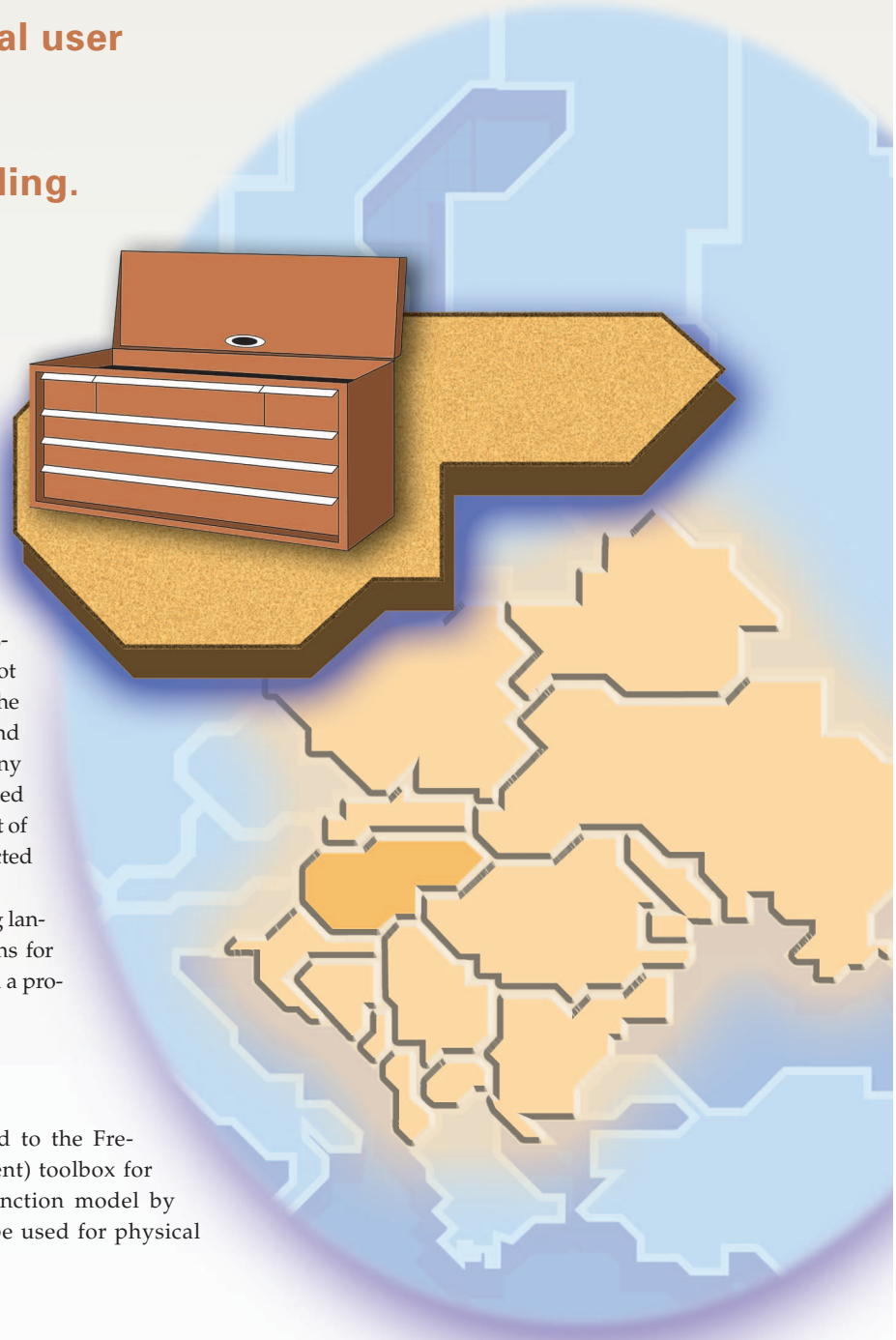
COMPLICATED PROCEDURES MADE EASY

Implementing a graphical user interface and automatic procedures for easier identification and modeling.

István Kollár,
Rik Pintelon,
Johan Schoukens,
and Gyula Simon



The theory of systems is beautiful, but the underlying mathematics is rather complicated. As a consequence, modern system identification methods are often not used even if their use is otherwise justified. The complicated structure of validity conditions and the many alternatives and possibilities keep many of us away. Moreover, because the use of advanced system identification methods often requires a lot of programming work, the attention can be deflected from the modeling issues.

The power of computers and of programming languages has allowed us to develop new solutions for identification and modeling. We have developed a program that effectively uses the following tools:

- graphical user interface (GUI)
- automatic procedures
- data structures.

These tools are simple to use and are added to the Frequency Domain System Identification (FDIdent) toolbox for MATLAB. FDIdent determines a transfer function model by measuring input and output data. This can be used for physical modeling or control.

The FDIdent Toolbox

The noisy nature of measured signals demands appropriate procedures for experiment design, measurement, data preprocessing, modeling, parameter estimation, and model

1094-6969/03/$17.00©2003IEEE

validation. Theoretical and numerical considerations need to be followed, and many users find the sum of these factors prohibitive. It is beneficial when the computer can replace some human decisions with a program. In other words, we need a specialized program.

The power of computers and of programming languages has allowed us to develop new solutions for identification and modeling.

MATLAB is a popular program used for scientific calculations [3]. It has easily accessible and reliable general-purpose algorithms, especially in linear algebra. Nice features of MATLAB are interactivity, powerful graphics, and that toolboxes can be added to the main program. Therefore, we developed this toolbox for MATLAB to perform system identification in the frequency domain [1].

In system identification, dynamic system models can be found that fit well to measured input and output data. During this procedure, the measured system can be anything from electrical systems (filters, machines) to mechanical systems (airplanes, cars, robot arm) and acoustical systems (airplane cabin, loudspeaker).

The toolbox seeks the transfer function of linear dynamic systems in the continuous domain (s-domain) in the form

$$H(s) = e^{-sT_d}\,\frac{b_0 s^0 + b_1 s^1 + \cdots + b_{nn} s^{nn}}{a_0 s^0 + a_1 s^1 + \cdots + a_{dn} s^{dn}} \tag{1}$$

or for discrete domain (z-domain) in the form

$$H(z) = e^{-f_s T_d}\,\frac{b_0 z^0 + b_1 z^{-1} + \cdots + b_{nn} z^{-nn}}{a_0 z^0 + a_1 z^{-1} + \cdots + a_{dn} z^{-dn}} \tag{2}$$

where nn is the order of the numerator, dn is that of the denominator, T_d is the system delay, and f_s is the sampling frequency. The task is to estimate from the measured data the order of the numerator and denominator polynomials, the numerator and denominator coefficients $b_0, b_1, \ldots, b_{nn}, a_0, a_1, \ldots, a_{dn}$, and the delay T_d.

The Graphical User Interface

The demand for easy-to-use tools was an important factor in developing a GUI on top of the existing subroutines. By seeing a graph and being able to control the data processing and modeling steps, the process becomes much easier. We hid the available algorithms from the user, so the user could concentrate on the data and the generation of the desired model instead of dealing with technical details.

A usable GUI is primarily practical, rather than theoretical. It is difficult to know beforehand what actions are "logical" for somebody. Therefore, we used the "paper prototyping" method to develop the program [6]. All windows and menus were prepared on paper, and we performed usability tests with colleagues. Each person acted as if using a program, and one of us played the role of the computer and reacted to each action. We recorded each session on video, analyzed why certain moves were made, and modified the design accordingly. The result was quite close to the final tool.

Even in this well-designed, intuitively clear tool, the users have several decisions to make. Obtaining good models with as few decisions as possible is best for many users. In the first design, we made two decisions.

- Almost all parameters have default values, set properly for the given data, which allow the user to accept the computer-suggested settings and pave the way for automatic runs of the program.
- Three different user levels can be chosen: automatic, interactive, and advanced processing. The automatic mode minimizes the number of questions to the user. The program decides whatever it can. Those who do not want to bother with many decisions can automatically run through the processing steps with default values. The interactive mode requires basic decisions, whereas the advanced mode allows the setting of all parameters.

The core of the GUI is a flow chart that allows you to access the data processing tools (controls) by opening any block, making the proper selection, and then letting it run. The whole identification procedure is illustrated in Figure 1.

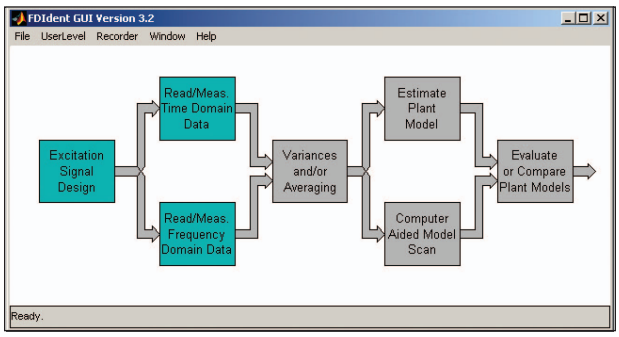


Fig. 1. Starting graph of the GUI of the Frequency Domain System Identification Toolbox (FDIdent).

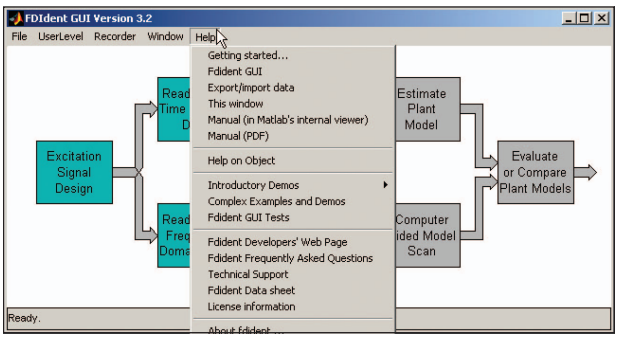


Fig. 2. The Help menu.

When starting, ***Help*** offers assistance. General help is offered under the menu items ***Getting Started, FDIdent GUI***, and ***This Window*** (Figure 2).

Special help is available under ***Help on Object***. When selecting this, the cursor changes to a question mark. When an object is selected by this mark, a help window opens up with an explanation of the object.

The Identification Procedure

In the following text, we use the setting **Interactive**, since we want to peek at the individual stages.

Let us follow the steps of an identification procedure. First, open the time-domain data acquisition window (Read Time-Domain Data). Figure 3 shows the situation after reading in a data object.

The data are periodic, so the Segmentation window offers segmentation (Figure 4). Segmentation is illustrated in a separate window (Figure 5).

After segmentation, the data are converted to the frequency domain, and then a selection of the frequency lines is offered. These are not illustrated here.

Now the Variance Analysis block can be opened. Figure 6 illustrates the situation after variance analysis.

When data preprocessing is finished, parameter estimation follows. The straightforward step is to open the **Estimate Plant Model** block. However, when experimentation with different model orders is desirable, **Computer Aided Model Scan** is the correct tool (Figure 7). In this block, we can select the desired combinations of the numerator/denominator orders and run the estimation subroutine for each of them.

Here, we would like to point out two modifier checkboxes.

- The transfer function is often badly conditioned from estimation and from evaluation points of view, especially when the numerator/denominator orders of the system transfer function are above 30-40. In such cases, round-off errors may deteriorate the results. A remedy is

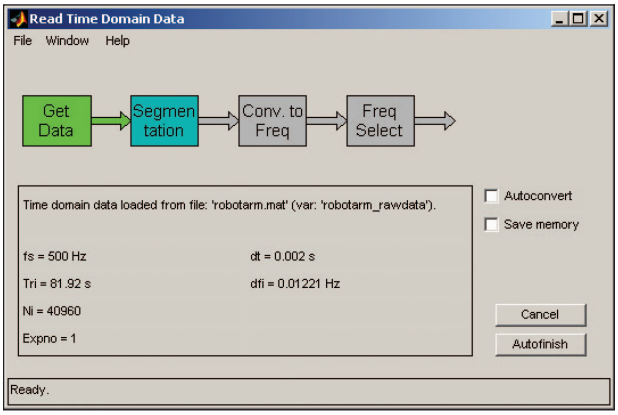


Fig. 3. The Read/Measure Time Domain Data window.

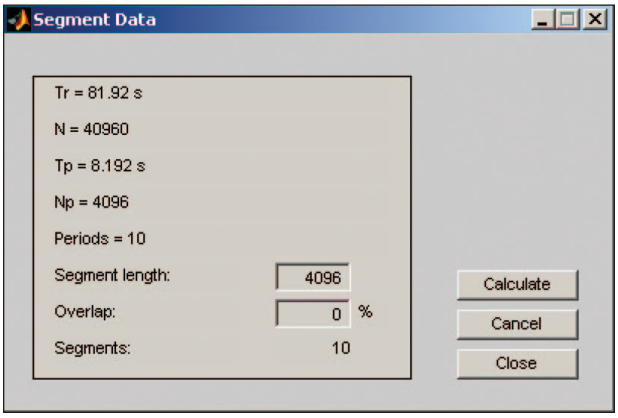


Fig. 4. The Segmentation window.

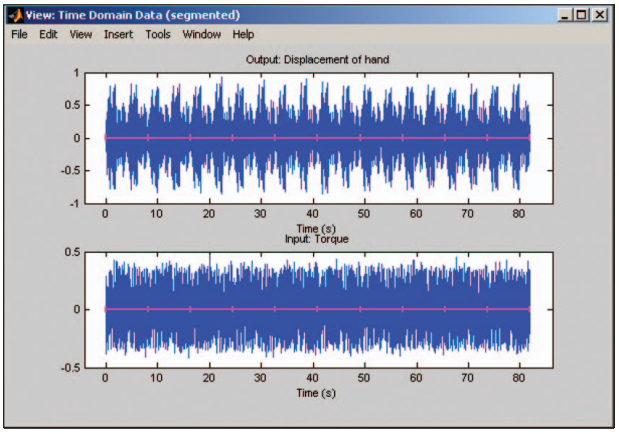


Fig. 5. Illustration of segmentation.

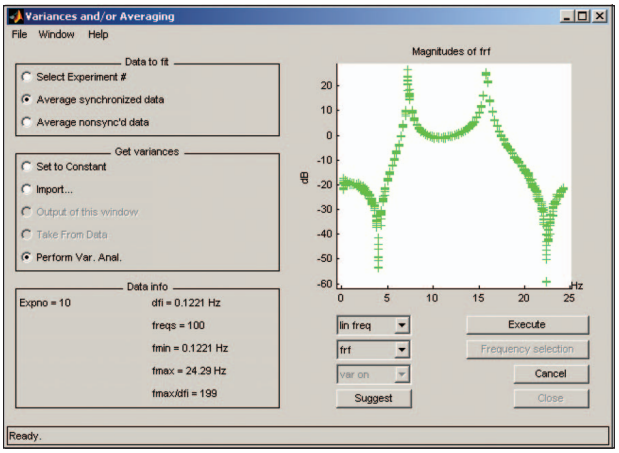


Fig. 6. Variance Analysis window.

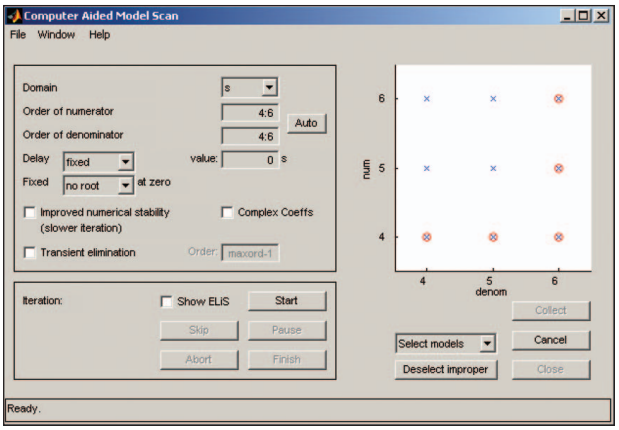


Fig. 7. Computer Aided Model Selection window.

the use of orthogonal (Forsythe) polynomials [7]. This is switched on by selecting **Improved numerical stability**. The user does not notice any difference in the interface (except slower processing speed), but a more involved code runs inside the program. This is typical for the GUI; a modification, which seems to be simple for the user, may significantly change the execution of the program.

- The situation is similar for the checkbox **Transient elimination**. This makes the identification algorithm deal with a transient component beside the periodic one, or to handle arbitrary excitation signals, whereas the user does not notice much difference in the interface [4].

During the run of the iterative minimization, **ELiS Run Info**, information about the run is shown (Figure 8).

When every desired identification result is at hand, the **Evaluate and Compare Plant Models** block can be opened. In Figure 9, the different properties of the identified models can be examined and compared.

People prefer automatic tools. Push a button, and get the result.

If we are interested in details of the selected plot, we can zoom into it by double-clicking. Further steps of model validation will not be illustrated here. Rather, we go back to the first block we skipped at the beginning, the **Excitation Signal Design** block (Figure 10).

Here we may observe another help, present in all windows: when the cursor is moved above a control object, after a short time, a one-line, balloon help pops up in a small frame. This gives the quickest information about graphics objects.

The **Excitation Signal Design** block assists in designing the desired input signals. As several parameters are interrelated, there are two appropriate "Adjust" buttons to assure consistency of all parameters. If a multisine is designed with the settings from Figure 9, the corresponding window is given in Figure 11.

Coming back now to the basic chart (Figure 12), every active element can be selected, and this starts the corresponding actions. Blocks open up to windows, and at the selection arrows, an action selection window appears (Figure 13). Thus, it is easy

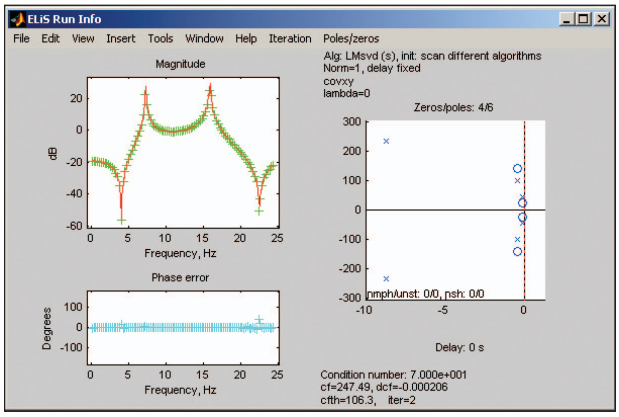


Fig. 8. Information window of ELiS.

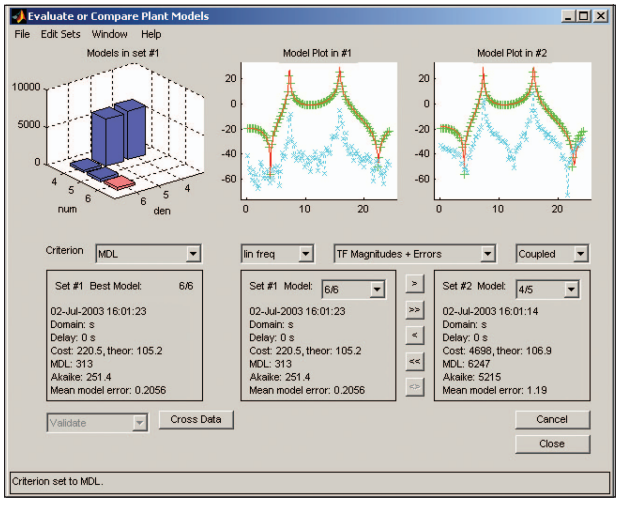


Fig. 9. Compare and Evaluate Models window.

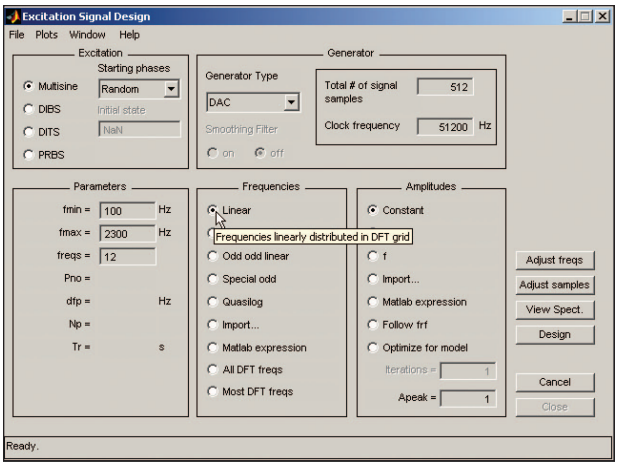


Fig. 10. Excitation Signal Design, with balloon help under the radio button "Linear."

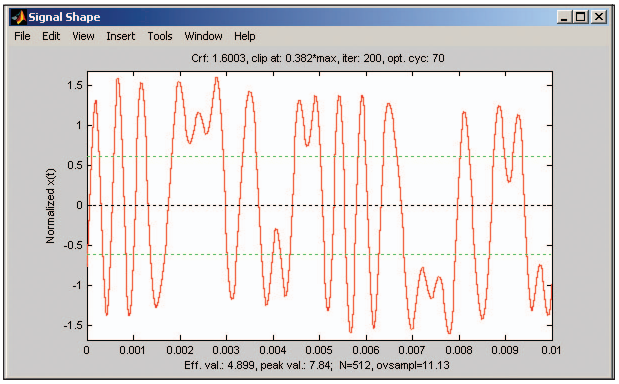


Fig. 11. Information window of crest factor minimization.

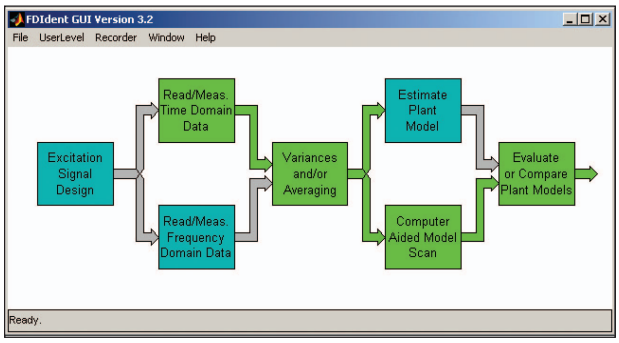


Fig. 12. The main window after identification. Colors denote the status of the corresponding objects: green boxes are finished ones with results, blue boxes are selectable but not yet executed, green arrows contain data, gray arrows contain no data.

Fig. 13. Arrow actions window.

to export/import data, perform graphical inspection, or display information about them.

In a good GUI, all actions are logical and consistent with each other. This was our goal with the above GUI. According to our experience, someone who has practice in identification can get along with support of the help; and after becoming familiar with the toolbox, identification can be made even without the helps.

Automatic Processing

People prefer automatic tools. Push a button, and get the result. Therefore, automatic processing steps have been added to the GUI. We localized the places for user decisions and made the whole process as automatic as possible. It is more important to design algorithms that minimize user interactions and are robust against misuse than to minimize the complexity of the algorithms because the computing power is increasing exponentially while the user capabilities do not change much. The ultimate goal was to have a tool that absorbs the data and returns verified results. In the automatic mode, identification can be accomplished with a minimum of user interactions in the graphics windows. A reasonable model is returned, ready for control or physical analysis.

In the interactive mode, selected steps of automatic processing can be executed. The points that we automated, and take over the task of decisions from the user, are the following:

- *Reliable Preprocessing of Periodic Data:* When measuring and processing periodic signals, there are two "catches." One is that it can often happen that the sampling frequency and the signal frequencies are not coherent, despite the efforts of the user; second, it is often cumbersome to extract the set of frequencies of the signal components. We have developed an automatic tool that determines the period length, major components, and variance of the measurement noise [8]. This allows the user to simply plug the measured signals into the GUI and watch how the estimated Fourier coefficients and variances come out.
- *Automatic Model Order Selection:* Determining the proper model order is a very difficult task. Different criteria exist (Akaike, Rissanen). These are implemented in the interface. Still, the most popular solution is to scan the reasonable numerator/denominator order combinations and select on the basis of some user-defined criteria. This is also realized in the **Computer Aided Model Scan** window.

  Analysis of the system equations in the proper mathematical basis allows use of a systematic approach: small singular values in the singular value decomposition tell how much the system is over modeled, and

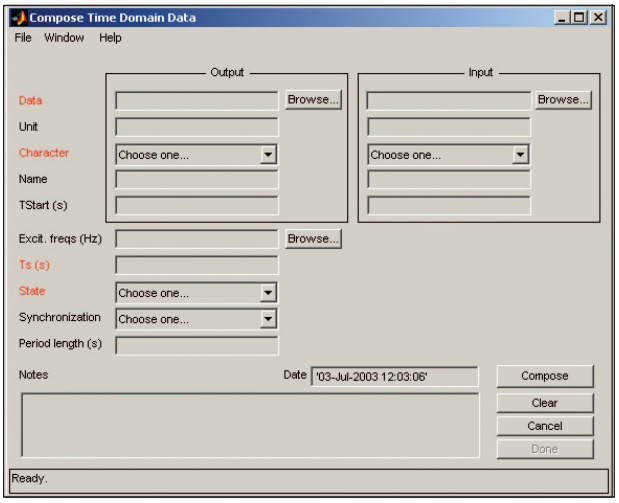


Fig. 14. Composition of time domain data: the names of the necessary fields are shown in red.

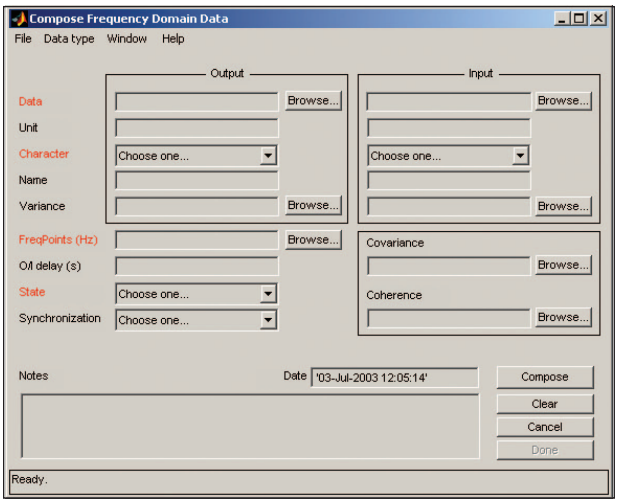


Fig. 15. Composition of frequency domain data: the names of the necessary fields are shown in red.

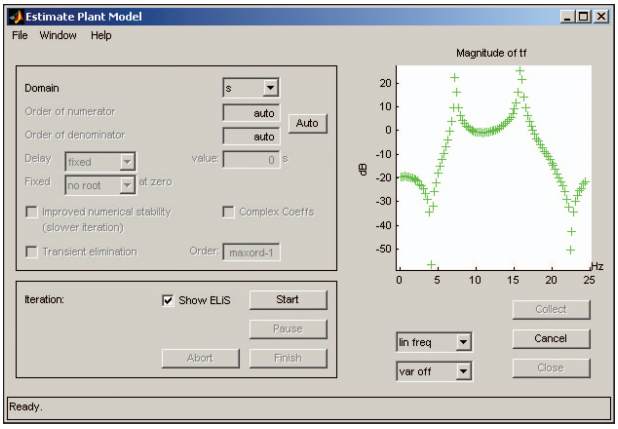


Fig. 16. In Automatic mode, almost all controls are inactive in the "Estimate Plant Model" window: the user needs only to select the domain and press "Start." The "auto" strings in the order windows mean that the best orders will be determined automatically.

careful "peeling" of the models reaches the proper minimum order model [9]. This provides a useful new tool in the toolbox.

Questions to Be Asked

The dream of a user is an intelligent machine that takes the data and returns good results while the user drinks coffee. This sounds good and can almost be achieved. In a typical automatic session, the user needs to give only the following answers:

- Bring the data into MATLAB, and tell the GUI import window where the data are. There are two possibilities: time-domain measurements and frequency response function measurements (Figure 14). (Providing consistent and properly described data to an algorithm is difficult. Despite different helps and descriptions, creation of a proper full object was not easy for someone who wanted to focus on identification rather than on the technical details of data import. We created a set of data import windows that made it possible to consistently generate data objects from the descriptors of the data and of the experiment (Figures 14 and 15).)
- Choose between *s*- and *z*-domain (continuous-time or discrete-time models). The domain could also be fixed beforehand, but in many cases its choice is required, and this is not a complicated question to answer (Figure 16).

Replaying Actions: The Action Recorder

Each programming job or program requires the ability to record and replay actions; the purpose can be to restore certain situations, reproduce errors, make demonstrations, generate and replay test sequences, or to automatically reprocess new data. The **Action Recorder** meets these requirements; details are given in [10].

Data Objects Examples

Here are a few examples based on the present implementation.

```
>>tdat = tiddata(randn(5,1),2*randn(5,1),100);
>> tdat.periodlength=500;
>>get(tdat)
        Version = 1.3
        Date = '13-Jun-2003 09:14:37'
        Input = [5x1 double]
        u = 'same as Input'
        Output = [5x1 double]
        y = 'same as Output'
        InputCharacter = 'BL'
        OutputCharacter = 'BL'
        PeriodLength = 500
        Ts = 100
        SampleNumber = 5
        Information = [4x34 char]
        OutputChNumber = 1
        InputChNumber = 1
        ChNumber = 2
        ExpNumber = 1
>>
>> fdat = fiddata(ones(5,1),1./(1+j*2*pi*
        [1:5]'),[1:5']);
>> get(fdat)
        Version = 1.3
        Date = '13-Jun-2003 09:18:21'
        Input = [5x1 double]
        u = 'same as Input'
        Output = [5x1 double]
        y = 'same as Output'
        InputCharacter = 'BL'
        OutputCharacter = 'BL'
        InputFreqPoints = [5x1 double]
        OutputFreqPoints = [5x1 double]
        FreqPoints = [5x1 double]
        AllFreqPoints = [5x1 double]
        FreqIndices = [5x1 double]
        FreqNumber = 5
        Information = [4x40 char]
        OutputChNumber = 1
        InputChNumber = 1
        ChNumber = 2
        ExpNumber = 1
>>
```

Data Structures

One difficulty in the use of general-purpose programs is the handling of several interdependent aspects of the data. A measurement not only consists of the measured samples but also of the sampling frequency, circumstances like periodic or random excitation, frequency contents, amplitude units,

names of measurement channels, date, and so on. If these are all to be given, function calls will usually become disorganized, difficult to check and debug, and frighten the user. The solution for this problem was the use of complex data types: cells, structures, and objects.

Objects allow consistent handling of related data and implementation of complex data-dependent procedures.

Objects improve the toolbox in two ways. First, all properties directly related to a given set of data or to a given model (model set) can be stored in a single entity. This entity has a built-in consistency check. It ensures that there is no contradiction within the object. Second, the toolbox functions can have a much simpler command-line form than before because fewer input/output arguments are necessary for their run.

Data Objects

Objects **tiddata** and **fiddata** implement the storage of measured or preprocessed data. They are the children of the generic object **iddat**, because many of the properties are common to them.

These two objects allow us to treat all information related to certain data as one single "variable" and set the properties in a simple way. A good example is the command **plot(obj)**, which plots the contents of any object in a reasonable form (Figure 17).

fiddata(frdobj) converts a **Frequency Response Data** object of the **Control System Toolbox** to a fiddata object, and **frd(fobj)** does the conversion in the other direction.

Model Objects

The object of identified models is **fidmodel**, which allows the straightforward handling of all properties and also offers much more. Direct transfer is possible to the **Control System Toolbox model objects**; moreover, certain functions of the **Control System Toolbox** can be called directly. For example, if the name of the model object is "mod," the commands **nyquist(mod)** or **bode(mod)** use the corresponding functions of the **Control System Toolbox**.

The implementation even defines simple arithmetic, so the commands **1/mod** (inverse of the model), **isstable(mod)** (is the model stable?), **stable(mod)** (stabilized model, by reflection of the unstable poles), and **mod1==mod2** (are the two models identical?) work just as expected. For comparison, models of the **Control System Toolbox** can be converted to a fidmodel object. Finally, this object is also prepared to accommodate models from the **System Identification Toolbox**, and smooth conversions between the two main model object types are also implemented.

Summary

There are several steps that can be made by programmers to facilitate the work of users. Computers can be brought closer to their original purpose: taking on difficult and repetitive tasks from us and, instead of dominating our work, facilitate it for us. The above examples point in this direction.

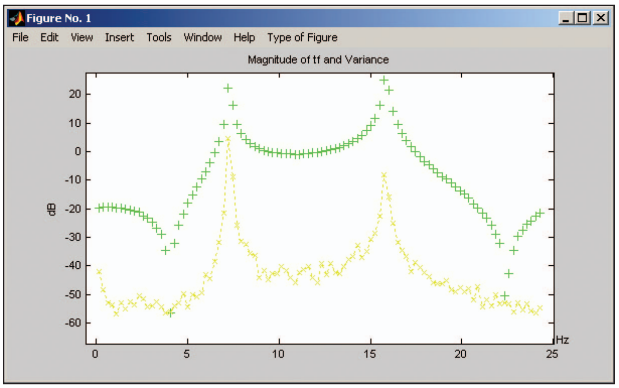


Fig. 17. Typical plot of a frequency domain data object.

Example of a Model Object

The model object can even store the data used for identification in the property data. Therefore, plot(fidmodelobj) can plot not only the transfer function but also the measured data and the error of the identification.

```
>>load bandpmod,
>>get(bkfit)
    Version = 2.2
    Date = '22-Oct-1998 09:09:26'
    History = {[1x48 char]}
    Data = [2x1x16 fiddata]
    Algorithm = [1x1 struct]
    Variable = 's'
    Representation = 'polynomial'
    num = [-4.0066e-018 -5.1034e-013
        -9.2494e-011...2.7921e-009
        1.5859e-005]
    denom = [-9.4278e-023 -2.3515e-019
        -3.3573e-015...-5.1609e-012
        -3.3479e-008 -2.3288e-005 -0.093552]
    FreqVect = [16x1 double]
    Fscale = 3835
    Delay = 0
    Covariance = [13x13 double]
    FitInfo = [1x1 struct]
>>
```

References

[1] FDIDENT (1999-2003). *Frequency Domain System Identification Toolbox Developers' Page* [Online]. Available: http://elecwww.vub.ac.be/fdident/

[2] K.R. Godfrey, *Perturbation Signals for System Identification*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[3] MATLAB home page. Available: http://www.mathworks.com/

[4] R. Pintelon and J. Schoukens. *System Identification—A Frequency Domain Approach.* Piscataway, NJ: IEEE Press, 2001.

[5] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 1986.

[6] M. Rettig, "Prototyping for tiny fingers," *Commun. ACM*, vol. 37, pp. 21-27, Apr. 1994.

[7] Y. Rolain, R. Pintelon, K.Q. Xu, and H. Vold, "Best conditioned parametric identification of transfer function models in the frequency domain," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 1954-1960, Nov. 1995.

[8] J. Schoukens, Y. Rolain, G. Simon, and R. Pintelon, "Fully automated spectral analysis of periodic signals," in *Proc. Instrumentation and Measurement Technology Conference,* Anchorage, AK, May 21-23, 2002, vol. 1, pp. 299-302.

[9] G. Simon, J. Schoukens, and Y. Rolain, "Automatic model selection for linear time-invariant systems," in *Proc. 12th IFAC Symp. System Identification, SYSID 2000*, Santa Barbara, CA, June 21-23, 2000, vol. I., pp. 379-384. (Extended electronic version: G. Simon, J. Schoukens, and Y. Rolain, "Automatic Model Selection for Linear Time-Invariant Systems—Practical Issues." Available: http://www.mit.bme.hu/~simon/publications/automodel.ps.zip)

[10] T. Dabóczi, I. Kollár, G. Simon, and T. Megyeri, "How to test graphical user interfaces?," *IEEE Instrument. Measure. Mag.*, vol. 6, pp. 27-33, Sept. 2003.

In a good graphical user interface, all actions are logical and consistent with each other.

Acknowledgment

This work has been supported by the Research and Development Fund for Higher Education (FKFP 0098/2001 and FKFP 0074/2001).

István Kollár received the M.S. degree in electrical engineering in 1977, the Ph.D. degree in 1985, and the D.Sc. degree in 1998, respectively. From 1989 to 1990, he was a visiting scientist at the Vrije Universiteit Brussel, Brussels, Belgium. From 1993 to 1995, he was a Fulbright scholar and visiting associate professor in the Department of Electrical Engineering, Stanford University, California. Currently, he is a professor of electrical engineering at the Budapest University of Technology and Economics, Hungary. His main interest is signal processing, with emphasis on system identification, signal quantization, and roundoff noise. He is a member of the IEEE Instrumentation and Measurement Society AdCom and EUPAS (European Project for ADC-based devices Standardisation).

Rik Pintelon received the degree of electrical engineer (burgerlijk ingenieur) in July 1982, the degree of doctor in applied sciences in January 1988, and the qualification to teach at the university level (geaggregeerde voor het hoger onderwijs) in April 1994, all from the Vrije Universiteit Brussel (VUB), Brussels, Belgium. From October 1982 until September 2000, he was a researcher of the Fund for Scientific Research—Flanders at the VUB. Since October 2000, he has been a professor at the VUB in the Electrical Measurement Department (ELEC). His main research interests are in the field of parameter estimation, system identification, and signal processing.

Johan Schoukens received the degree of electrical engineer and the degree of doctor in applied sciences from the Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 1980 and 1985, respectively. He is presently a professor at the VUB in the Electrical Measurement Department (ELEC). His research interests are in the field of identification of linear and nonlinear systems and growing tomatoes in his greenhouse.

Gyula Simon received the M.Sc. and Ph.D. degrees in electrical engineering from the Budapest University of Technology, Budapest, Hungary, in 1991 and 1998, respectively. Since 1991, he has been with the Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest. His research interest includes digital signal processing, embedded systems, adaptive systems, and system identification.