7

Graphical User Interface of System Identification Toolbox for MATLAB

Hiroyuki Takanashi[1] and Shuichi Adachi[2]
[1]*Akita Prefectural University*
[2]*Keio University*
Japan

1. Introduction

This chapter describes a Graphical User Interface (GUI) of a system identification device used with MATLAB. MATLAB is a well-known software package that is widely used for control system design, signal processing, system identification, etc. However, users who are not familiar with MATLAB commands and system identification theory sometimes find it difficult to use, typically because there are many different approaches to system identification. We propose using a GUI, which is especially suitable for beginners, to provide system identification procedures. The difficulties encountered by beginners in performing system identification might be reduced by using a GUI. The effectiveness of a GUI is illustrated using demonstration data in MATLAB.

Modeling of a plant is one of the most important tasks in control system design. There are two main approaches to modeling: white-box modeling based on first principles and black-box modeling based on input and output (I/O) data of a plant. The former is referred to as first principle modeling, while the latter is termed system identification.

Computers have become powerful and useful tools in control system design. Several sophisticated software packages (*e.g.*, MATLAB, SCILAB, Octave and MaTX) have been developed and are used for control system design and analysis.

MATLAB is a well-known software package that is widely used not only in engineering fields but also in other fields, including economic and biomechanical systems. MATLAB has many advantages for control system design and analysis. Important features include toolboxes for specific applications and a user-friendly programming environment.

A toolbox is a collection of functions that are appropriate for specific objectives. In particular, the system identification toolbox (SITB) (Ljung, 1995) provides useful functions for system identification. In the application of system identification theory to black-box modeling, using the SITB can dramatically reduce the user workload. However, because MATLAB interacts with the user via a command window, the user needs to know MATLAB commands.

MATLAB has user-friendly programming environment since variables need not be declared prior to being assigned and multidimensional arrays can be used as well as scalar variables. In contrast, C-language, Fortran and other programming languages require variables to be declared and arrays to be assigned.

Source: User Interfaces, Book edited by: Rita Mátrai,
ISBN 978-953-307-084-1, pp. 270, May 2010, INTECH, Croatia, downloaded from SCIYO.COM

System identification procedures for real plants consist of many steps, such as generating identification input signals for the plant, collecting I/O data, preprocessing and conditioning these data, executing a system identification algorithm and verifying the identification results (Adachi, 1996). Table 1 shows a standard system identification step and representative processing.

Step 1	Design of experiment	Determination of input signal, sampling frequency, etc.
Step 2	Identification experiment	Collecting I/O signals
Step 3	Preprocessing	Signal processing. Eliminating biases, trends, outliers, etc.
Step 4	Structural identification	Selection of model structure, model order, etc.
Step 5	Parameter estimation	Executing an identification algorithm
Step 6	Validation of the model	Comparison of output, pole-zero cancellation, etc.

Table 1. Several steps of system identification.

However, the accuracy of the estimated models depends on which procedures are used and the technical experience of the user. It is also difficult for beginners to judge to what extent the estimated model reflects the physical phenomenon. As a result, beginners in system identification find it difficult to apply the theory, so they are apt to avoid using it.

If the software were to provide a standard procedure for executing system identification, beginners might find the procedures easier. A GUI environment has the capability to provide such an environment. Moreover, if there was a device that could handle system identification processes automatically (or semi-automatically), similar to the way in which FFT analyzers or servo analyzers function, system identification theory might be more extensively used in engineering fields.

The purpose of this study is to develop a system identification device that can provide a structured framework to assist the user in performing system identification tasks. In particular, we develop a GUI environment for system identification based on the SITB (GUI-SITB).

The remainder of this chapter is organized as follows. Section 2 gives an overview of MATLAB software and system identification. Section 3 introduces the GUI for the SITB. The key topics of GUIs are described. Finally, Section 4 summarizes the chapter and describes open problems associated with the proposed GUI-SITB.

2. What is MATLAB and system identification?

This section first introduces the general aspects of MATLAB software. Then, an overview of system identification and the system identification toolbox are given.

2.1 MATLAB software

MATLAB is one of the most famous numerical computation software. It is widely used not only in control engineering communities but also in other research communities. MATLAB

has a C-like programming environment, but it has three distinctive features (Higham & Higham, 2000):

- Automatic storage allocation:
  Variables in MATLAB need not be declared prior to being assigned. Moreover, MATLAB expands the dimensions of arrays in order for assignments to make sense.
- Functions with variable arguments lists:
  MATLAB contains a large collection of functions. They take zero or more input arguments and return zero or more output arguments. MATLAB enforces a clear distinction between input and output. Functions can support a variable number of input and output arguments, so that on a given call not all arguments need be supplied.
- Complex arrays and arithmetic:
  The fundamental data type is a multi-dimensional array of complex numbers. Important special cases are matrices, vectors and scalars. All computation in MATLAB is performed in floating-point arithmetic, and complex arithmetic is automatically used when the data is complex.

2.2 System identification and MATLAB toolbox

One of the most popular modeling methods is first principle modeling. This method is sometimes called white-box modeling because it depends on the dynamical structure of the system under study. The dynamical structure is represented by physical laws, chemical laws, and so on. Thus, the structure of the system must be clear.

However, not all the dynamical structure of a system is always clear. System identification is a method for inferring dynamical models from observations of the system under study. System identification is sometimes called black-box modeling. The models are constructed under the assumption that the system structure is unknown. White-box and black-box modeling represent very different approaches, but they complement each other.

Fig. 1 illustrates some representative models and their relations. The relations allow the user to produce models according to their purposes and the situation of the system under study.

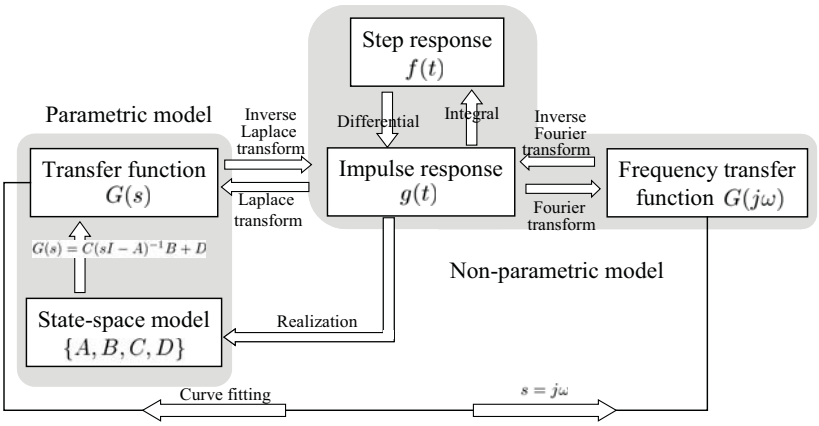

Fig. 1. Relations of parametric and non-parametric models.

To obtain an accurate model, the systems should be excited by an input signal because the model represents dynamical properties. White noise or a pseudo random binary signal

(PRBS) are representative input signals for system identification experiments. Systems should be excited sufficiently for system identification. On the other hand, systems should not be excited for control.

After performing system identification experiments, the raw data needs to be preprocessed to obtain accurate models. This step greatly influences the accuracy and quality of the model, because the raw data contains unnecessary frequency components, biases, trends, outliers, etc. These unnecessary components have a detrimental influence on models.

The remaining steps (Steps 4–6) are repeatedly executed. Thus, estimating the parameters and evaluating the model should be performed as successive processes.

MATLAB supports the above-mentioned steps. MATLAB includes some toolboxes that are designed for special objectives. Users can add any toolbox to their own environment. The SITB is based on system identification theory developed by L. Ljung (Ljung, 1995). However, the user requires experience to obtain a high-quality model by system identification.

3. Graphical user interface for system identification toolbox

3.1 Basic concept of GUI-SITB

For system identification methods to be widely used in practical engineering fields, it is desirable for the underlying theory to be as tractable as possible. Since system identification theory is based on statistical theory, signal processing, etc., the user needs *a priori* knowledge about these topics. However, if system identification theory could be realized in a measurement device, engineers could conduct system identification without needing to consider the theory.

The ultimate goal of this research is to produce a measurement device that performs system identification, that functions in a similar manner to FFT analyzers or servo analyzers and that is based on the underlying theory. One of the most important requirements of the measurement device is that everyone must be able to obtain the same results using it. Therefore, it is necessary to standardize system identification procedures in such a way that different users obtain the same result for the same problem if they follow the standard procedure.

Fig. 2 illustrates the basic elements of a system identification device. The simplest structure for the device consists of a personal computer (PC) running MATLAB with AD/DA converters attached. Ideally, MATLAB would perform all the processing.

System identification algorithms can utilize many types of model. To obtain a more accurate model, I/O signals must be processed before executing the system identification algorithm. Thus, the accuracy of the estimated model depends on the preprocessing and the models utilized.

For these reasons, it is difficult for beginners in system identification to obtain accurate and reliable models without considerable trial and error. However, if system identification and preprocessing procedures could be made very clear, there would be more likelihood that everyone would obtain the same models.

The first step in such a clarification is to establish an environment for system identification that consists of a set of standard procedures. Using a GUI is an effective strategy for realizing such an environment. Thus, in this chapter, we discuss the development of a GUI-based system identification toolbox (GUI-SITB) within MATLAB.

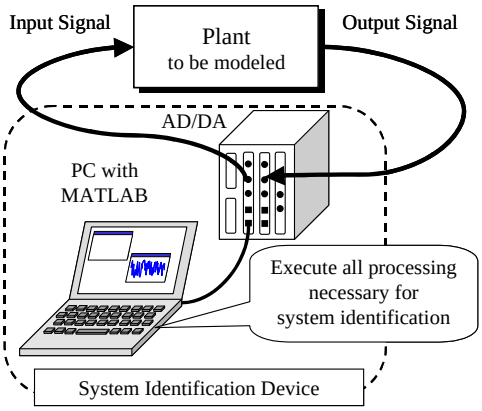

Fig. 2. Composition of a system identification device.

The SITB already contains a GUI environment called by the command "ident", which operates on preprocessing and system identification operations. However, the GUI-SITB in this study also supports other procedures, such as generating input signals and system identification experiments. Moreover, it provides identification procedures in a controlled stepwise manner by utilizing typical GUI features.

3.2 Features of GUI-SITB

In this section, we describe the features and functions of the GUI-SITB in detail. The GUI-SITB performs the following functions:

- generating input signals
- collecting I/O signals (system identification experiment)
- preprocessing I/O signals
- executing the system identification algorithm
- designing control systems

These functions and their sequences of application have been selected from a set of general system identification procedures. Although control system design is not strictly part of system identification, one of the main purposes of system identification is "modeling for control system design", thus it is natural to include control system design within system identification procedures.

Fig. 3 shows the main screen of the GUI-SITB that has been developed. Although the main screen shows a menu of five push-button functions, only certain operation sequences are allowed. In the following subsections, we describe the first four functions in detail.

Table 2 summarizes the software environment. Some of the following results have been obtained using the data used in the MATLAB demonstration program "iddemo1" (Ljung, 1999).

3.3 Generating input signals

In system identification experiments, input signals that contain many frequency components are required, since all dynamics of the plant must be excited. In the GUI-SITB, input signals are generated using the MATLAB command "idinput". This command generates several types of signals:

Fig. 3. Main screen of GUI-SITB.

Software	Version
Operating System	Windows 2000 (SP4)
MATLAB	6.5 (R13) SP1
System Identification Toolbox	5.0.2
Signal Processing Toolbox	6.1
Simulink	5.1

Table 2. Software environment.

- PRBS
- Gaussian random signal
- random binary signal
- sinusoidal signal

The minimum number of frequency components is defined by the persistently exciting (PE) condition. If the order of the plant to be identified is n, the order of the PE should be greater than or equal to $2n$. It is preferable for the input signal to contain as many frequency components as possible. From this viewpoint, a white noise signal would be ideal, although physically impossible to realize. As a result, the ideal input signal for linear system identification experiments is considered to be a PRBS.

There are some user-definable parameters when generating input signals using the GUI-SITB, including the number of samples, the maximum and minimum amplitudes, the upper and lower frequencies, the number of signals, and other parameters that depend on the type of signal.

Fig. 4 shows an example of a generated input signal. The figure shows some characteristics of the MATLAB subplot style, but each subplot can also be individually displayed by clicking the "View" option on the menu bar, as indicated in the figure.

For multiple input signals, only the first input signal is displayed and cross-correlation functions are also calculated. Since multiple-input system identification experiments require uncorrelated input signals, cross-correlation functions are calculated for all input signal pairs, and the results for correlations between the first input signal and each of the other input signals are displayed graphically.

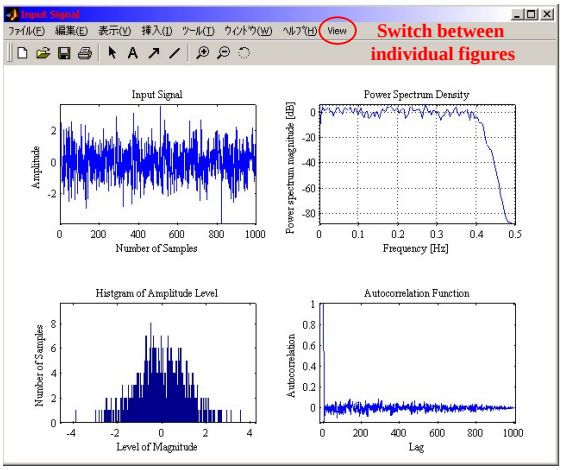

Fig. 4. An example of an input signal and its characteristics (upper left: input signal; upper right: power spectral density; lower left: histogram; lower right: auto-correlation function).

3.4 Collecting I/O signal (identification experiment)

Ordinarily, system identification experiments are carried out for real plants. Since one of the most important purposes of the GUI-SITB is to assist the user to learn the process of system identification, it includes an option of performing system identification experiments by simulations. A virtual environment is prepared for simulations.

The experimental environment in the GUI-SITB uses Simulink. A few Simulink models have been prepared for system identification experiments in the toolbox. The difference between using real plants and Simulink models is the target; the basic procedures and functions of the toolbox are the same.

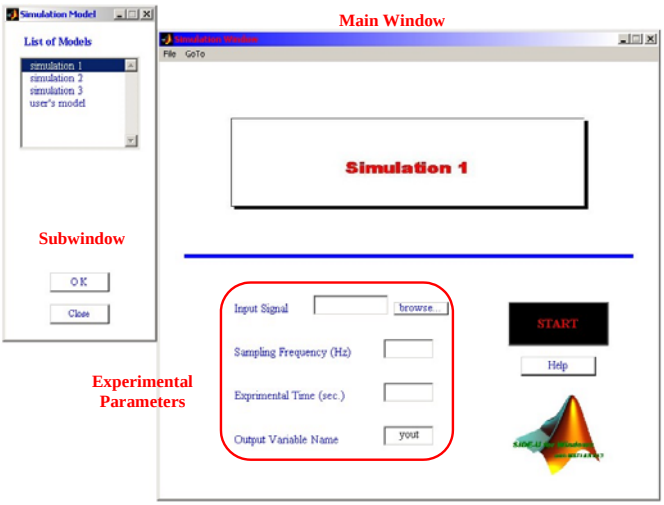

Fig. 5. System identification experiment window.

Fig. 5 shows the window for system identification experiments using Simulink models. The user selects a Simulink model from the left subwindow and then, in the main window, specifies the input signals (which have been saved as a mat file), the sampling frequency, the experimental time and the name of the output signal. After specifying these parameters, the "START" button is pressed. The output signal of the plant is then displayed and the I/O signals are saved as separate mat files. System identification experiments for real plants are currently being developed.

3.5 Preprocessing I/O signals

Preprocessing of I/O signals must be performed subsequent to system identification experiments. The raw data is contaminated with trend, drift and noise. Consequently, estimating the model operations will fail (*i.e.*, it will give bad estimates) if the identification algorithm is applied directly to the raw data. Therefore preprocessing is an essential prerequisite for system identification. Applying appropriate signal processing (The MathWorks Inc., 1998) will give an accurate model.

Typical preprocessing tasks are

- removing trends and biases
- resampling (decimation and interpolation)
- scaling
- filtering (enhancement of frequency ranges)

The trend removal procedure eliminates bias and any linear trends from the data. Time and frequency domain data are useful for this purpose.

In the system identification experiments, the I/O data is collected at an appropriate sampling frequency, which is usually determined based on information about the plant (*e.g.*, the band width of the closed-loop system and the rise time of the step response). However, when the information about the plant is unknown, it is desirable that the data collected over as short an interval as possible. After collecting the data, resampling can be applied to convert the sampling frequency.

The filtering procedure employs three types of filter: low-pass, high-pass, and band-pass filters. In the filtering process, the user specifies the frequency range (which is normalized by the sampling frequency) and the order of the filter. A Butterworth filter is then utilized for which the user specifies the order.

Several processing methods are listed in a drop-down menu. After the user selects one of these processing methods, the effect of preprocessing is displayed in both the time domain (as illustrated in Fig. 6) and the frequency domain. The upper part of Fig. 6 shows the unprocessed data, while the lower part shows the data after processing has been used to remove a trend.

Other preprocessing methods are also necessary sometimes. For example, treatment of missing data is one of the most important advanced preprocessing tasks (Adachi, 2004). The GUI-SITB cannot currently handle missing data, but there is a MATLAB command ("misdata") available via the command line.

3.6 Executing system identification algorithm and evaluation of the model

There are several model structures in system identification. However, basic system identification can be performed using only a few model structures. In this study, representative parametric model structures are prepared.

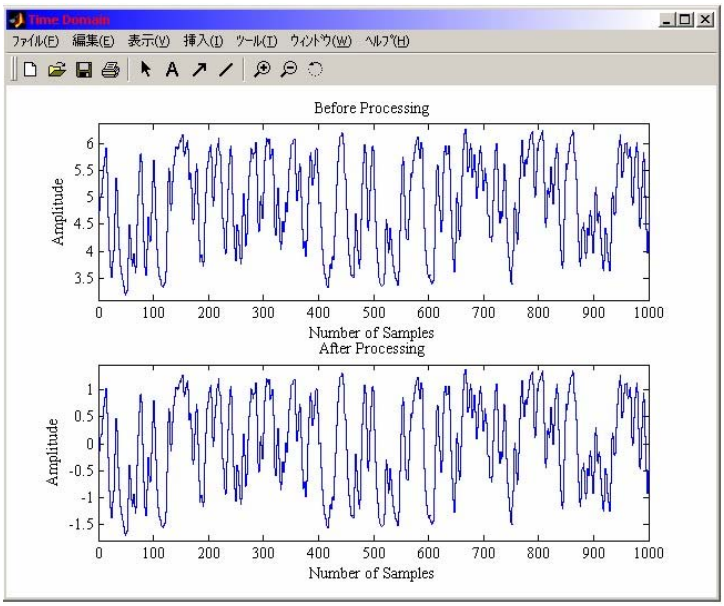

Fig. 6. An example of preprocessing of output signal (upper subplot: before processing, lower subplot: after processing).

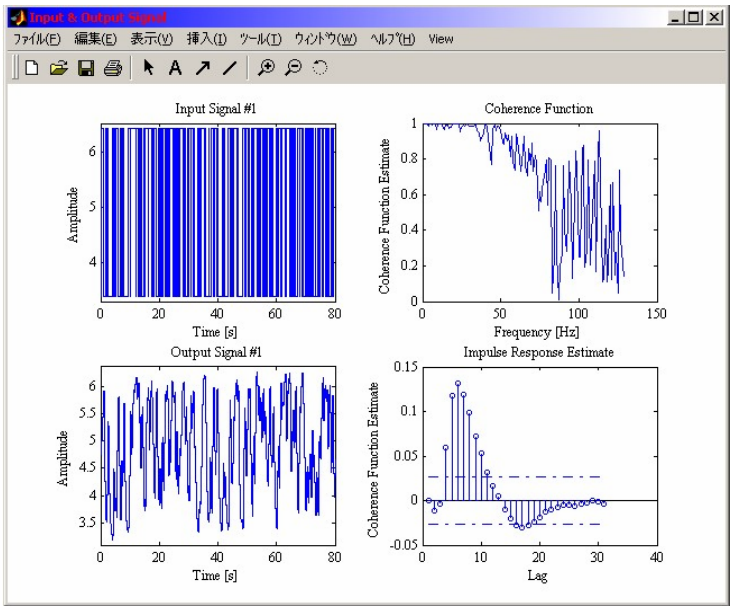

Fig. 7. I/O data for identification and their characteristics (left subplots: input and output signals; upper right subplot: coherence function of I/O; lower right subplot: impulse response estimate via correlation method).

The most basic parametric model structure is the ARX (auto-regressive exogenous) or a least-squares (LS) model. Other models include the ARMAX (auto-regressive moving average exogenous), OE (output error), and state-space models.
After the user has loaded the I/O data, this data and some of its characteristics are displayed as in Fig. 7. The I/O data, coherence functions of the I/O data, and impulse response estimate by the correlation method are illustrated. The number of samples for estimation is a user-definable parameter. In the default setting, if either the number of samples for estimation or the validation is not specified, the first half of the data is used for model estimation and the latter half is used for validation. When all the data is specified for estimation, the same data set is used for model validation. However, the low number of samples for the estimation results in poor estimates.
The available model structures in the GUI-SITB are

- ARX model via least squares and IV (instrumental variable) method,
- ARMAX model,
- OE model, and
- State-space model via the subspace method (Overschee, 1994; Viberg, 1995).

The user specifies the model order and the time delays for each model. The term "model order" refers to the orders of polynomials for the ARX, ARMAX and OE models and the number of states for the state-space model. Time delays can be estimated from the impulse response estimates, as shown in Fig. 7. In the bottom right figure, the dashed lines indicate a 99% confidence interval. The number of impulses within the confidence interval, starting from lag-0, is used estimate the time delay of the system.
Fig. 8 shows the frequency characteristics of the estimated ARX model, Fig. 9 shows a comparison of the outputs and Fig. 10 shows a pole-zero map. The frequency characteristics

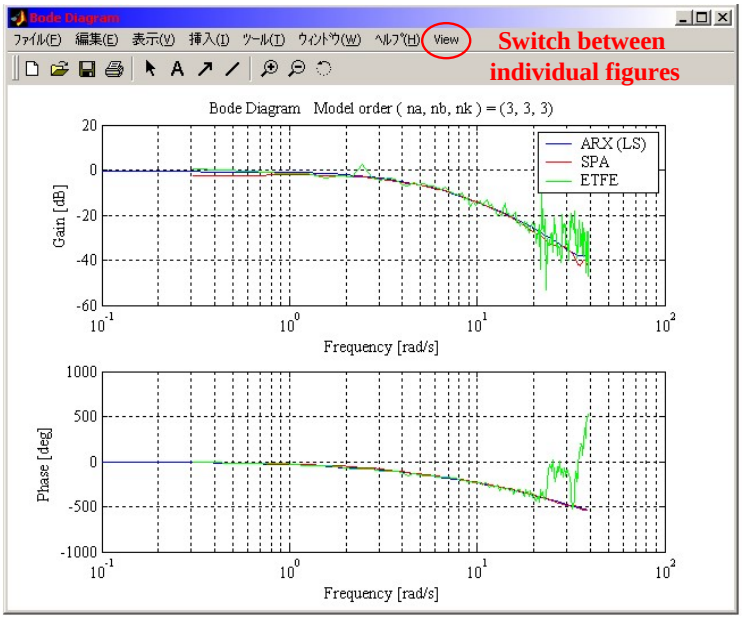

Fig. 8. Bode diagram of estimated model and non-parametric models.

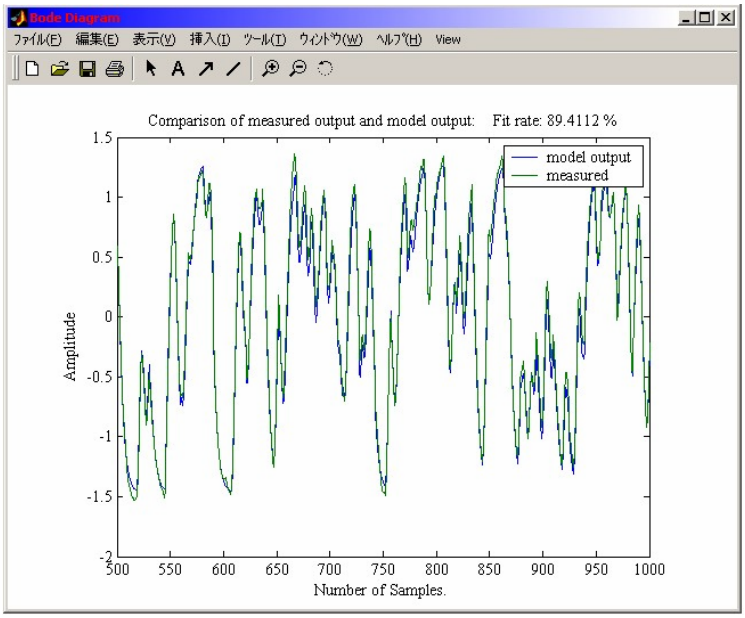

Fig. 9. Comparison of model output and measured output (validation data).

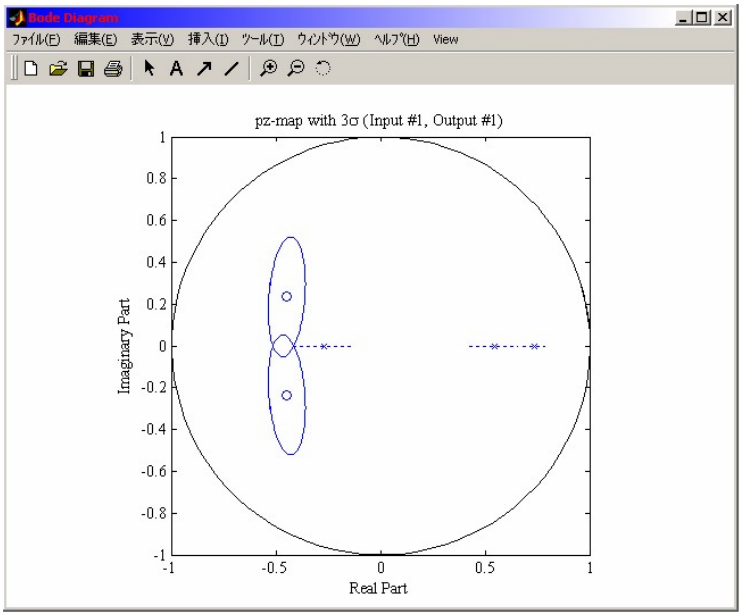

Fig. 10. Pole and zero locations of estimated model with 3σ range.

in Fig. 8 can be compared with the spectral analysis (MATLAB command "spa") model and empirical transfer function estimates (MATLAB command "etfe") (Ljung, 1999).

The models generated by the "spa" and "etfe" commands are used as references for the identified model. Fig. 9 shows cross validation, while Fig. 10 illustrates pole-zero map with a 3σ range.

These figures are switched by clicking the "View" menu in the figure window.

- Bode diagram,
- comparison of the output, and
- pole-zero map within a range of 3σ, where σ is the standard deviation.

The fit rate in Fig. 9 is the mean square fitting (MSF) of the output and is computed by

$$\mathrm{FIT}(\%) = 100 \times \left(1 - \frac{\sqrt{\sum_{k=1}^{N} (\hat{y}(k) - y(k))^2}}{\sqrt{\sum_{k=1}^{N} (y(k) - \bar{y})^2}} \right) \tag{1}$$

Where, $\hat{y}(k)$ is the model output, $y(k)$ is the measured output and $\bar{y}$ is the mean of the measured output, which is defined by

$$\bar{y} = \frac{1}{N} \sum_{k=1}^{N} y(k) \tag{2}$$

Currently, the accuracy of the estimated model is evaluated using the function given in Eq. (1) only. The MSF is calculated using all the validation data. A function for evaluating the model in the frequency domain, similar to the function defined by Eq. (1) for the time domain, is also required.

In the system identification operations, some parameters should be determined by the user, including the model structure, the model order and the sampling frequency for I/O signals. Model structures are determined from the system under study. The sampling frequency for I/O signals depend on that of the measurement system and the region of interest, which are determined in the experimental design step.

Sampling theory states that the sampled signal should contain more than $2Fs$ [Hz] frequency components if some signal that contains up to Fs [Hz] is reconstructed from the sampled data. In other words, the sampled signal with a sampling frequency of $2Fs$ [Hz] is sufficient to recover information of a signal with a frequency less than Fs [Hz]. The upper frequency of the region of interest is determined based on this theory. However, none of the region below Fs [Hz] can be recovered from the sampled signal. In system identification, the lower frequency limit is determined empirically. For example, the LS method provides reliable models between $0.01Fs$ – $0.2Fs$ [Hz] (Goodwin, 1988).

The model order should be determined based on the system structure. Users can obtain the model order using the SITB, *e.g.*, AIC (Akaike Information Criteria), MDL (Minimum Description Length) and singular value decomposition. When the system under study is a vibration structure, the model consists of a sum of second-order models. Consequently, the model order should be set as the product of second order and the number of degrees of freedom of the system.

The real order of the system is generally very high and the model describes the characteristics of interest. Since the above-mentioned guideline for the model order does not

account for the effects of disturbance, users may need to set a higher model order to obtain an accurate model.
The estimated model is saved in theta format as a mat file. Since the theta format contains information about the estimated model, including polynomial coefficients, the loss function, the final prediction error (FPE) and the sampling time, it contains sufficient information to reproduce the Bode diagram or pole-zero maps of the estimated model.

4. Conclusions and future work

In this chapter, we have described the advantages of using a GUI environment in system identification and the development of a GUI-SITB. The effectiveness of the toolbox was demonstrated by a simple example.
We confirmed the operation of the GUI-SITB only on MATLAB for a Windows platform. The GUI-SITB has been developed using MATLAB R13. The GUI-SITB may operate on the latest MATLAB version (R14 or later) with slight modification of the programs.
Since evaluation of the identification results is one of the most important parts in system identification, the evaluation method and of the system identification algorithm need to be extended to achieve this. Because the GUI-SITB currently displays results only graphically (as illustrated in Figs. 8-10), it would be desirable to implement numerical evaluation methods, one of which would display parameters of the estimated model in an appropriate format.
Furthermore, currently incomplete functions, such as the identification experiments for real plants and control system design, need to be rapidly developed. A part of the MIMO system identification procedure has been realized, but it is not yet complete.

5. References

MATLAB. http://www.mathworks.com/products/matlab/ [September, 2009]
Scilab. http://www.scilab.org/ [September, 2009]
Octave. http://www.gnu.org/software/octave/ [September, 2009]
MaTX. http://www.matx.org/ [September, 2009]
Ljung, L. (1995), *System Identification Toolbox – For Use with MATLAB (Third Printing)*, The MathWorks Inc.
Adachi S. (1996). *System Identification for Control Systems with MATLAB* (in Japanese), Tokyo Denki University Press.
Higham D. J. and Higham N. J. (2000). *MATLAB Guide*, Society for Industrial and Applied Mathematics.
Ljung L. (1999). *System Identification - Theory for the User (2nd Ed.)*, Prentice Hall PTR, Englewood Cliffs, NJ.
The MathWorks Inc. (1998). *Signal Processing Toolbox User's Guide*, The MathWorks Inc.
Adachi S. (2004). *Advanced System Identification for Control Systems with MATLAB* (in Japanese), Tokyo Denki University Press.
Van Overschee, P. and De Moor, M. (1994). N4SID: Subspace algorithm for the identification of combined deterministic-stochastic system, *Automatica*, Vol.30, pp.75-93.

Viberg, M. (1995). Subspace-based methods for the identification of linear time-invariant systems, *Automatica*, Vol.31, pp.1835-1851.

Goodwin C. G., M. E. Salgado and R. H. Middleton. (1988). Indirect Adaptive Control -An Integrated Approach, *Proceedings of American Control Conference*, pp.2440-2445.