :	LS # SR

Parameters

Order Den. (A)	ARX model order of the Denominator. Changing this order automatically updates the ARX model order of the numerator. Order Den. can be a number or a vector of numbers.
Order Num. (B)	ARX model order of the numerator. Order Num. can be a number or a vector of numbers.

Defaults

Max. Order	The maximal order (maximal number of regression lags) for the calculation of the least squares models. All models with an order smaller than Max. Order can be obtained.

Feedthrough	Create a Feedthrough term in the model or not. Default: Feedthrough.
Lattice	Computation of the square root using a fast (but numerically less stable) lattice algorithm. Default: No Lattice.
Show On Plot	Choice between the relative error (the standard deviation of the error divided by the energy in the original signal) and the absolute error. Note that these errors do not detect an unstable system i.e. they will still be finite for unstable systems. Default: Absolute Error.

Plot

The plot shows either the relative or the absolute norm of the diagonal terms of the prediction error variance matrix (depending on Show On Plot). These values indicate the order of the identified model(s). Normally, the error norms are (almost) constant above a certain order. The smallest order that obtains this level is a good choice.

The order of the identified model can be altered by clicking with the left mouse on the bar corresponding to the required order. To obtain more than one order, click and drag.

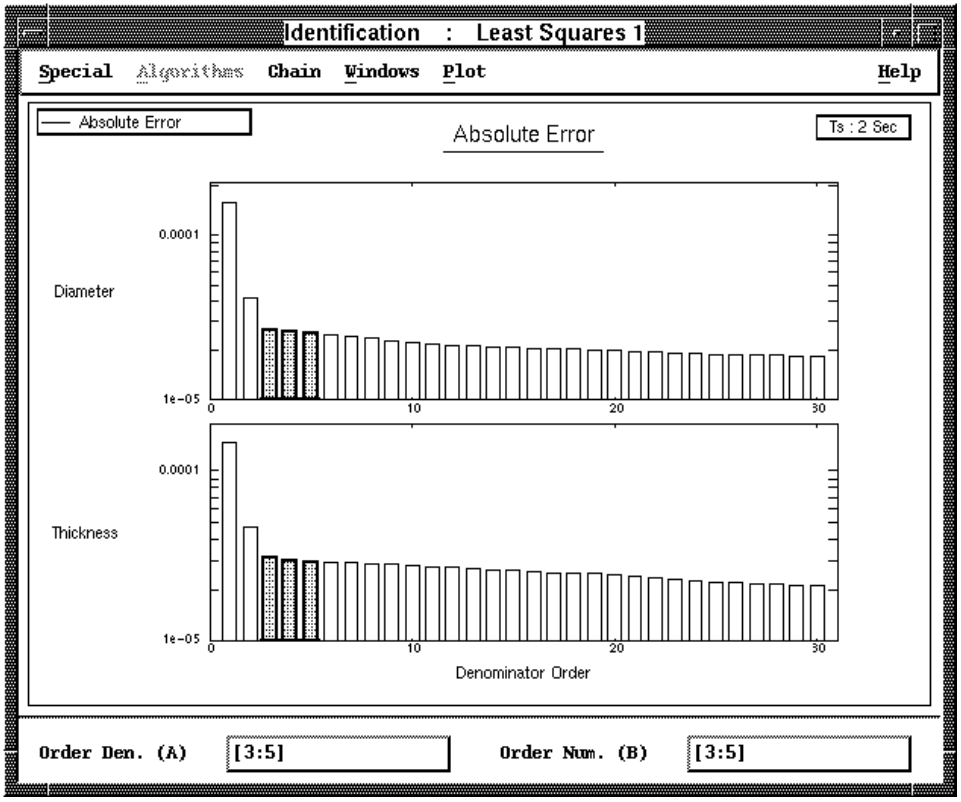
Algorithm

The algorithm minimizes the equation error variance. This often leads to very high order models. The order can be further reduced by calculating the frequency response followed by a Frequency Weighted Least Squares identification. Alternatively, one can also calculate the impulse response followed by an impulse Realization.

For more details, see the ISID Part 1 manual.

Xmath Functions

ls()



The Least Squares Identification Window.

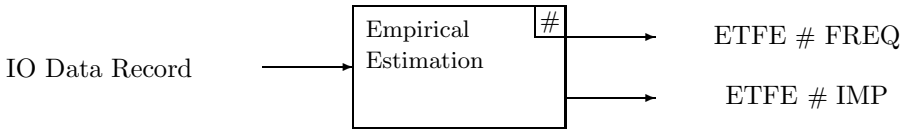
Empirical Estimation (ETFE)

Description

Find the Empirical Transfer Function Estimate (ETFE) by dividing the input-output cross spectral density by the input spectral density.

An empirical estimate of the impulse response is also computed. A frequency weight to emphasize certain frequencies can be introduced.

Both the ETFE and the impulse response are available at the output.



Inputs

IO Data Record

Outputs

Frequency Response :	ETFE # FREQ
Impulse Response :	ETFE # IMP

Parameters

Show On Plot	Show the Emperical Transfer Function Estimate (ETFE), the weighted impulse response (Impulse), the coherence between the signals (Coherence), the (cross)spectral density of the signals (SDF) and the covariance of the signals (see also IO Combination for the definition of the signals). Default: ETFE .
Mode	When a response in the frequency domain is shown (ETFE, Coherence or SDF), it can be Magnitude or Phase. Default: Magnitude.

Defaults

Computation	See algorithm below. Default: Averaging SDFs.

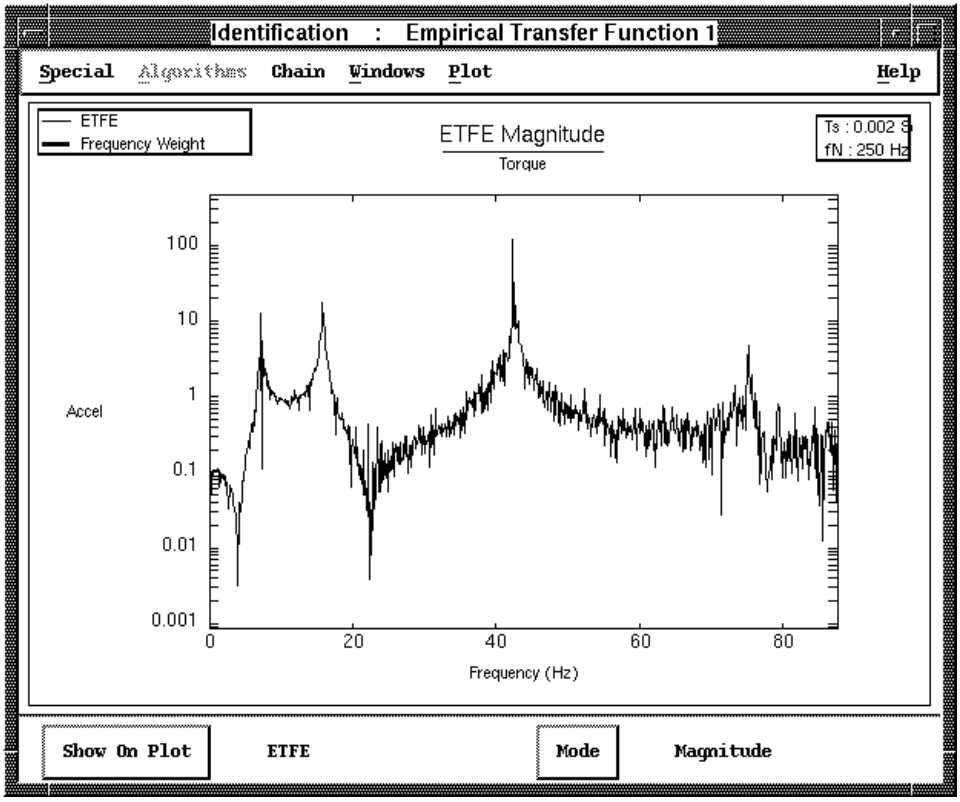
Window Type	Choice between Blackman, Hamming, Hanning, Triangular and Rectangular. Default: Hamming.
Window Width	See algorithm below.
Window Overlap	See algorithm below.
SDF-AR Order	Only active when Computation is set to Auto Regressive.
IO Combination	The plotted spectral density, covariance or coherence of these signals is computed (Input-Output, Input-Input or Output-Output). This parameter does not influence the result at the output (which is always equal to the estimated transfer function and impulse response). Default: Input - Output.
Min. Freq. Weight	The minimum break frequency of the weight.
Max. Freq. Weight	The maximum break frequency of the weight.
Attenuation Weight	The attenuation of the weight.
Show Weight	When a frequency response is plotted, the plotting of the attenuation weight can be suppressed. Default: No.

Plot

When a frequency response is plotted (Show On Plot is ETFE, Coherence or SDF), the frequency weight is also plotted (when Show Weight is Yes). This frequency weight influences the estimation of the impulse response (the ETFE is weighed before the inverse FFT is computed). The weight also influences the ETFE at the output (see algorithm).

To interact with this frequency weight, click and drag near one of the corners and/or lines. When a time domain response is plotted (impulse, covariance), no frequency weight is shown.

The "IO Combination" parameter determines the combination of plotted signals. This will only have an effect when the plot shows a coherence, SDF or covariance. The ETFE and impulse response (the outputs of the algorithm) are not affected by this parameter.



The Empirical Estimation Window.

Algorithm

The ETFE at the output is computed as follows. The (cross) spectral density function estimates (SDF) form the basic elements for any computational method:

(1) "Averaging SDFs": First, the data is partitioned into subwindows containing the number of data points indicated under "Window width". The data in each subwindow is tapered, using the selected Window Type, to prevent spikes in the Fourier transforms. Next, a sample SDF is computed for each data window after which all SDFs are averaged for the final result.

(2) "Averaging Covs": First, the data is partitioned into subwindows containing the number of data points indicated under "Window width". The covariance function of the data in each subwindow is computed. Next, they are averaged

over all windows and tapered with the window indicated under "Window Type", and transformed back to the frequency domain using FFT.

(3) "Auto Regressive": Instead of classical FFT-based techniques, an Auto Regressive model (AR) is used to estimate the SDF. In this case, the window type, overlap and window width have no effect.

The impulse response at the output is computed as follows. The calculated ETFE is weighted with the frequency weight, which consists of 2 frequency break points and 1 attenuation level. The weight is equal to 1 between the minimum and maximum frequency breakpoint, and equal to minus the attenuation (in dB) outside this frequency interval. After the weighting, the inverse FFT is computed to obtain the impulse response. The weighting is useful to suppress high or low frequency noise.

For more details, see the ISID Part 1 manual.

Xmath Functions

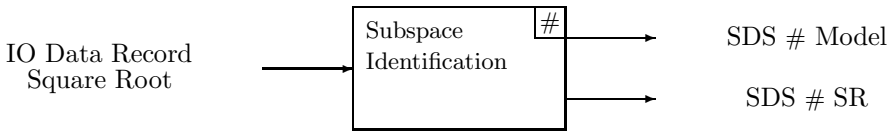
etfe(), sdf(), fft(), taper(), spectrum()

Subspace Identification (SDS)

Description

Use subspace identification to compute a state space innovations model from raw input-output data or from an intermediate square root.

The identified model and calculated square root are available at the output.



Inputs

IO Data Record or Square Root

Outputs

Model :	SDS # Model
Square Root :	SDS # SR

Parameters

Order	The order of the identified model. This could be a number or a vector of numbers (the output consists then of multiple models).

Defaults

Max. Order	Maximal order of the model that can be identified. This parameter indirectly determines the number of block rows in the block Hankel matrices.
Bias	Determines if the identified model is asymptotically unbiased or not. Default: No Asymptotic Bias.
Basis	Determines the Basis in which the model is calculated. When using angles, the resulting model is independent of scaling. Default: Singular Values.
Scaling	Scales the inputs and outputs so they have a standard deviation of 1 (Equal Energy). Default: No Scaling.

Plot

The plot shows either angles or singular values (depending on the Basis). These values help to decide the order of the identified model(s). The last significant singular value (different from zero) or the last angle significantly different from 90 degrees determines the order.

The order of the identified model can be altered by clicking with the left mouse on the bar corresponding to the required order. To obtain more than one order, click and drag.

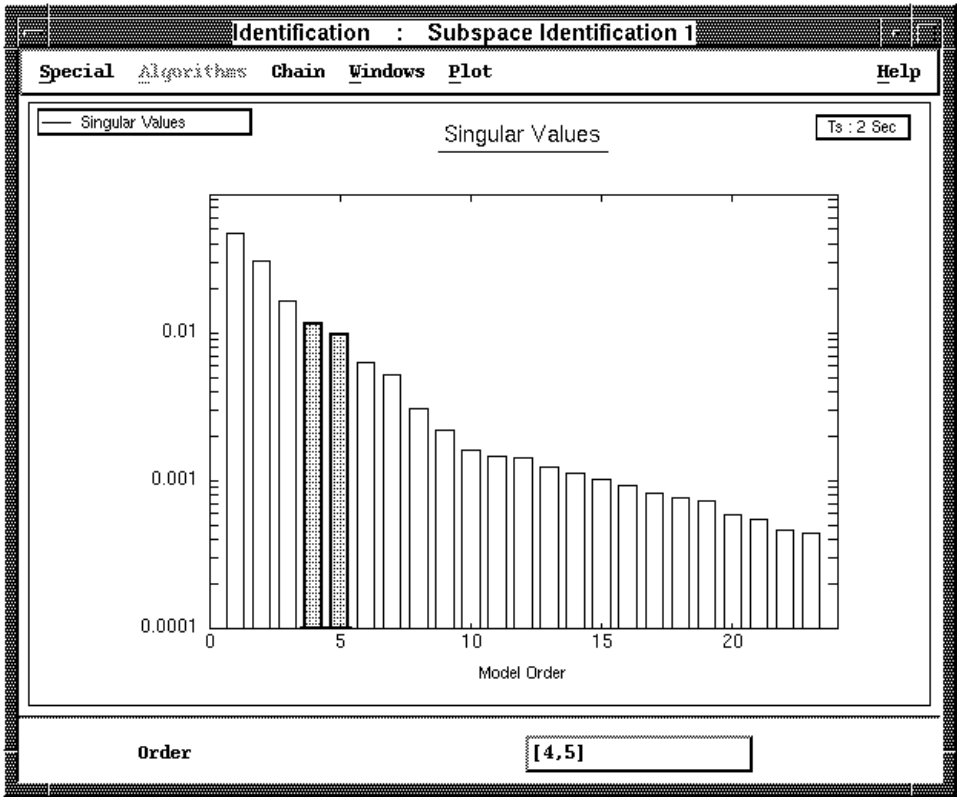
Algorithm

The algorithm first projects the "future" outputs onto the "past" and "future" inputs and the "past" outputs. From this projection, the Kalman filter states are determined (without prior knowledge of the system). The system is obtained from these states via a least-squares solution.

For more details, see the ISID Part 1 manual.

Xmath Functions

sds()



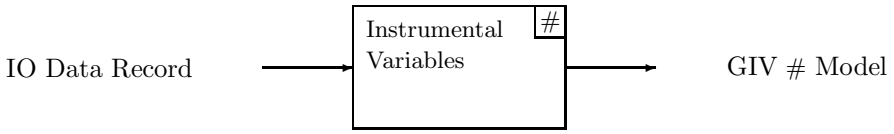
The Subspace Identification Window.

INSTRUMENTAL VARIABLES (GIV)

Description

Use the Instrumental Variables method to identify a model from input-output data. The inputs are used as "instruments".

The identified model is available at the output.



Inputs

IO Data Record

Outputs

Model :	GIV # Model

Parameters

Model Order	Model order of the identified model. This could be a number or a vector of numbers.

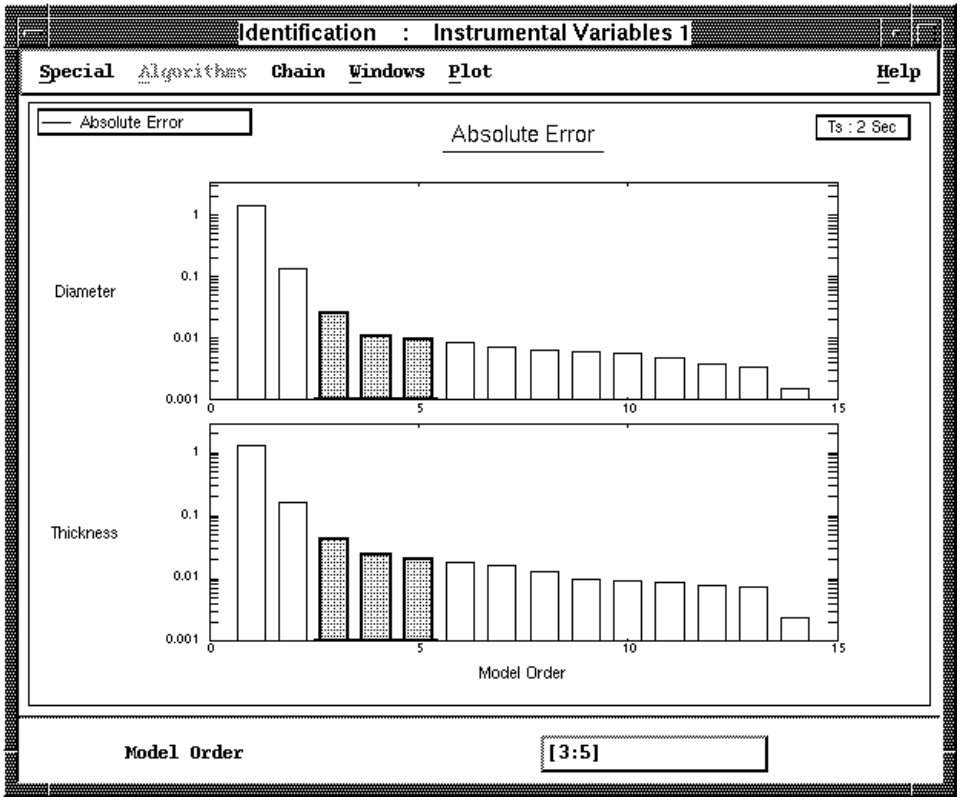
Defaults

#Lags Past	Number of instrumental variable lags in the past.
#Lags Future	Number of instrumental variable lags in the future.
Feedthrough	Identify a feedthrough term or not. Default: Feedthrough.

Plot

The plot shows diagonal terms of the prediction error variance matrix. These values indicate the order of the identified model(s). Normally, the error norms are (almost) constant above a certain order. The smallest order attaining this level is a good choice.

To alter the order of the identified model click on the bar corresponding to the required order. To select more than one order, click and drag.



The Instrumental Variables Window.

Algorithm

The instrumental variable method computes an ARMA model (no noise term, only an input-output model) by assuming the noise at the output to be uncorrelated with the input signal.

For more details, see the ISID Part 1 manual.

Xmath Functions

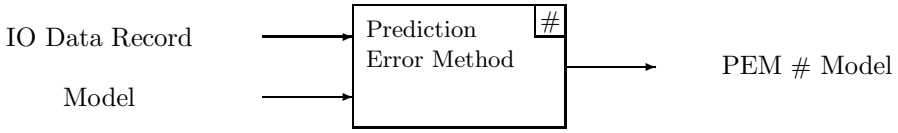
giv()

PREDICTION ERROR METHOD (PEM)

Description

Use the Prediction Error Method to identify a state space innovations model from input-output data, starting from a user-supplied model.

The identified model is available at the output.



Inputs

IO Data Record and Model

Outputs

Model :	PEM # Model

Parameters

Show On Plot	Plot the norm of the error signal or the frequency response or impulse response of the calculated models. Default: Error Norm Evolution.
Model At Input	Use the full model at the input as an initial model or use only the input-output part of the initial model (in this case, the noise model is computed internally). Default: Use As Initial.

Defaults

Order Stoch. Sys	When the stochastic model is calculated internally, this indicates the order of the stochastic part.
#Iterations	Number of large iterations.
Gauss Newton Step	Initial relative step size for the Gauss-Newton iterations.
#GN Iterations	Number of Gauss-Newton iterations.

Steep Des. Step	Initial step size for the steepest descent iterations, relative in norm to the initial parameter vector.
#SD Iterations	Number of steepest descent iterations.
Accuracy	The iterations will abort as soon as the relative improvement of the prediction error criterion between two subsequent iterations is less than (100-Accuracy) percent.
Canonical Form	Canonical form type selected by initmodel: Observability, Controllability, Minimum Norm (Obs. or Contr. depending on which has the minimal norm), Minimal Parameters (Obs. or Contr. depending on which has the minimum number of parameters). Default: Observability.

Plot

When Show On Plot is equal to "Error Norm Evolution", the error norms of each of the "Number of Iteration" models is shown. If this option is set before calculation, the error norms of the "inner" Gauss Newton and steepest descent iterations are plotted (together with the error norm of the previous outer iteration as a horizontal line). When the improvement of the error norm is less than (100-Accuracy) percent, the computation is stopped and the error norms of each of the "Number of Iteration" models are shown.

When Show On Plot is equal to "Frequency Response" or "Impulse Response", the frequency/impulse response of each of the "Number of Iteration" models is shown.

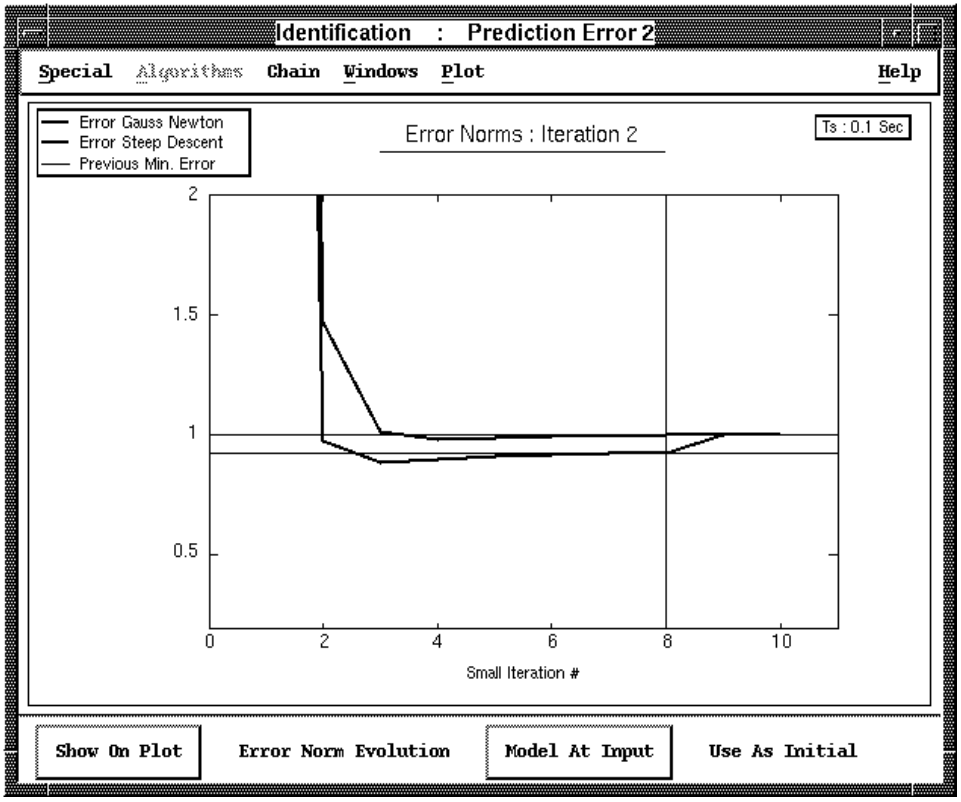
Algorithm

Starting from the initial model, the Gauss Newton direction and steepest descent direction are determined. A "Gauss Newton Step" and "steepest descent step" is taken in the respective direction. The inner iterations are performed on the segment between the initial model and the stepped model. From these models, the one with the lowest prediction error is retained to serve as an initial model for the next iteration.

For more details, see the ISID Part 1 manual.

Xmath Functions

pem()



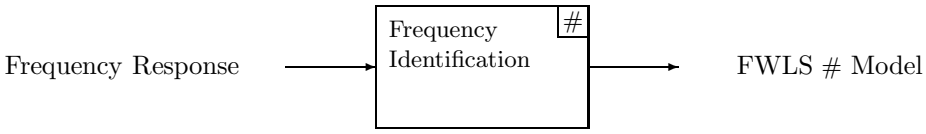
The Prediction Error Method Window.

FREQUENCY IDENTIFICATION (FWLS)

Description

Use the Frequency-Weighted Least Squares to identify an innovations model from frequency response data. With an optional weighting function, certain frequency bands can be weighed.

The identified model is available at the output.



Inputs

Frequency Response

Outputs

Model :	FWLS # Model

Parameters

Order Den. (A)	ARX model order of the Denominator. Changing this order automatically updates the ARX model order of the numerator. Order Den. can be a number or a vector of numbers.
Order Num. (B)	ARX model order of the numerator. Order Num. can be a number or a vector of numbers.

Defaults

Max. Order	The maximal order or maximal number of regression lags for the calculation of the least squares models. All models with an order smaller than Max. Order can be obtained.
Feedthrough	Create a Feedthrough term in the model or not. Default: Feedthrough.

Show On Plot	Choice between the absolute error and the relative error (the standard deviation of the error divided by the energy in the original signal). Note that these errors do not detect an unstable system i.e. they will still be finite for unstable systems. A third choice consists of viewing the original frequency magnitude together with the estimated magnitude and the frequency weight (see Help on Plot below). Default: Absolute Error.
Min. Freq. Weight	The minimum break frequency of the weight.
Max. Freq. Weight	The maximum break frequency of the weight.
Attenuation Weight	The attenuation of the weight.

Plot

The plot shows either the absolute or relative norm of the diagonal terms of the prediction error variance matrix (depending on Show On Plot). These values indicate the order of the identified model(s). Normally, the error norms are (almost) constant above a certain order. The smallest order that attains this level is a good choice.

To alter the order of the identified model, click with the left mouse on the bar corresponding to the required order. To obtain more than one order, click and drag.

When Show On Plot is equal to "Magnitude + Weights", the plot shows the original frequency response (the input to the algorithm) together with the frequency weight and the identified responses. One can interact with this frequency weight by clicking and dragging near one of the corners and/or lines.

Algorithm

Solves a least squares problem in the frequency domain (trying to fit the frequencies as well as possible in a least squares sense).

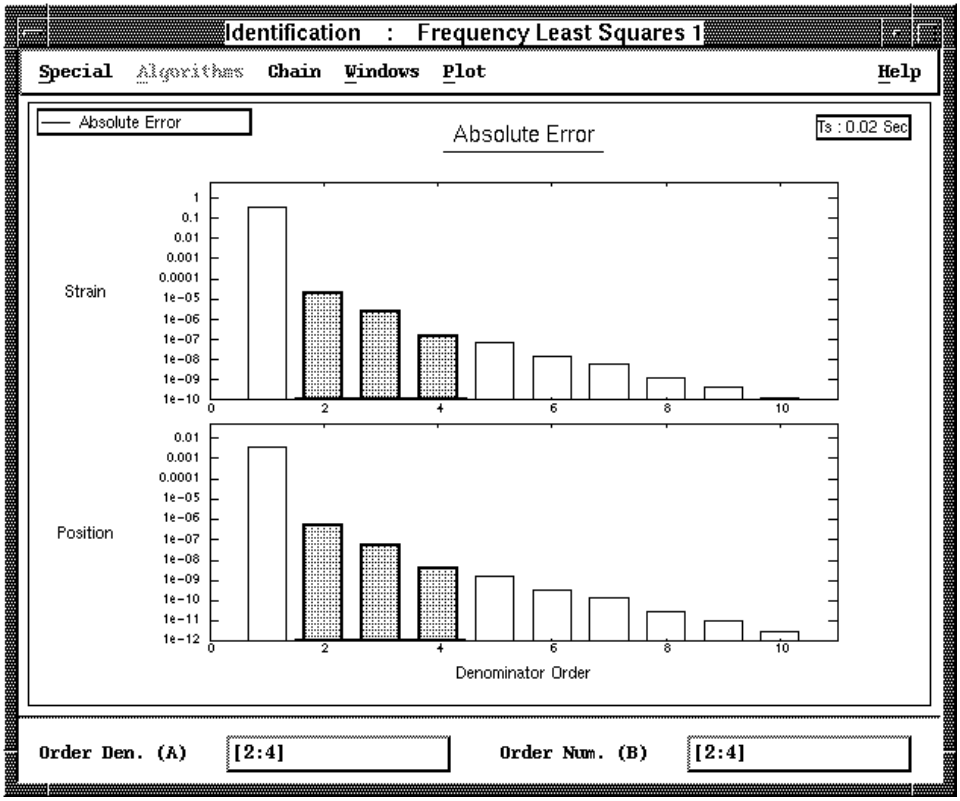
The accuracy of the model in certain frequency regions can be influenced with the weight function.

FWLS is typically used in one of the following three cases: 1) ETFE - FWLS , 2) LS (high order) - FREQ - FWLS, or 3) frequency data box (measured response) - FWLS. The second possibility allows for the reduction of high order models.

For more details, see the ISID Part 1 manual.

Xmath Functions

fwls()



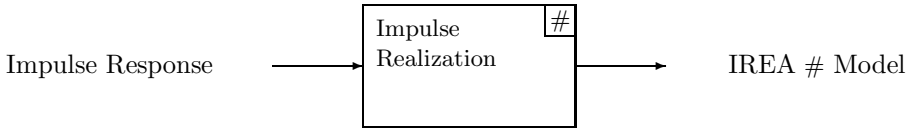
The Frequency Identification Window.

IMPULSE REALIZATION (IREA)

Description

Use the Impulse Realization method to identify an ARMA model from impulse response data.

The identified model is available at the output.



Inputs

Impulse Response

Outputs

Model :	IREA # Model

Parameters

Model Order	A number or vector of numbers indicating the state space model order.
Show On Plot	Choice between the singular values and the impulse response (see Plot below). Default: Singular Values.

Defaults

Max. Order	Maximal possible order. This indirectly determines the number of block rows and columns in the block Hankel matrices.
Method	Choice between Zeiger / McEwen and Kung / Kailath. Default: Zeiger / McEwen.

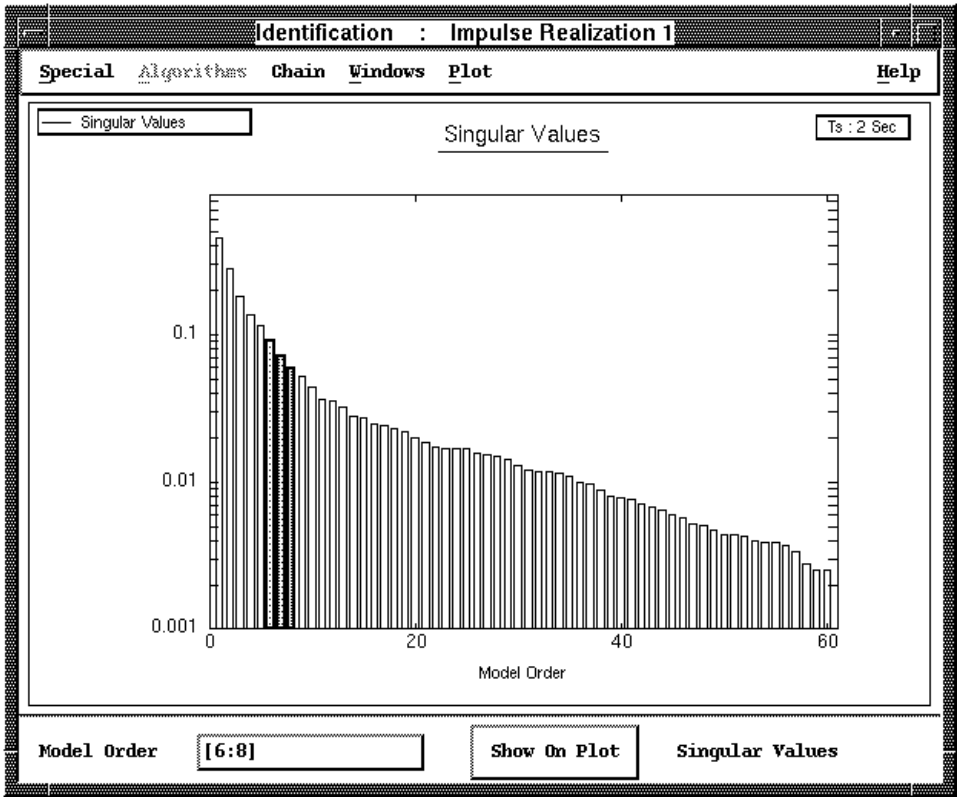
Plot

When Show On Plot indicates Singular Values, the singular values of the Hankel matrix are plotted. These values help decide the order of the identified model(s). The last significant singular value determines the order.

To alter the order of the identified model click on the bar corresponding to the required order. To obtain more than one

order, click and drag.

When Show On Plot is equal to "Impulse Responses", the plot shows the original impulse response (the input to the algorithm), together with the identified responses.



The Impulse Realization Window.

Algorithm

Hankel matrices are constructed from the impulse response. The order is decided from the Hankel singular values. From the singular value decomposition of the Hankel matrix, the model is identified (using the selected "Method").

This function is typically used in one of the three cases: 1) ETFE - ERA, 2) BLS (high order) - IMP - ERA, or 3) impulse data box (Measured response) - ERA. The second possibility allows for the reduction of high order models.

For more details, see the ISID Part 1 manual.

Xmath Functions

irea()

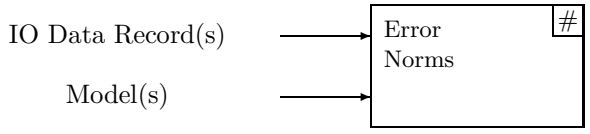
4.4 Validation

Error Norms (ERNO)

Description

Displays Simulation and Prediction Error Norms of identified models applied to different input-output data records.

This is a useful tool for fast validation of identified models.



Inputs

IO Data Record(s) and Model(s)

Outputs

None

Parameters

Mode	Simulation (using input-output model and past inputs) or prediction (using input-output model/noise model and past inputs/outputs). Default: Simulation.
Pred. Horizon	When Mode is Prediction, the output is predicted "Prediction Horizon" steps ahead.

Defaults

Compare	See Plot below. Default: Models.
Start Sample	The starting sample that is used to calculate the error norms. This can be set different from 1 (default) when transient effects are to be neglected.

Least Squares The Least Squares mode can only be turned on when there is one Least Squares model at the input (and one input-output data record). In this case, the plot shows the least squares validation errors of ALL the models of the Least Squares algorithm. Default: Mode Off.

Plot

The plot consists of bar graphs representing the error in percentage. These errors are calculated by dividing the standard deviation of the simulation or prediction error by the standard deviation of the corresponding output.

The organization of the bar graphs can be done in 3 different ways (depending on the Compare variable):

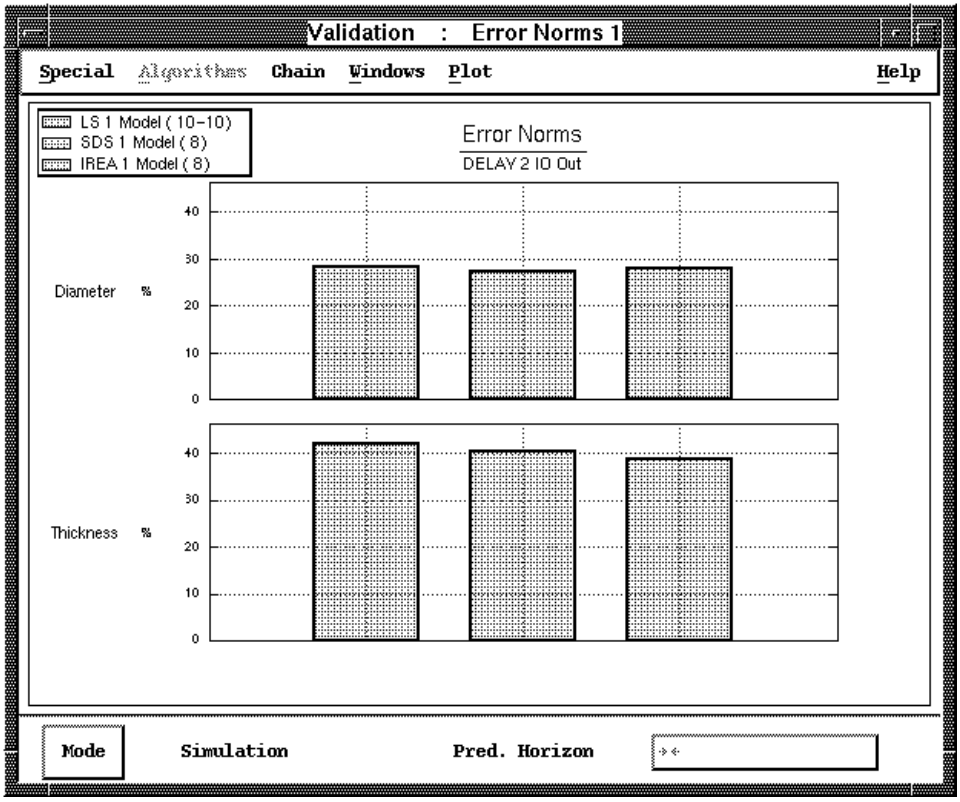
1) models: Columns = data , Rows = outputs , Compare On Plot = models

2) data: Columns = models , Rows = outputs , Compare On Plot = data

3) outputs: Columns = data , Rows = models , Compare On Plot = outputs

Xmath Functions

idsim()



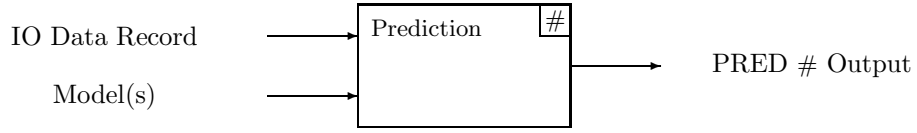
The Error Norms Window.

PREDICTION (PRED)

Description

Displays the simulation or the prediction signal obtained from applying the identified models to the input-output data record.

The simulated/predicted output data is available at the output.



Inputs

IO Data Record and Model(s)

Outputs

IO Data Record : PRED # Output

Parameters

Mode	Simulation (using input-output model and past inputs) or Prediction (using input-output model/noise model and past inputs/outputs). Default: Simulation.
Pred. Horizon	When Mode is Prediction, the output is predicted "Prediction Horizon" steps ahead.

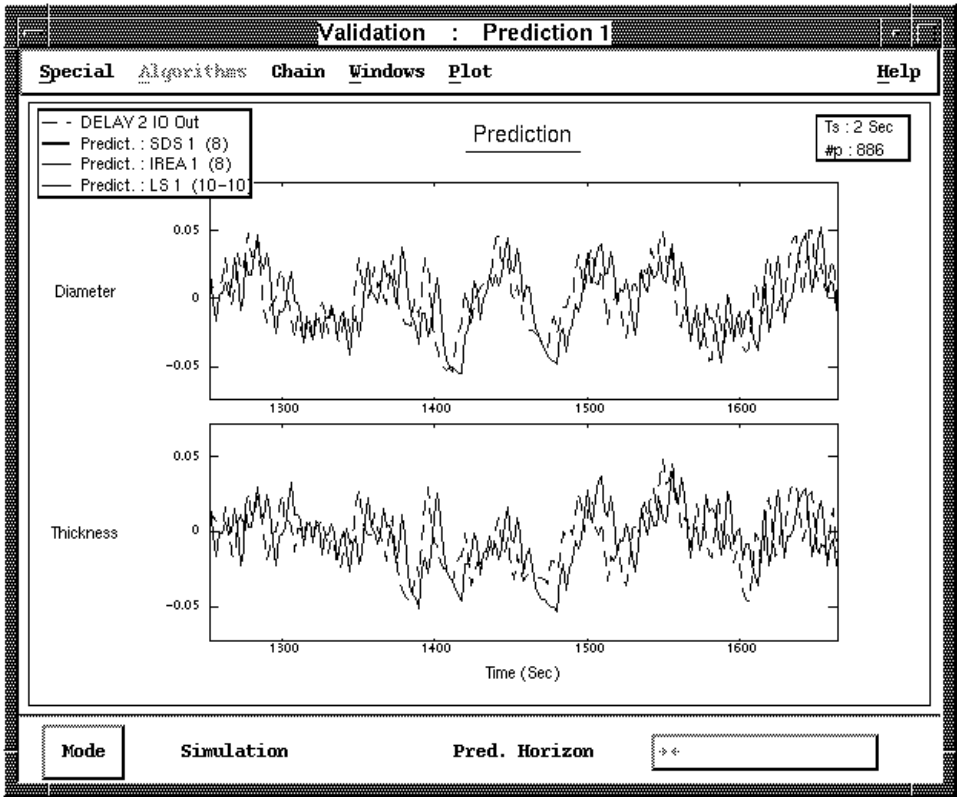
Defaults

None

Plot

The plot shows the original and predicted data. All models at the input are applied to the input-output data record. Each combination leads to a simulated/predicted signal. Even though the legend displays all models, there is only one predicted signal plotted (to not overload the plot). To switch between the different signals click on the legend entry (the text field of the legend containing the model name).

Because the output signal of this algorithm is a prediction error it has only outputs (no inputs).



The Prediction Window.

Xmath Functions

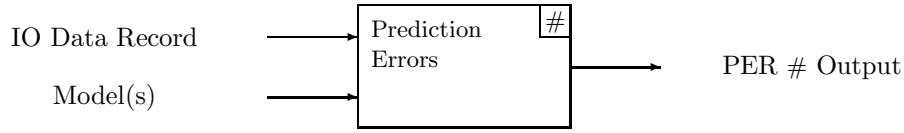
idsim()

PREDICTION ERRORS (PER)

Description

Displays the simulated or prediction error signal obtained from applying the identified models to the input-output data record.

The simulation/prediction error is available at the output.



Inputs

IO Data Record and Model(s)

Outputs

IO Data Record : PER # Output

Parameters

Mode Simulation (using input-output model and past inputs) or Prediction (using input-output model/noise model and past inputs/outputs). Default: Simulation.

Pred. Horizon When Mode is Prediction, the output is predicted "Prediction Horizon" steps ahead.

Defaults

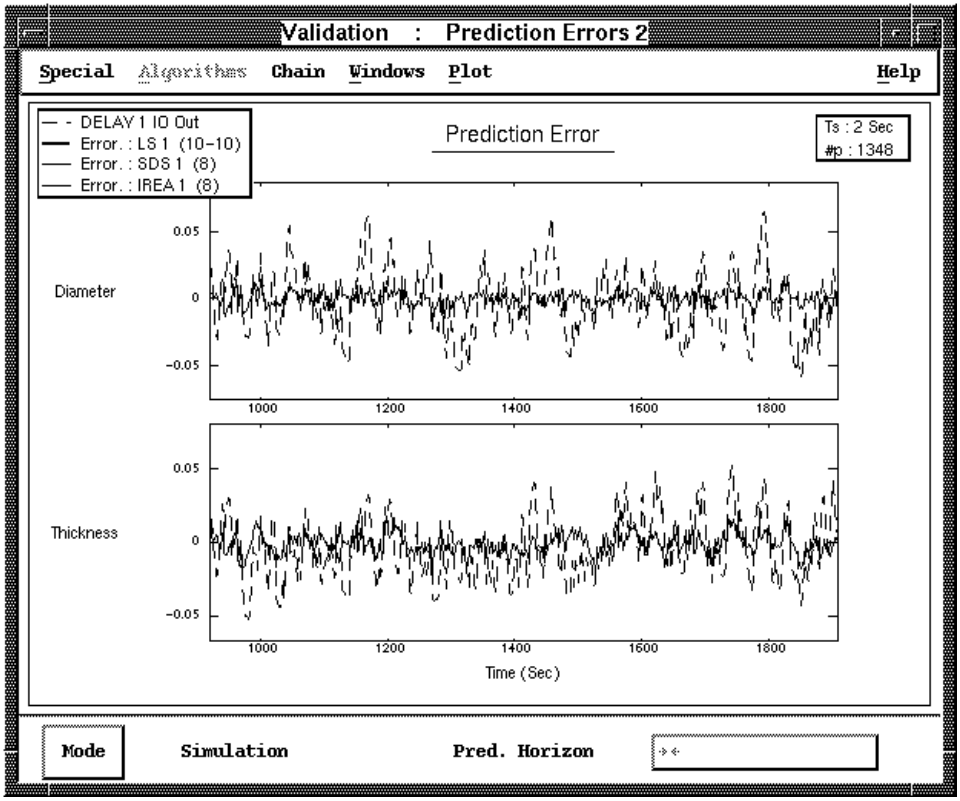
None

Plot

The plot shows the original data and the prediction error. All models at the input are applied to the input-output data record. Each combination leads to a simulated/predicted error signal. Even though the legend displays all models, there is only one prediction error plotted (to not overload the plot). To switch

between the different signals, click on the legend entry (the text field of the legend containing the model name).

Because the output signal of this algorithm is a prediction error it has only outputs (no inputs).



The Prediction Errors Window.

Xmath Functions

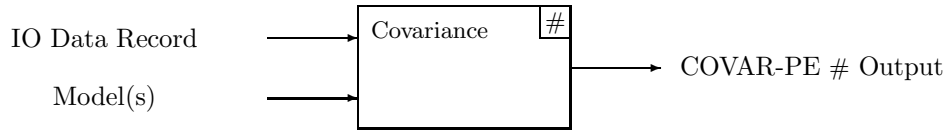
idsim()

COVARIANCE (COVAR-PE)

Description

Displays the covariance sequence of the simulation or prediction errors. These errors are obtained by applying the identified models to the input-output record. The confidence levels are also displayed.

The plotted covariance is available at the output.



Inputs

IO Data Record and Model(s)

Outputs

Impulse Response : COVAR-PE # Output

Parameters

Mode	Simulation (using input-output model and past inputs) or prediction (using input-output model/noise model and past inputs/outputs). Default: Prediction.
Pred. Horizon	When Mode is Prediction, the output is predicted "Prediction Horizon" steps ahead.

Defaults

#Lags	Number of lags of the covariance response.
Start Sample	From this sample on, the errors are used to calculate the covariance. This parameter can be set different from 1 (default) when transient effects are to be neglected.
Show Covar. Model	Show or suppress the covariance model (of the model at the input). See also plot below. Default: No.

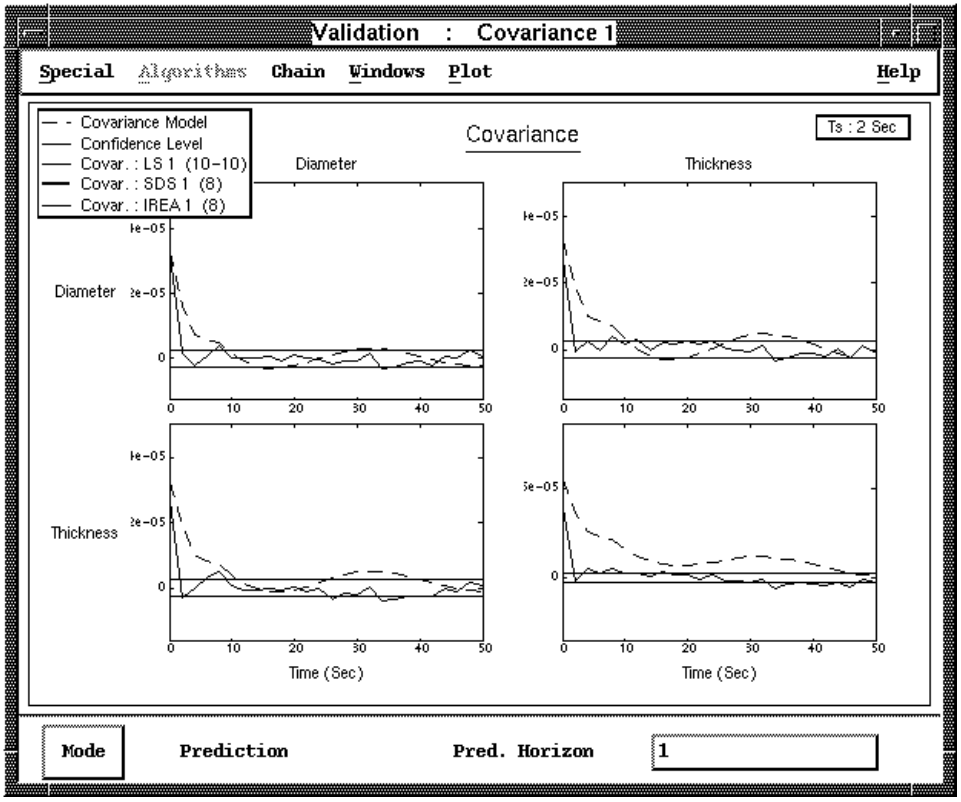
Plot

The plot shows the covariance response calculated from the prediction or simulation errors, together with the (possible) confidence levels and covariance responses of the model. All models at the input are applied to the input-output data record. Each combination leads to a predicted signal and thus to a covariance response. Even though the legend displays all models, there is only one covariance plotted so the plots won't be overloaded. To switch between different signals click on the legend entry (the text field of the legend containing the model name). When the model is good, the covariance function of the model and the covariance function of the simulation errors are close. The covariance function of the prediction errors should be an impulse.

The confidence levels are 95 percent levels - which implies that 5 percent of the covariance can lie outside the levels. Note that the confidence levels are only valid for prediction and NOT for simulation.

Xmath Functions

idsim(), get_corr()



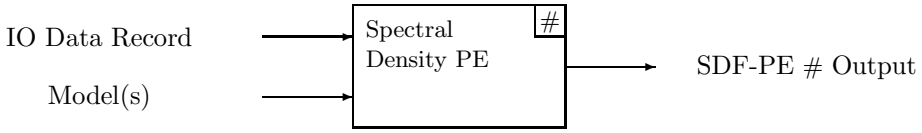
The Covariance Window.

Spectral Density PE (SDF-PE)

Description

Displays the spectral density function of the simulation or prediction errors. These errors are obtained from applying the identified model to the input-output record.

The spectral density is available at the output.



Inputs

IO Data Record and Model(s)

Outputs

Frequency Response : SDF-PE # Output

Parameters

Mode	Simulation (using input-output model and past inputs) or Prediction (using input-output model/noise model and past inputs/outputs). Default: Simulation.
Pred. Horizon	When Mode is Prediction, the output is predicted "Prediction Horizon" steps ahead.

Defaults

Window Width	Window width used to calculate the SDF (see also ETFE).
Start Sample	From this sample on, the errors are used to calculate the spectral density. This parameter can be set different from 1 (default) when transient effects are to be neglected.

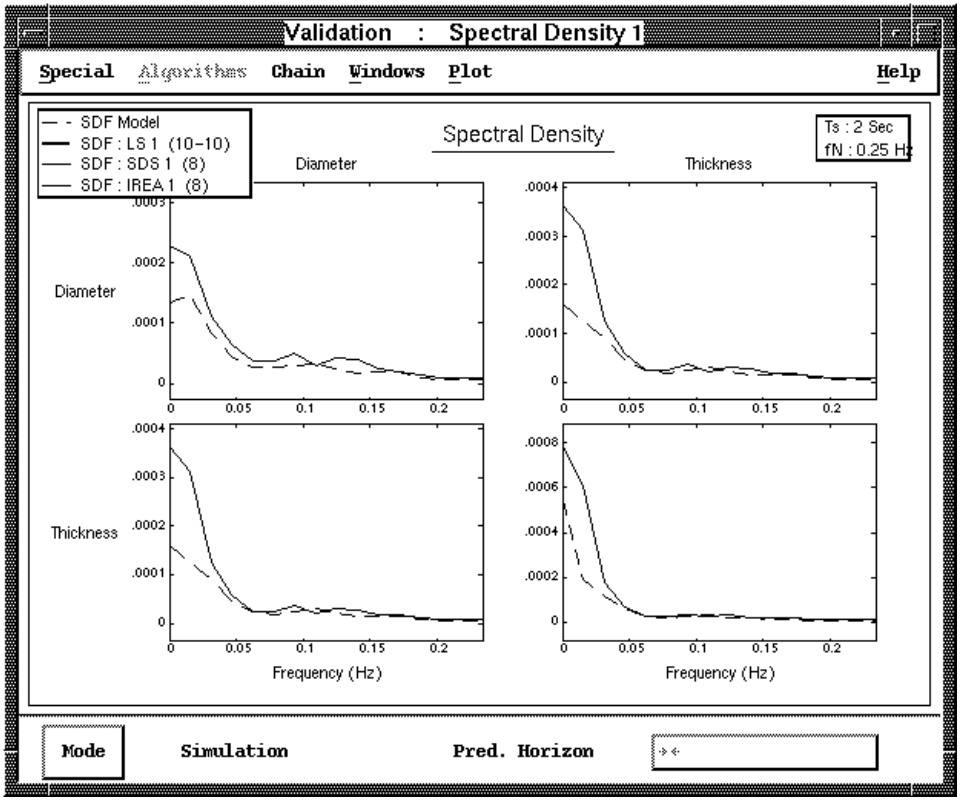
Plot

The plot shows the spectral density function calculated from the prediction or simulation errors. All models at the input are applied to the input-output data record. Each combination leads to a predicted signal and thus to a spectral density. Even though the legend displays all signals, there is only one spectral density plotted so the plots won't be overloaded. To switch between the different signals click on the legend entry (the text field of the legend containing the model name). The spectral density function of the model (the noise part of the model) is shown in the background.

For a good model, the spectral density of the model and of the simulation errors should be close. The spectral density of the prediction errors should be flat (equal to one on the diagonals and zero on the off-diagonals).

Xmath Functions

idsim(), sdf()



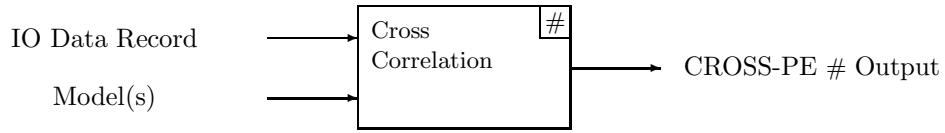
The Spectral Density PE Window.

CROSS CORRELATION (CROSS-PE)

Description

Displays the cross correlation between the simulation or the prediction errors and the input signal. These errors are obtained by applying the identified models to the input-output record. The confidence levels are also displayed.

The correlation is available at the output.



Inputs

IO Data Record and Model(s)

Outputs

Impulse Response : CROSS-PE # Output

Parameters

Mode	Simulation (using input-output model and past inputs) or Prediction (using input-output model/noise model and past inputs/outputs). Default: Prediction.
Pred. Horizon	When Mode is Prediction, the output is predicted "Prediction Horizon" steps ahead.

Defaults

#Lags	Number of lags of the cross correlation sequence.
Start Sample	From this sample on, the errors are used to calculate the cross correlation. This parameter can be set different from 1 (default) when transient effects are to be neglected.

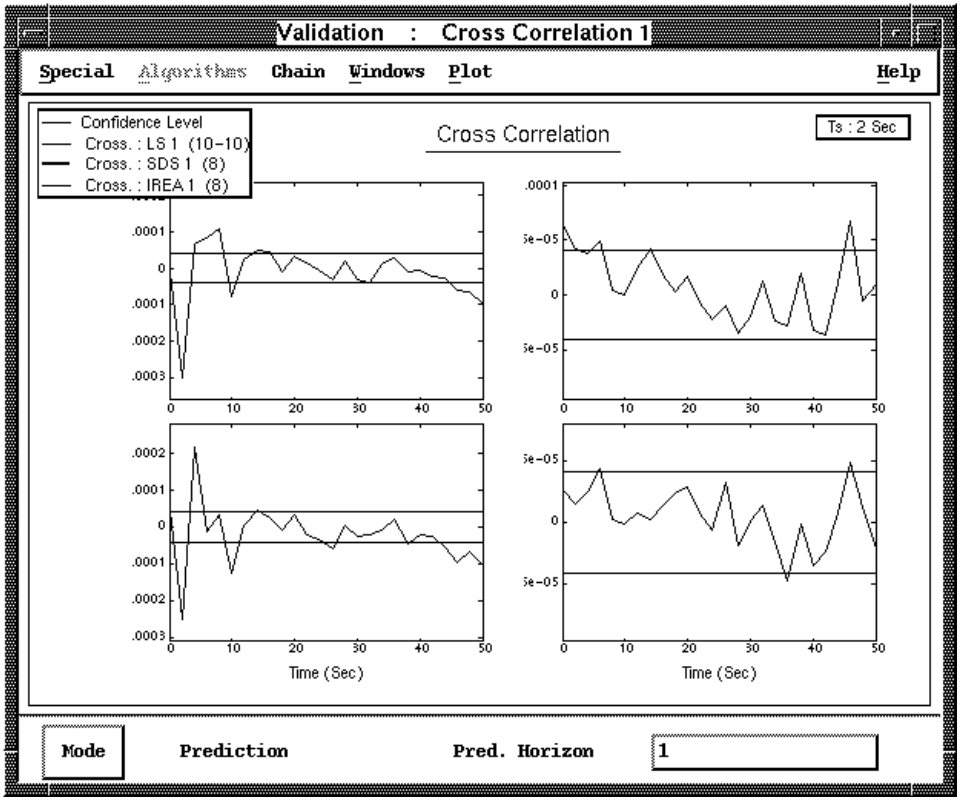
Plot

The plot shows the Cross Correlation responses calculated from the prediction or simulation errors, together with the confidence levels. All models at the input are applied to the input-output data record. Each combination leads to a predicted signal and thus to a Cross Correlation. Even though the legend displays all models, there is only one correlation plotted so the plots won't be overloaded. To switch between the different signals click on the legend entry (the text field of the legend containing the model name).

The confidence levels are 95 percent levels - which implies that 5 percent of the covariance can lie outside the levels. Note that the confidence levels are only valid for prediction and NOT for simulation.

Xmath Functions

idsim(), get_corr()



The Cross Correlation Window.

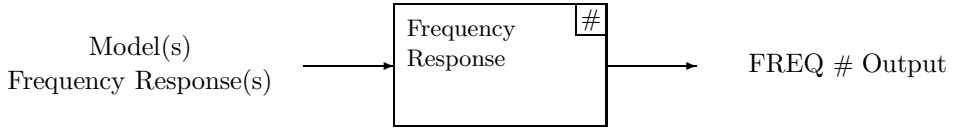
4.5 Display

Frequency Response (FREQ)

Description

Displays the transfer function of the model(s) at the input. The frequency response data records at the input are also displayed.

The plotted frequency response is available at the output. When there is more than one frequency response plotted, the highlighted frequency response is available at the output.



Inputs

Model(s) and/or Frequency Response(s)

Outputs

Frequency Response : FREQ # Output

Parameters

Show On Plot	Magnitude or Phase. Default: Magnitude.

Defaults

min. freq.	Minimum frequency of the axis used to determine the frequency response of the models.
max. freq.	Maximum frequency of the axis used to determine the frequency response of the models.
# points	Number of frequency axis points used to determine the frequency response of the models.

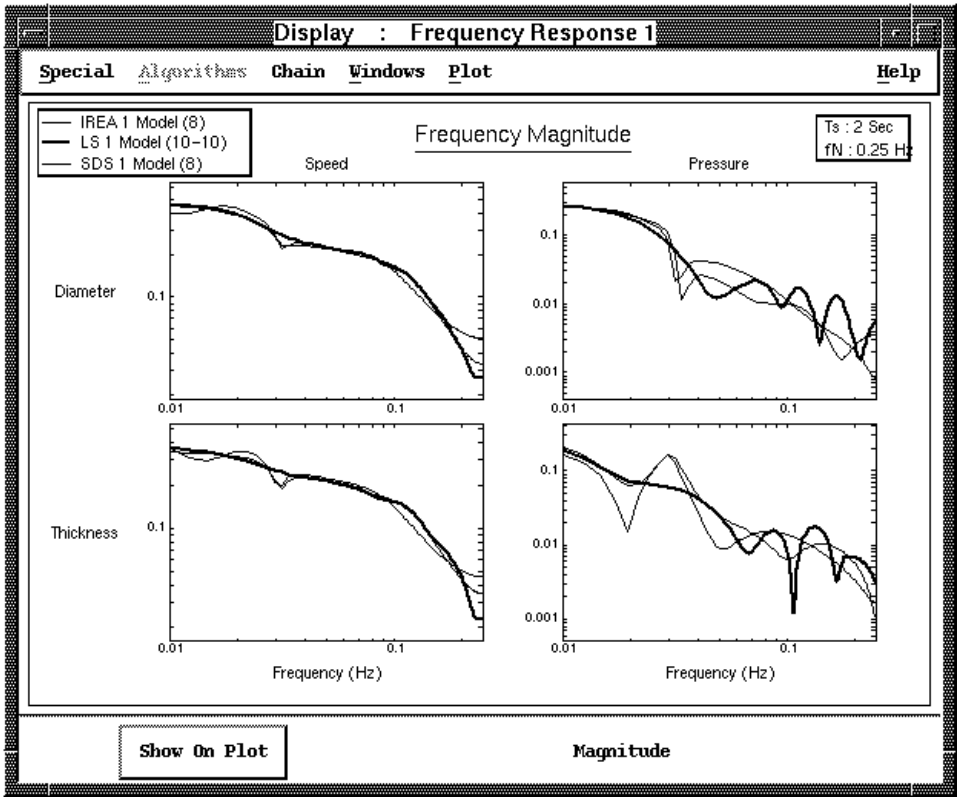
Plot

The plot shows the frequency responses calculated from the models (using the defined frequency axis). The (possible) frequency response data records available at the input are also plotted. In this way it is easy to compare frequency responses of different identified models (different identification algorithms). Frequency response data (from ETFE or from a frequency data box) can also be compared.

The output of the algorithm is determined as follows: if there is only one response plotted, than this response is available at the output. If there are more responses plotted, the highlighted response (legend entry and plot are highlighted) is available at the output. To change the highlighted signal select a plotted response with the left mouse or select the corresponding legend entry (the text entry indicating the object name).

Xmath Functions

idfreq()



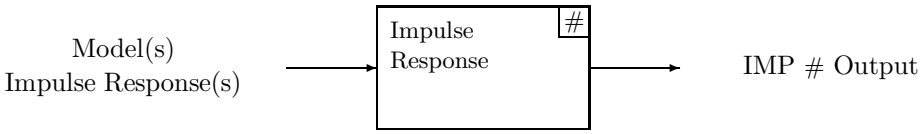
The Frequency Response Window.

IMPULSE RESPONSE (IMP)

Description

Displays the impulse response of the model(s) at the input. The impulse response data records at the input are also displayed.

The plotted impulse response is available at the output. When there is more than one impulse response plotted, the highlighted response is available at the output.



Inputs

Model(s) and/or Impulse Response(s)

Outputs

Impulse Response : IMP # Output

Parameters

None

Defaults

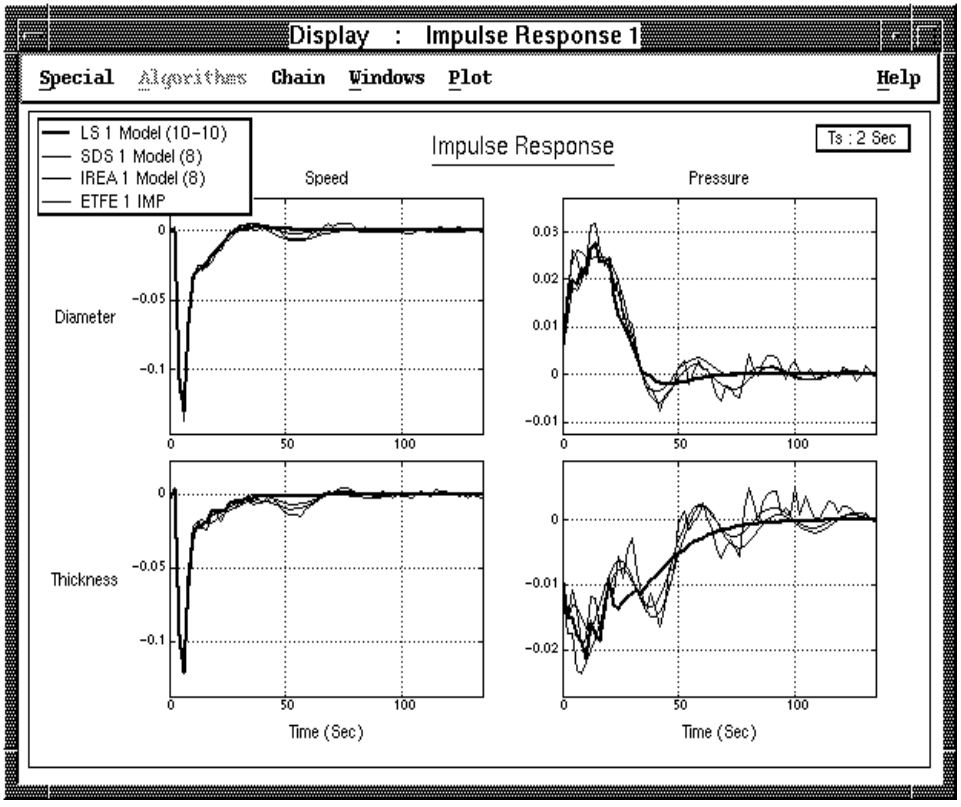
# points	Number of time axis points used to determine the impulse response of the models.

Plot

The plot shows the impulse responses calculated from the models (using the defined time axis). The (possible) impulse response data records available at the input are also plotted. In this way it is easy to compare impulse responses of different identified models (different identification algorithms). Impulse response data (from ETFE for instance) can also be compared.

The output of the algorithm is determined as follows: if there is only one response plotted, than this response is available at the output. If there are more responses

plotted, the highlighted response (legend entry and plot are highlighted) is available at the output. To change the highlighted signal select a plotted response with the left mouse or select the corresponding legend entry (the text entry indicating the object name).



The Impulse Response Window.

Xmath Functions

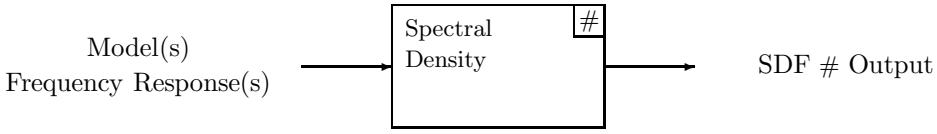
idimpulse()

SPECTRAL DENSITY (SDF)

Description

Displays the spectral density function (SDF) of the model(s) at the input. The frequency response data records at the input are also displayed.

The plotted SDF is available at the output. When there is more than one SDF plotted, the highlighted SDF is available at the output.



Inputs

Model(s) and/or Frequency Response(s)

Outputs

Frequency Response : SDF # Output

Parameters

Show On Plot	Magnitude or Phase. Default: Magnitude.

Defaults

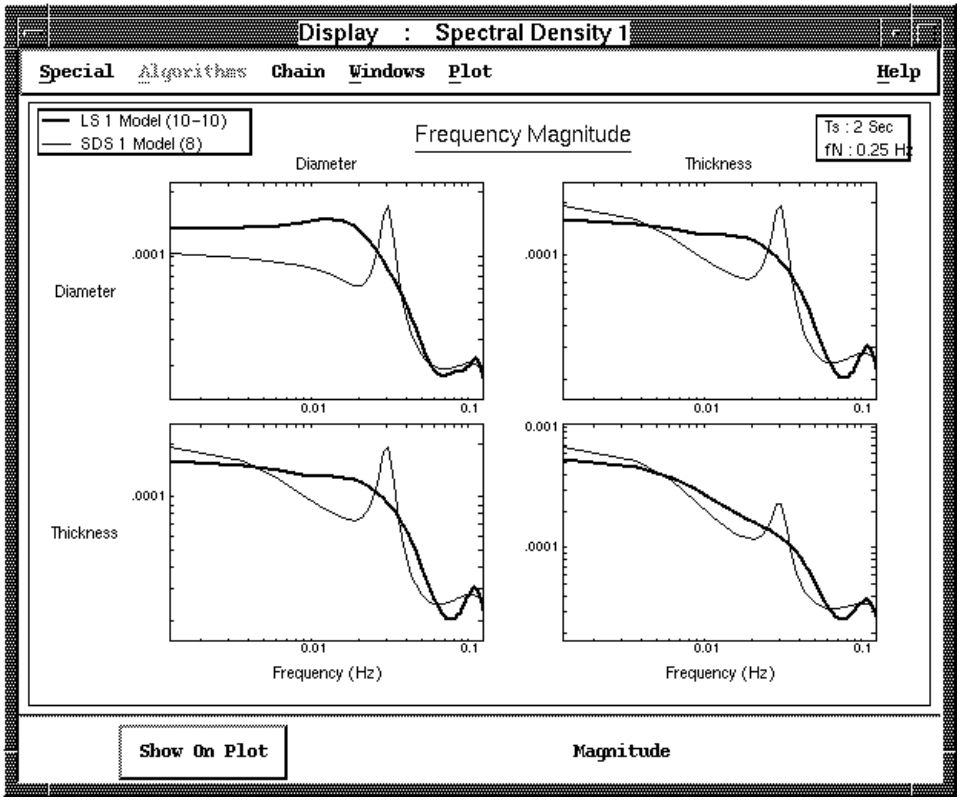
min. freq.	Minimum frequency of the axis used to determine the spectral density of the models.
max. freq.	Maximum frequency of the axis used to determine the spectral density of the models.
# points	Number of frequency axis points used to determine the spectral density of the models.

Plot

The plot shows the spectral densities calculated from the models (using the defined frequency axis). The (possible) frequency response data records available at

the input are also plotted. In this way it is easy to compare spectral density functions of different identified models (different identification algorithms). Frequency response data can also be compared.

The output of the algorithm is determined as follows: if there is only one SDF plotted, than this SDF is available at the output. If there are more responses plotted, the highlighted SDF (legend entry and plot are highlighted) is available at the output. To change the highlighted signal select a plotted response with the left mouse or select the corresponding legend entry (the text entry indicating the object name).



The Spectral Density Window.

Xmath Functions

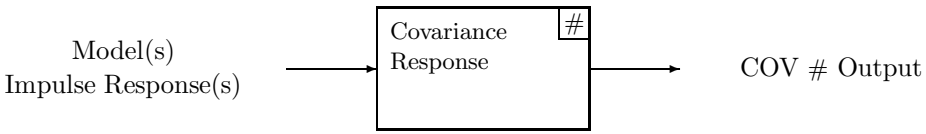
idfreq()

COVARIANCE RESPONSE (COV)

Description

Displays the covariance response of the model(s) at the input. The impulse response data records at the input are also displayed.

The plotted covariance response is available at the output. When there is more than one covariance response plotted, the highlighted covariance response is available at the output.



Inputs

Model(s) and/or Impulse Response(s)

Outputs

Impulse Response : COV # Output

Parameters

None

Defaults

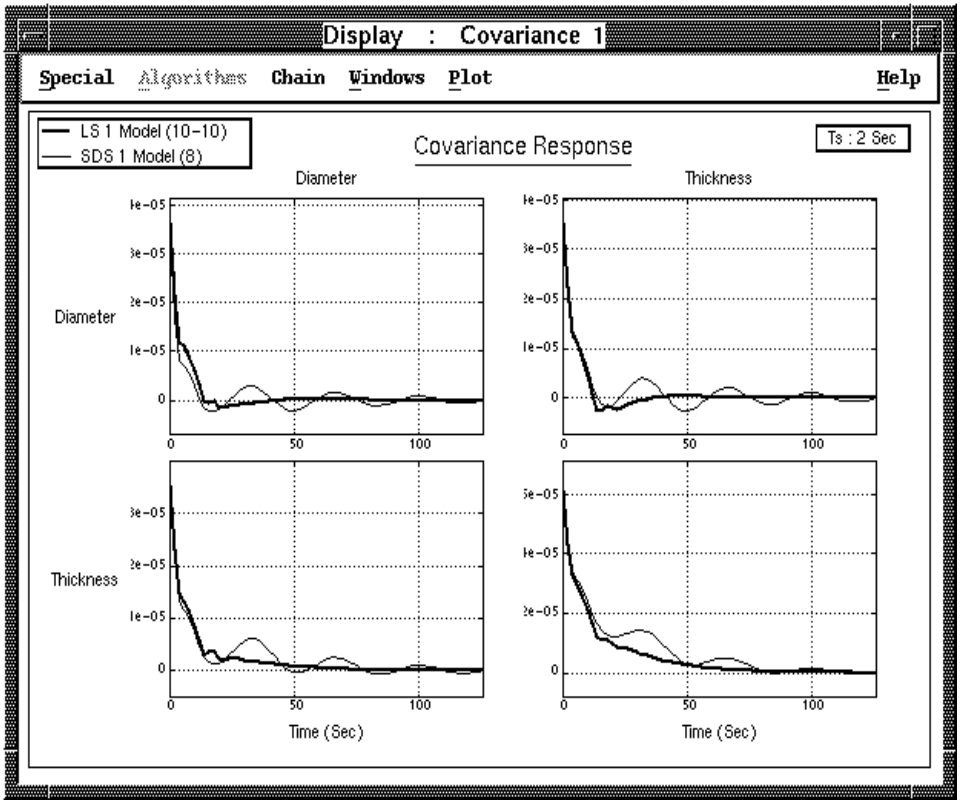
 # points Number of time axis samples used to determine the covariance response of the models.

Plot

The plot shows the covariance responses calculated from the models (using the defined time axis). The (possible) impulse response data records available at the input are also plotted. In this way it is easy to compare covariance responses of different identified models (different identification algorithms). Impulse response data can also be compared.

The output of the algorithm is determined as follows: if there is only one covariance response plotted, than this covariance response is available at the output. If there

are more responses plotted, the highlighted covariance response (legend entry and plot are highlighted) is available at the output. To change the highlighted signal select a plotted response with the left mouse or select the corresponding legend entry (the text entry indicating the object name).



The Covariance Response Window.

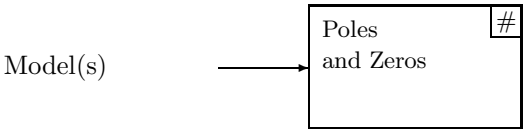
Xmath Functions

idimpulse()

POLES AND ZEROS (PZ)

Description

Displays the Transmission poles and zeros of the model(s) at the input.



Inputs

Model(s)

Outputs

None

Parameters

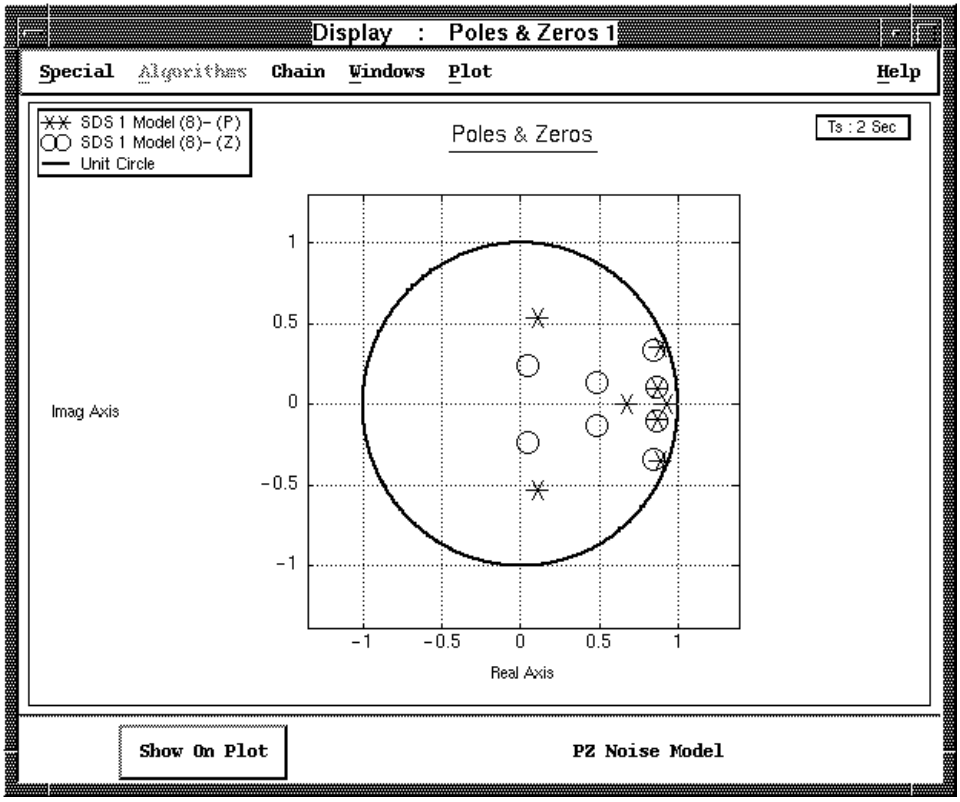
Show On Plot	Choice between Poles and Zeros of the input-output model or of the Noise model. Default: PZ IO Model.

Defaults

None

Plot

The plot shows the poles and transmission zeros of the models at the input (together with the unit circle). Note that when the number of inputs is different from the number of outputs, there are no transmission zeros.



The Poles and Zeros Window.

Xmath Functions

idpolezero()

Chapter 5

Case Studies

This chapter describes several case studies, which are intended to illustrate ISID's extensive capabilities. Each case study consists of a default chain combined with default data.
To start a case study, load one of the default chains with the menu entry File → Load Default Chain. This will pop up the "DEFAULT CHAIN" window in Figure 3.5. The entries below the line are the chains corresponding to the case studies. Select one of the cases by clicking on the diamond shaped button to the left of the name and hit OK.
Next load the corresponding default data with the menu entry File → Load Default Data. This pops up the "DEFAULT DATA" window in Figure 5.1. Select the default data corresponding to the default chain you just loaded and hit the OK button. The "LOAD OPTIONS" window will pop up. All you have to do now is to redirect the loaded data to the existing data box. The chain will automatically start execution, and the demo example will be recomputed for you.
In the rest of the chapter we shortly review where the default data come from and what the main results are. We will not go into the details of how and why each chain was constructed in its specific way, since the main idea behind the case studies is twofold:

- encourage you to experiment with ISID. You can for instance change parameters in the chains and add/delete new algorithms to the chain.
- demonstrate the capabilities of ISID.

The default data can still be loaded if a chain does not yet exist. You can redirect the default data to new data boxes and start building your own chain from there on.

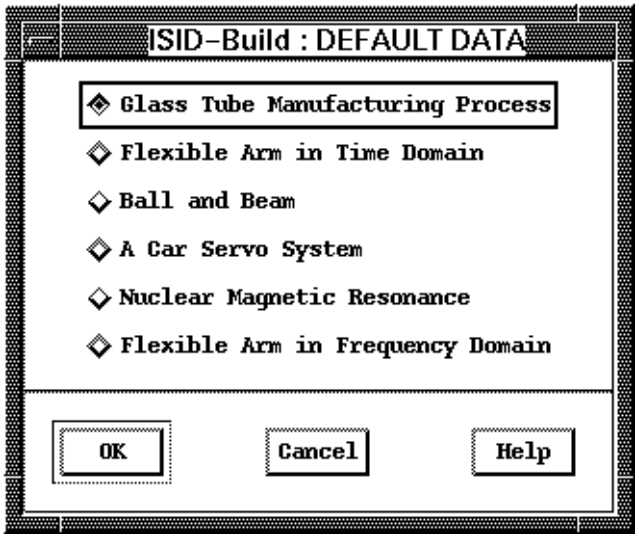


Figure 5.1 The "DEFAULT DATA" window allows you to load any of the default data sets.

5.1 A Glass Tube Manufacturing Process

Problem Description

To produce glass tubes, quartz sand is fed into the top of a furnace. The sand is melted to glass inside the furnace. The glass tubes are drawn at the bottom.

- **Inputs:** The inputs are drawing speed and mandrel pressure. The drawing speed is the speed at which the tubes are pulled out of the bottom of the machine. The mandrel pressure is the pressure applied to the mandrel at the top of the machine. Through the mandrel, this pressure is then applied to the inside of the tubes when they are pulled out of the machine.

- **Outputs:** The important outputs are the geometrical parameters of the tubes: mean diameter and thickness.

Two input-output data sequences were measured (sampling frequency 2 Hz). The first one (1355 points) is used for the identification of the process. The second one (893 points) is used for the validation of the results. For both input signals a pseudo random binary noise sequence was used as an input.

Chain Description

Figure 5.2 shows the chain that was used to identify the process.

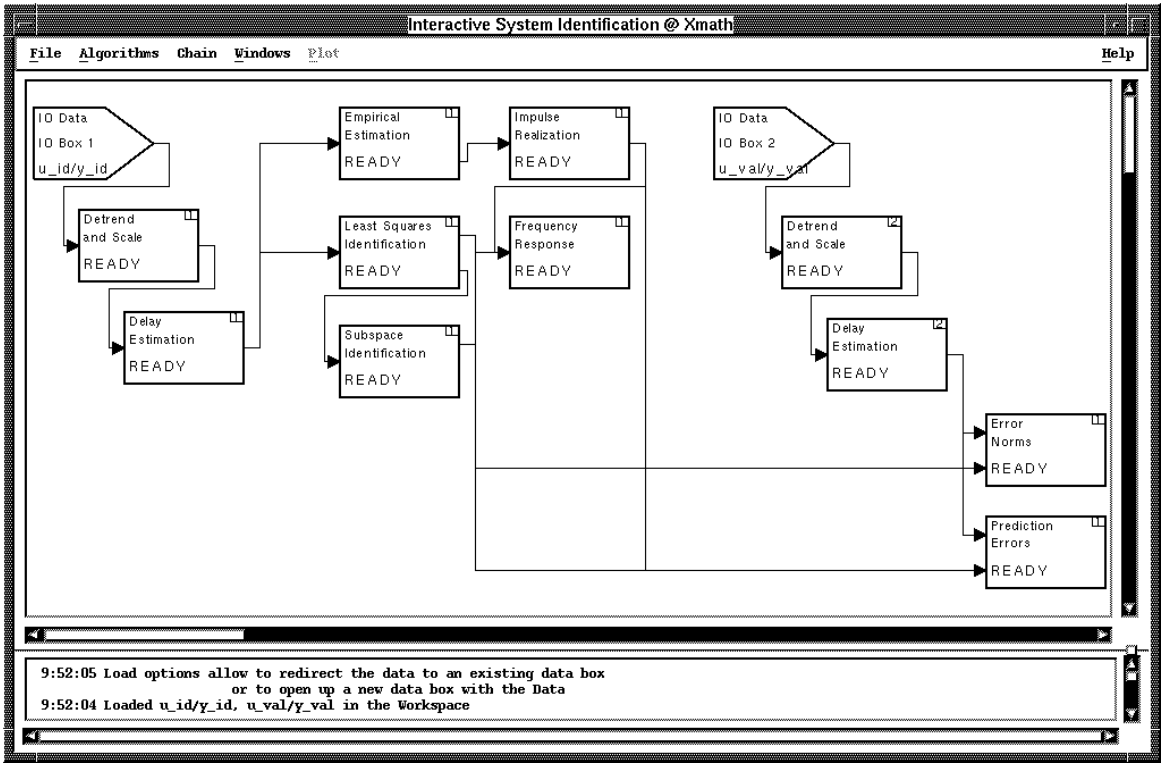


Figure 5.2 Chain used for the identification of the glass tube manufacturing process

- **Processing:** Both input-output data records are first detrended to remove the mean and the linear trends in the data. Since the outputs can only be measured after the tubes are sufficiently cooled down, there are significant time delays. The two delay estimation blocks estimate these delays and compensate for them by shifting the input and output signals in the appropriate direction.
- **Identification:** Three different identification algorithms are applied to the data. By comparing the resulting models, we can enhance our confidence in them.
  - A first model is obtained by realizing the impulse response obtained from the empirical transfer function estimate.

  - A second model is obtained from a least squares identification.
  - A last model is obtained from a subspace identification algorithm. The subspace algorithm has the square root output of the least squares algorithm as its input. This will speed up the chain execution.

- **Validation:** Error norms and prediction errors for the identified models are inspected.

- **Display:** We plot the transfer functions of the obtained models on the same plot to compare the models in the frequency domain.

Discussion

Look at the first delay estimation block. The response of the thickness is about 8 sec slower than the response of the diameter. This is because the thickness is measured *after* the diameter. The negative impulse response from speed to diameter and thickness indicate that an increase in drawing speed leads to a smaller diameter and thickness. On the other hand, increasing the pressure leads to an increase of diameter and a thinner wall.
Looking at the empirical transfer function identification block, you find the same impulse response as in the time delay block, but this time without the delays (this can be clearly seen when the axes of the empirical impulse response are adjusted). This impulse response is realized using the impulse realization block. The order of the model (8) can be determined in different ways. First, there is a slight "leveling off" of the singular values at 8. Second, examine the error norms for different model orders. To do this, disable the frequency response and prediction error icon, and change the order of the impulse realization to `[4:12]`. From the resulting error norms you can conclude that 8 is a good order (small simulation errors). The original and realized impulse responses can be compared by changing the Show On Plot parameter from "Singular Values" to "Imp. Responses". They agree well.
The order of the least squares and subspace identification model can be determined in similar ways (from the bar graphs or from the error norms). We choose order `10-10` for the least squares identification and order `8` for the subspace identification.
Looking at the error norm window, we find that the three models are comparable. Changing the "Mode" parameter of the error norm window from "Simulation" to "Prediction" shows the prediction errors.
Looking at the frequency response window, we detect different anti-resonances in the "impulse" and "subspace" model. The "least squares" model does not display these dips. This could be an explanation for the slightly worse error norm of the least squares model.

Suggestions:

- Plot the empirical transfer function in the frequency window. Some of the anti-resonances also show up in the empirical transfer function.
- Switch the role of validation and identification data.
- Check lower order models to see if we have over-modeled the system.

5.2 A Flexible Arm in Time Domain

Problem Description

This problem compares different identification methods on data from a flexible robot arm[1]. The arm is installed on an electrical motor (having a non-linear behavior). Because we are interested in linear systems, we have modeled the transfer function from the measured reaction torque to the acceleration.

- **Input:** The input of this process is the reaction torque of the structure on the ground.
- **Output:** The output is the acceleration of the flexible arm. The acceleration contains information about the flexible resonances and anti-resonances.

One input-output data sequence of 8192 points was measured (sampling time 0.002 sec). The input applied to the system is a multi sine with 200 frequency points equally spaced over the frequency band from 0.122 Hz to 24.4 Hz.

Chain Description

Figure 5.3 shows the chain that was used to identify the process.

- **Processing:** Since the input only contains energy to up to 25 Hz, the data is first filtered with a low pass filter with cut off frequency 25 Hz. It is than post sampled by a factor of 10. Finally the processed data record is split into a validation and an identification data set.
- **Identification:** Parametric models are identified using a least squares and a subspace identification algorithm. A non-parametric frequency domain model is also identified using the empirical transfer function estimation algorithm.

[1]We are grateful to Hendrik Van Brussel and Jan Swevers of the laboratory of Production Manufacturing and Automation of the Katholieke Universiteit Leuven, who provided us with this data, which was obtained in the framework of the Belgian Programme on Interuniversity Attraction Poles (IUAP-nr.50) initiated by the Belgian State - Prime Minister's Office - Science Policy Programming.

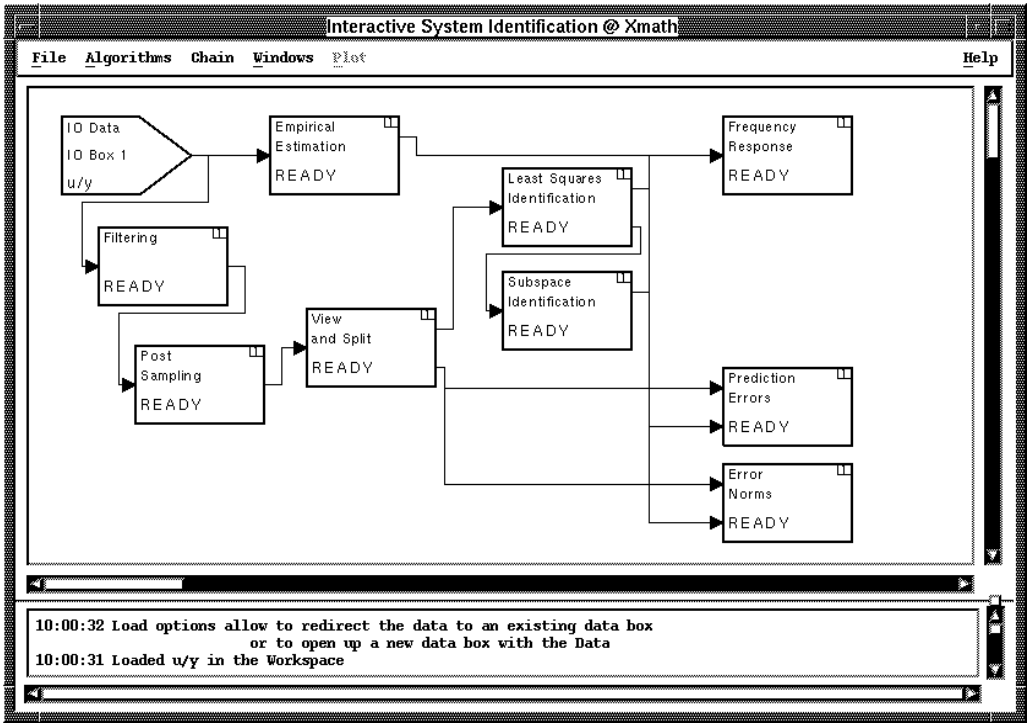


Figure 5.3 Chain used for the identification of the flexible robot arm

- **Validation:** The error norms and prediction errors are inspected.
- **Display:** The frequency responses of the three different identification techniques are compared.

Discussion

The input was designed in such a way that it is periodic over 4096 samples. That is why the window width of the empirical transfer function estimation block is set to 4096 and the window type is rectangular. The sharp peaks in the empirical transfer function are due to the discrete frequency energy in the input signal (the 200 discrete frequency points in the input).
Filtering is required since the empirical transfer function clearly shows a third resonance peak around 42 Hz. If we did not filter the data, we would identify this resonance peak too. Since the data is badly conditioned around 42 Hz (no input energy anymore, check the fast Fourier transform in the post sampling block) this would lead to inaccurate results.

The subspace identification algorithm clearly indicates a fourth order system (the two resonance peaks lead to four states). The order of the least squares model is also set to four.
From the frequency response window we can conclude that the parametric and non-parametric models agree well. For this case the error norms of the subspace identification algorithm are about a factor of 2 smaller than the least squares error norms.

Suggestions:

- Try out different identification algorithms on this data.
- Find the optimal order of the least squares model by looking at the validation error norms.
- Try to optimize the subspace identification model using the prediction error approach.

5.3 Ball and Beam

Problem Description

The ball and beam system consists of a horizontal beam with a ball placed on top of it. The angle of the beam (with the horizontal axis) can be controlled, making the ball roll back and forth on the beam. The system is considered to be linear around its horizontal working point.

- **Input:** The input of the process is the angle (in radians) of the beam.
- **Output:** The output of the process is the velocity of the ball. In fact we are interested in the position of the ball, but that means we have to identify an integrator (one integrator from beam position to ball position). We have thus differentiated the position signal to obtain the ball velocity.

One input-output data sequence of 1000 points was measured (sampling time 0.1 sec). The input signal is generated manually in this case. Since we want to avoid the ball running against the edges of the beam to avoid hard non-linearities.

Chain Description

Figure 5.4 shows the chain that was used to identify the process.

- **Processing:** We only subtract the mean of the input output data.

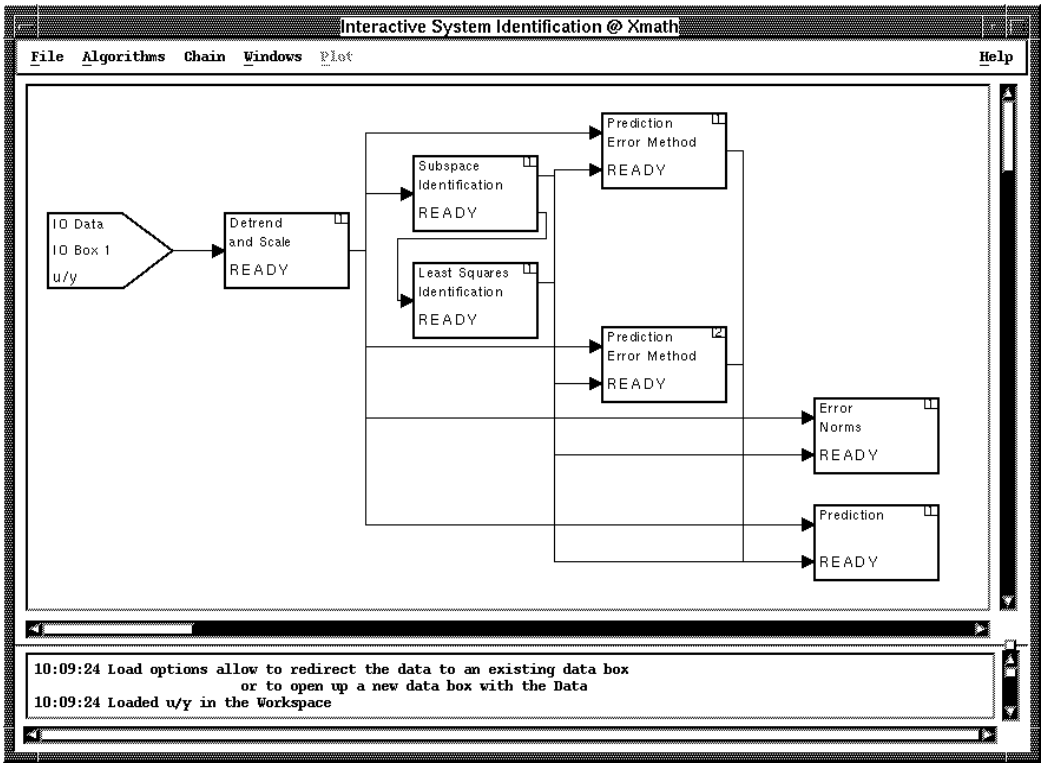


Figure 5.4 Chain used for the identification of the Ball and Beam setup.

- **Identification:** Two identification algorithms are used to calculate initial models. These initial models are then further optimized using the prediction error method algorithm.
- **Validation:** The error norms and the predicted signals are used for validation.

Discussion

The models calculated by the least squares and the subspace identification algorithm are used as initial models for the prediction error method algorithms. Note that any other model calculated by other identification algorithms could also be used as the initial model. Note also that the prediction error method is quite slow, and that the improvement is not very spectacular for this example.
Turning our attention to the error norm plot, we see that the simulation error of the optimized least squares model is worse than the simulation error of the initial model.

This is because the prediction error methods optimize the *prediction* error. Changing the "Mode" parameter of the error norm algorithm reveals that the optimized *prediction* error is indeed better than the prediction errors of the initial models.

Suggestions:

- Look at the frequency and impulse responses of the different models.
- Try different identification techniques to find the initial model and compare the results with the results of the default chain.

5.4 A Car Servo System

Problem Description

This problem models a hydraulic actuator for an automatic gear box of a car. The actuator has two discs (primary and secondary) with a belt in between. The conic discs are constructed in such a way that a sideways pressure changes their radius. In this way the transmission ratio can be altered. We study the actuator that delivers the pressures. The inputs are signals from a Pulse Width Modulator (the servo signals). The outputs are the actuator pressures.

- **Inputs:** Required pressures coming from a PWM: Secondary servo, Primary servo and Clutch servo.
- **Outputs:** Actuator pressures: Secondary pressure, Primary pressure and Clutch pressure.

One input-output data record of 3000 samples was measured (sampling time 0.002 sec).

Chain Description

Figure 5.5 shows the chain that was used to identify the process.

- **Processing:** First the fast Fourier transform of the data is displayed to inspect its frequency contents (no oversampling is performed). The data is then detrended and filtered with a 25 Hz lowpass filter (to remove spurious frequencies in the output data). Finally, the data is split in an identification and a validation data record.
- **Identification:** The system is identified using the least squares algorithm.
- **Validation:** The models are validated by looking at the error norms and at the predicted output signals.
- **Display:** The frequency response of the model is displayed.

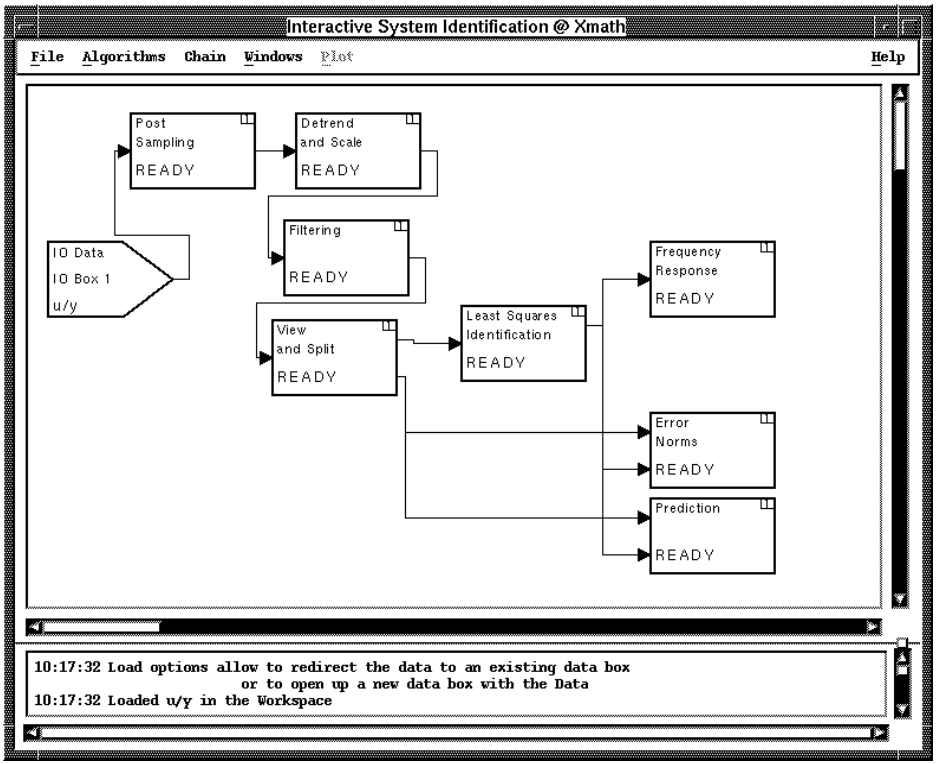


Figure 5.5 Chain used for the identification of the car servo system.

Discussion

The 61 Hz disturbance of the Pulse Width Modulators is clearly visible in the Fourier transform (use data viewing to determine the exact frequency). Since this frequency band is not of interest to us, a low pass filter is applied at 25 Hz.
Note that the quality of the linear model of this fairly non-linear system (that would require much physical insight and expertise to be modeled by first principles) is quite high: a simulation error of around 15 %. This illustrates the power of black box identification techniques, even for complex multi-variable systems.

Suggestions:

- Try different orders of the least squares model to find the best model order.
- Try to identify the model using an empirically estimated impulse response. Realize the non-parametric impulse response with the impulse realization algorithm.

5.5 Nuclear Magnetic Resonance

Problem Description

This problem studies a Nuclear Magnetic Resonance (NMR) signal of the Human skull[2]. The problem shows how impulse response data can be used by ISID to identify a model. The complex signal coming out of the scanner is split into a real and an imaginary part. These are considered as the two outputs of the system. The sampling time is 0.001 sec.

Chain Description

Figure 5.6 shows the chain that was used to identify the process.

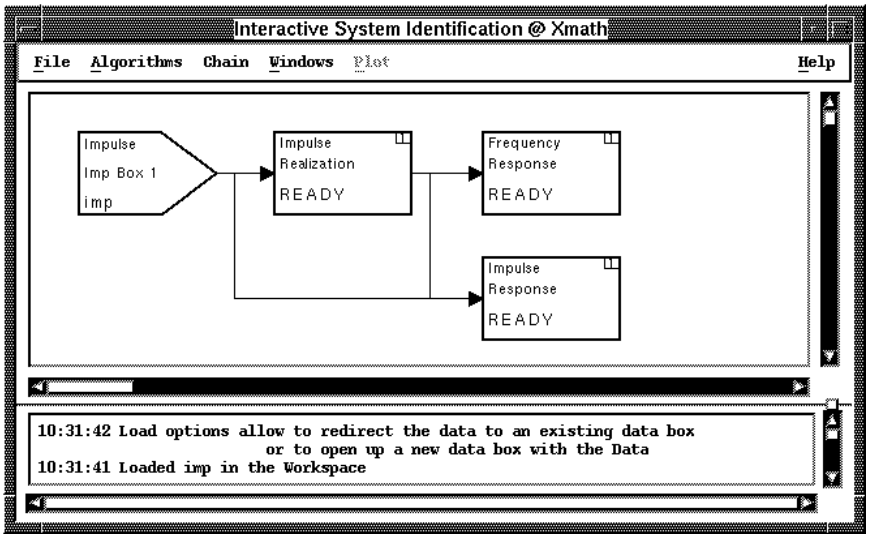


Figure 5.6 Chain used for the identification of the NMR signal.

- **Identification:** The impulse realization algorithm is used to realize the NMR impulse response.

- **Display:** The frequency response of the identified model is displayed. The identified and original impulse response are compared.

[2]The NMR was performed at the Universitäts Klinik of the University of Cologne, Germany.

Discussion

The frequency response allows physicians to extract the concentration of certain substances in the human skull. When you zoom in on the peaks in the frequency domain (use <Ctrl> left mouse to draw a lasso) you will see the peaks of Choline (around 95 Hz), Creatine (around 106 Hz) and N-acetyl-asparate (around 172 Hz). Data viewing will help you in identifying the location of the peaks.
The modeled and original impulse response correspond very well.

Suggestions:

- Model only the real part of the impulse response by changing the defaults of the data box containing the impulse response.

5.6 A Flexible Arm in Frequency Domain

Problem Description

As in the second case study (see Section 5.2), a flexible arm is mounted on a direct drive motor. However, this is a different type of arm and motor, so the results of this case study and the second case study should not be compared. This case study shows how frequency response data can be used by ISID to identify a model.

- **Input:** The voltage applied to the motor is taken as the input of the system.
- **Output:** The outputs are the strain in the arm (measured with a strain gauge - note that the strain is proportional to the acceleration) and the radial position of the arm (measured with a radial encoder).

The frequency response data was measured by exciting the arm with sine waves at different frequencies. The sampling time of the identified model is taken equal to 0.02 sec.

Chain Description

Figure 5.7 shows the chain that was used to identify the process.

- **Identification:** The frequency identification algorithm is used.
- **Display:** The impulse response, frequency response and poles and zeros of the identified model are displayed.

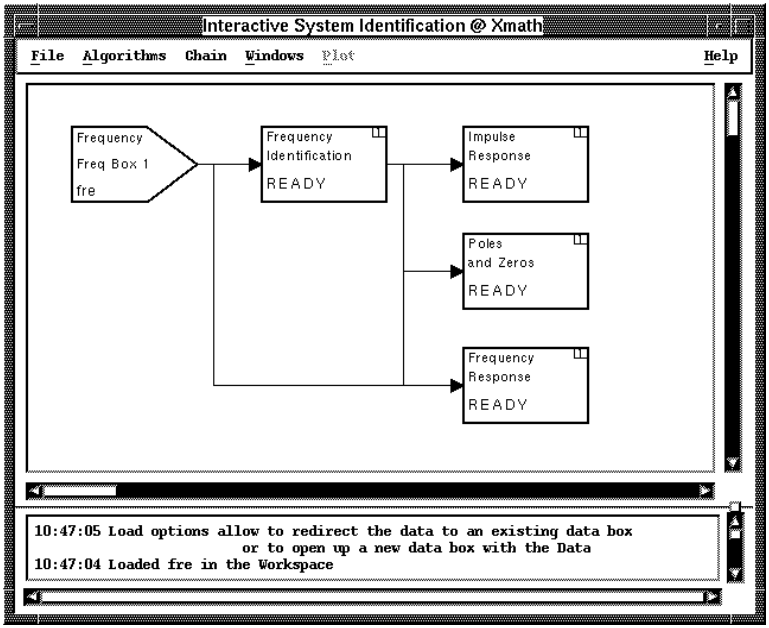


Figure 5.7 Chain used for the identification of the flexible arm in the frequency domain

Discussion

Since the voltage is proportional to the force, we expect to have two integrators in the transfer from voltage to position (output 2). In the frequency plot, the slope of the second transfer function (the one of the second output) can be checked to have a slope of -40dB/decade indeed. You can easily check this using data viewing.
The existence of this double integrator can also be checked by looking at the impulse response and at the poles of the identified system. Zoom in on the pole and zero plot and use data viewing to check that two poles are indeed lying close to one.
The frequency response clearly shows the first anti-resonance (3 Hz) and resonance (5 Hz) of the mechanical system.

Suggestions:

- Use the weighting function in the frequency identification algorithm to influence the results.
- Identify the outputs of the system separately.

Appendix A

Loading Data in Xmath

The Xmath command `load` can bring in Xmath and MATRIX$_X$ data. The function `read()` allows you to read data in other formats to an Xmath variable.

- Xmath files and MATRIX$_X$ FSAVE files can be loaded by typing:

  ```
  load "file_name"
  ```

- Matlab data must be saved as ASCII and brought into Xmath with `read()`. If you have a variable `uy` in Matlab (m rows and n columns), save it as follows:

  ```
  save file_name uy -ascii -double
  ```

  To read the file into Xmath, type:

  ```
  uy=read("file_name",m,n,"ascii",0);
  ```

- **Character Data Files :** Often, files are delivered from process computers in ASCII format. This implies that each of the lines in these files consist of (ASCII) numbers, separated by spaces. Each line represents the measurement of all channels at a certain time instance. If there are n channels and m time points, then you can read the file with the same command as above :

  ```
  uy=read("file_name",m,n,"ascii",0);
  ```

- **Other formats :** The `read()` function also allows you to read files consisting of "floating" or "double" numbers.

Index

C

D

I

L

M

N

O

Graphical Interaction

Chain Plot interaction	
Left click on icon	select/deselect/connect
Left double click on icon Left <Shift> click on icon	deiconify/raise
Left <Ctrl> click on icon	close
Left <Alt> click on icon	disable/enable
Left drag on icon	move icon
Middle click on icon	deiconify/raise disable/enable/close help/information
Left <Ctrl> click on connection	delete connection
Middle click on connection	delete connection information on connection
Left click background	deselect all/place algorithm
Left <Ctrl> click background	cancel placing
Middle click background	automatic execution on/off start/step/stop chain
Right click background	iconify/reorder/deselect all

Algorithm Plot interaction	
Left click on data/background/legend	algorithm specific
Left <Ctrl> click in subplot	zoom subplot/show original data
Left <Ctrl> drag in subplot	lasso zoom
Left <Ctrl> click/drag on axis	re-range axis (inward)
Left <Ctrl> click outside axis	re-range axis (outward)
Plot → Scale → Automatic	re-range axis automatic
Left <Ctrl> click on title	toggle original data/zoomed data
Middle click/drag	hourglass
Middle <Ctrl> click/drag	bigger hourglass
Middle <Shift> click/drag	hourglass/larger magnification
Middle <Ctrl> and <Shift> click/drag	bigger hourglass/larger magnification
Right click/drag on data	data viewing

Accelerators

<Ctrl> - e	Start chain execution
<Ctrl> - s	Step chain execution
<Ctrl> - c	Stop chain execution
<Ctrl> - w	Show chain window
<Ctrl> - a	Iconify all windows
<Ctrl> - r	Reorder all windows

Building Blocks

Data Boxes	Short	Input	Output	Function
Input Output Data Box	IO Box	Xmath Var.	IO	Load IO data record
Frequency Data Box	Freq Box	Xmath Var.	Freq	Load frequency response
Impulse Data Box	Imp Box	Xmath Var.	Imp	Load impulse response
Model Data Box	Mod Box	Xmath Var.	Model	Load discrete model
Square Root Data Box	SR Box	Xmath Var.	SR	Load square root

Processing	Short	Input	Output	Function
Detrend and Scale	DETREND	IO	IO	Scaling and trend removal (drift ...)
Peak Shaving	PEAK	IO	IO	Removal of outliers (sensor failure ...)
Delay Estimation	DELAY	IO	IO	Estimation and compensation of delays
Filtering	FILT	IO	IO	Low, high or band pass filtering
Post Sampling	SAMPLING	IO	IO	Subsampling of a data set
View and Split	SPLIT	IO	IO (2)	Split in Iden. and Val. data set

Identification	Short	Input	Output	Function
Least Squares	LS	IO | SR	Model & SR	Least squares in time domain
Empirical Estimation	ETFE	IO	Freq & Imp	Non-parametric identification
Subspace Identification	SDS	IO | SR	Model & SR	Geometrical identification
Instrumental Variables	GIV	IO	Model	Using inputs as instruments
Prediction Error Method	PEM	IO & Model	Model	Cost function minimization
Frequency Least Squares	FWLS	Freq	Model	Least squares in freq. domain
Impulse Realization	IREA	Imp	Model	Realization of impulse resp.

Validation	Short	Input	Output	Function
Error Norms	ERNO	Model(s) & IO(s)	-	Display pred. error norms
Prediction	PRED	Model(s) & IO	IO	Predicted outputs
Prediction Errors	PER	Model(s) & IO	IO	Prediction errors
Covariance	COVAR-PE	Model(s) & IO	Imp	Covariance of pred. errors
Spectral Density	SDF-PE	Model(s) & IO	Freq	Spectral density of pred. errors
Cross Correlation	CROSS-PE	Model(s) & IO	Imp	Correlation inputs $\leftrightarrow$ pred. arrows

Display	Short	Input	Output	Function
Frequency Response	FREQ	Model(s) + Freq(s)	Freq	Display frequency responses
Impulse Response	IMP	Model(s) + Imp(s)	Imp	Display impulse responses
Spectral Density	SDF	Model(s) + Freq(s)	Freq	Display spectral densities
Covariance	COV	Model(s) + Imp(s)	Imp	Display covariances
Poles & Zeros	PZ	Model(s)	-	Display poles & zeros

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at `ni.com` for technical support and professional services:

- **Support**—Online technical support resources at `ni.com/support` include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at `ni.com/exchange`. National Instruments Application Engineers make sure every question receives an answer.
- **Training and Certification**—Visit `ni.com/training` for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit `ni.com/alliance`.

If you searched `ni.com` and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.