Educational Aspects of Identification Software User Interfaces

Lennart Ljung,

Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden
WWW: <http://www.control.isy.liu.se>
E-mail: ljung@isy.liu.se, @isy.liu.se

3rd December 2003

Report no.: [LiTH-ISY-R-2562](#)

Submitted to The 13th IFAC Symposium on System Identification,
Rotterdam, The Netherlands

Technical reports from the Control & Communication group in Linköping are
available at <http://www.control.isy.liu.se/publications>.

EDUCATIONAL ASPECTS OF IDENTIFICATION SOFTWARE USER INTERFACES

Lennart Ljung*

** Division of Automatic Control, Linköping University,
SE-58183, Linköping, Sweden, email: ljung@isy.liu.se*

Abstract: Apparently many users of identification techniques learn the topic only via use of commercial software. This may or may not include the software manual. This means that the software user interface – graphical or not – plays a major role in teaching identification theory and methodology to large number of users. This contribution deals with such educational / pedagogical aspects software user interfaces.

In particular we focus on issues to hide certain design variables as defaults, and what can be done in case no defaults are obvious. Other questions are how to force the user to appreciate and understand the quality of an identified model, and to know what optional design choices and methods that are available, in particular if there is no Graphical User Interface (GUI).

1. INTRODUCTION

It is illuminating to compare sales figures of text books on System Identification, such as (Ljung 1999), (Ljung and Glad 1994), with figures of circulation of commercial software packages on Identification, such as (Ljung 2000). They show that most users of the software have not read (or at least not bought) comprehensive treatments of the topic. Combine that with the well-known fact that most software users do not open the manual, and we realize that a large number of users actually learn system identification from just running software packages.

This means that the software user interface (syntax, help texts, graphical aids, etc) plays a major educational role in teaching System Identification. It is the purpose of the current contribution to discuss such issues. Compare also with other identification software packages like (Van Overschee *et al.* 1994), (Larimore 1997), (Landau 1990), (Garnier and Mensler 2000), and (Kollar 1994).

These packages have some features in common, but have also chosen different paths to syntax and GUI-features. Not much has been written about

the educational aspects of the different choices, and it would be interesting to see more research around this topic.

2. WHAT ARE THE SALIENT FEATURES OF THE IDENTIFICATION PROCESS?

What are the most important steps/items in the identification process that any user must understand? We would like to point the the following things:

- The role of subjective judgments vs automatic procedures.
- The concept of *model validation*.
- Data requirements for reasonable results.

2.1 *Subjective and Objective Decisions*

System Identification is as much an art as a science. Models of physical processes are built and estimated for a certain purpose. It could be control design, decision making, simulation, monitoring, and several other things. Depending on the model's purpose, different aspects of the model

may be of different importance. For control design, for example, the low frequency properties of the system may be estimated with low accuracy (since the control system typically will have sensitivity function that is small at low frequencies). On the other hand, the properties around the intended cross over frequency need to be estimated with much higher accuracy.

The model quality can therefore not be summarized in a single number, but the user has to evaluate the model properties in a subjective way, taking the intended model use into account. This means, unfortunately, that there cannot be a fully automated route to the ideal model, but the user will have to interfere with subjective decisions. This is perhaps the single most important feature of the process, and it requires the user not only to know what properties to study, but also to know what measures can be taken to construct models with improved desired properties.

Unfortunately the list of model structures, possibilities, options and design variables is very long and clearly overwhelming to the newcomer. This is the nature of the pedagogical challenge in identification software: To make clear what are the choices, at the same time as not choking the user.

2.2 Model Validation

It is necessary to understand that a model is not necessarily “good” just because it is “best” within a certain chosen class of models. It is necessary to twist and turn the model to look at it from different angles to evaluate if it might be good enough for its purpose. The different techniques for this are more or less sophisticated. While a software package must contain a multitude of such options, it is important to “force” or “encourage” the inexperienced user to look at least at plots that compare model outputs with measured outputs. Such comparisons are very intuitive, immediate and instructive. They do not require any statistical understanding, except for the fact that they should be carried out on a fresh *validation* data set that was not used to estimate the model.

A special challenge of such comparisons is to make clear what the difference between “simulated” are “predicted” outputs are. For example, when building a neural network dynamical model, such as NARX, the easy and natural comparison will be between 1-step ahead model predictions and measured outputs – it is much more complicated to compute the simulated model output in typical Neural Network estimation packages. Now, a (fast sampled) model with excellent one step ahead predictions could very well be useless, but this could be a subtle message to convey.

2.3 Data Quality

One of the most common problems when inexperienced users apply identification software is that the data is not sufficiently exciting for the chosen model structure. There is both a technical difficulty and an information-theoretical issue involved in this.

It is not so difficult to use common sense to understand and explain that complicated models cannot be built from noisy, single step responses: The system must simply have had occasion to show its dynamical properties in an unambiguous way, to allow for a reasonable model. This information issue revolves around the concept of *persistence of excitation*, and there should be an early-on indication to the user how complex models can be reliably estimated from the data at hand.

There is a technical side that is more treacherous: It is quite common to see applications where the data is a single step or impulse response, where the step/impulse occurs in the first few data points. Such data could very well be used for reasonable modeling if the signal-to-noise ratio is good. However, many estimation techniques (like estimating ARX-model and subspace methods) effectively shift the data (up to a number of samples that is about the model order) in order to avoid estimation data points prior to the start of the data collection. This makes the data look like a response to a constant input, and no reasonable models can be built. It is my experience that this problem is the single most common reason for complaints about the identification process from first-time users.

3. DEALING WITH THE WORK FLOW

Identifying models from real data is as much an art as a science. The identification process involves a very large number of choices:

- Character of model structure (State-space/ ARX/ Output Error etc)
- Size of model structure (Model orders and delays)
- Choice of method (Subspace/Instrumental Variables/ Prediction error Method ...)
- Options in the method (For example: should initial filter conditions be estimated or not)
- Algorithmic aspects in the method (For example: auxiliary orders in subspace methods, parameters association with function minimization for prediction error methods)
- Pretreatment of data (Detrending/ Prefiltering/ Decimation/ etc)
- Validation process

It is indeed a challenge to make the inexperienced user aware of available options and the choices that have to be made, without leaving him/her in total confusion. Actually, to avoid confusion one simply has to enforce certain choices and hide options for a user who does not need or want to know about them.

This problem of how to guide the user with a gentle and insightful hand through the identification marsh has to be solved in different ways in a GUI (Graphical User Interface) and in a non-GUI environment.

In a GUI-environment one may hint the possibility of e.g. prefiltering, by, say, having a pop-up menu **focus**. Sooner or later the user will check the tooltip of this menu, investigate its options, and at least intuitively understand that the model fit can be focused to different frequency ranges.

It is much more difficult in a non-GUI environment to inform that such an option actually exists. It is then very helpful to have data and models represented in an object-oriented way. This will, e.g., allow suitably tailored descriptions of models and data without any specific commands. In MATLAB this is accomplished by the function **display** that is called in the appropriate object method directory for any object. An average user will also know that **get** and **set** for any object will give a succinct summary of options, without having to read help texts.

General GUI aspects are discussed in Section 5 below. In a non-GUI environment, the problem has been dealt with by a general command **advice** in (Ljung 2003a). This is further discussed in Section 4.

4. THE ADVICE FEATURE

The function **advice** in (Ljung 2003a) takes any data object or any model object as an argument and delivers a text that comments the objects and gives some hints how to proceed.

This is accomplished by a hidden property (called **utility**) of these objects, that is a structure with several fields. The MATLAB functions **assignin** and **inputname** allows you to update a workspace variable (in this case its **utility**-property) even if it is not the output of a function called. This makes it possible to adjoin to a model or a data set a “diary” of what has been done, and thus give relevant advice of further things that can be tested.

For data objects, **advice** tests things like

- excitation
- feedback in data

- if a step or impulse occurs “too early” in the set (cf the discussion in Section 2.3.)
- possible detrending
- points to the possibility to assign input inter-sample behavior
- etc

For model objects **advice**

- points out that **compare** and **resid** should be run if this has not been done. It also checks whether this was done on a separate validation data set.
- comments on what has been seen in the residual analysis test **resid**. Based on χ^2 -test it gives advice whether higher order dynamics and noise models are required. Furthermore tests are performed to check whether feedback might be present in the (validation) data.
- checks if the model order can be reduced.
- allows comparisons of several models, **advice(m1,m2,m3)** and points out which one should be preferred.
- etc

The function is best illustrated by an example of how it works: (See also (Ljung 2003b))

```
>>load iddata1
>>advice(z1)
```

All your inputs have been denoted as 'zero order hold' ('zoh'), i.e. they are assumed to be piecewise constant over the sampling interval. If the input is a sampled continuous signal and you plan to build or convert to time continuous models, it is recommended to mark the Intersample property as 'First order hold': `Data.int = 'foh'` or `Data.int = {'foh','foh', ...}` for multiinput signals. You may mix 'zoh' with 'foh' for the different inputs in the latter case.

All inputs and outputs are not zero mean. It is generally recommended to remove the means by `DAT = DETREND(DAT)`, except in the following two cases:

1. The signals are measured relative to a level that corresponds to a physical equilibrium. This could e.g. be the case if step responses are recorded from an equilibrium point.
2. There is an integrator in the system, and the input and output levels are essential to describe the effect of the integration.

There is no significant indication of feedback in the data.

The input is persistantly exciting of order more than 50. This means that you will not encounter identifiability problems if estimating models of order lower than so.

```
>>m=arx(z1,[1 1 1]);  
>>advice(m)
```

You should run a comparison test

```
COMPARE(VDATA,m)
```

where VDATA preferably is a different data set from the estimation data z1. You can then run ADVISE(m) again.

You should run a residual test

```
RESID(VDATA,m)
```

where VDATA preferably is a different data set from the estimation data z1. You can then run ADVISE(m) again.

```
>>compare(z1,m);  
>>resid(z1,m);  
>>advice(m)
```

There is a very strong indication that the dynamics of the model is not adequately described. A first general advice is to run RESID(VDATA,m,'FR') to check in which frequency ranges the model error is present. If the model error is unacceptable, you will have to increase the model order.

In particular you should pay attention to lags 2 3 6 from input 1. Modify KU and the orders NB so that these lags are included in the model.

There is a very strong indication that the residuls are not white. To get a good noise model you need to increase the orders associated with the noise parameters, or just increase the order of a state-space model.

5. GUI ISSUES

In several ways, a graphical user interface (GUI) makes it easier to make the software accessible to inexperienced users. The GUI can display choices

in various menus that point to what possibilities are available in a certain situation. More advanced design variables can be hidden deeper into nested dialogs etc.

There are two basic issues or compromises in designing a GUI. The first one concerns the trade off between

- Immediate appeal and understanding for the first time user
- Ease and convenience of use for the power user in “model production mode”

Both these properties are of course desirable, but they are partly in conflict with each other. Many tests are required to find a good balance.

The second important question when designing a GUI for system identification is whether it should be

- *process-oriented* or
- *result-oriented*.

A process-oriented GUI gives a flowchart of the identification process: first select data, then pre-process data, then select a model structure, then estimate a model, then validate the estimated model, etc. One works, so to speak “from left to right” in this flowchart. This no doubt has advantages for the first time user, but may turn out to be tiresome as one produces models *en masse* that have to be compared with each other.

In (Ljung 2003a) a result-oriented GUI is chosen. The focus is to keep an explicit view of the modified data sets and the different models that have been produces during an identification session. This is essential for the experienced user, for whom the biggest problem may be to keep track of what he/she has done. For the inexperienced user, a visually prominent board with empty slots for several models also carries the important message that the identification task is not finished just because you have constructed one model. On the other hand such a user needs more help in getting started.

Figure 1 shows the result-oriented main **Summary Board** of the GUI in (Ljung 2003a). It contains all derived data sets and all estimated models. It also has the main menus for selecting ways of modifying (preprocessing) data and for estimating models of various structures.

6. HOW TO HIDE MANY NOT-SO-IMPORTANT CHOICES?

The identification process is characterized by a myriad of choices, parameters and design variables. Just listing the choices will be utterly confusing to the newcomer. The obvious way to solve

Fig. 1. The main board of the GUI of the SYSTEM IDENTIFICATION TOOLBOX

this is to allow a syntax that does not require all choices be specified but uses defaults in most cases. This is standard practice, and is no problem is case typical defaults are data-independent. Such examples are **MaxIter** for the maximum number of iterations to perform when searching for the minimum, or **Tolerance** for the tolerance to decide when to stop iterations.

Another case is when the desired value of a design variable indeed depends on data properties. An example of this is **InitialState**: Should the initial state (filter initializations for the predictor filters) be estimated, or set to zero? The answer to this question is data/system dependent: If the true predictor filter is slowly decaying and/or the data record is short, then it may be very important to estimate the initial conditions. This fact is not so easy to explain or make visible to a new user. The solution in (Ljung 2003a) is to use the default value **auto**, which means that an automatic choice is made in the course of the calculations. Then the model property **EstimationInfo** will, after the model is estimated, contain a field that gives the value that was chosen, **Estimate**, **Zero** or **BackCast**.

Another example of this kind concerns the important choice of various horizons and weights in so called subspace identification methods. The theory of the influence of these options is not well understood even among researchers, so it would be futile to force the user to make the choices. Again the value **auto** for the corresponding properties is the solution, with an option for expert users to try explicit values. The choice **auto** also gives the further advantage that the underlying defaults can be changed as research progresses.

7. CONCLUSIONS

In this paper we have pointed to some issues in software interfaces that will guide the inexperienced user through the complex process of practical system identification. For many users that may be the only effective training in the theory and practice of identification. The educational/pedagogical aspects of the interface are thus most essential. Not much has been written about this, and more research around these questions would be welcome.

8. REFERENCES

- Garnier, H. and M. Mensler (2000). The CONTSID toolbox: A Matlab toolbox for CONTinuous-Time System IDENTification. In: *Proc. 12th IFAC Symposium on Identification*. Santa Barbara (USA).
- Kollar, I. (1994). *Frequency Domain System Identification Toolbox for use with MATLAB*. The MathWorks Inc., Natick, MA, USA.
- Landau, I. D. (1990). *System Identification and Control Design Using P.I.M. + Software*. Prentice Hall, Englewood Cliffs.
- Larimore, W. E. (1997). *AdaptX Automated System Identification Software, Users Manual*. Adaptics, Inc., 1717 Briar Ridge Road, McLean, VA.
- Ljung, L. (1999). *System Identification - Theory for the User*. 2nd ed.. Prentice-Hall, Upper Saddle River, N.J.
- Ljung, L. (2000). *System Identification Toolbox for use with MATLAB. Version 5..* 5th ed.. The MathWorks, Inc. Natick, MA.
- Ljung, L. (2003a). *System Identification Toolbox for use with MATLAB. Version 6..* 6th ed.. The MathWorks, Inc. Natick, MA.
- Ljung, L. (2003b). Version 6 of the system identification toolbox. In: *Proc. of the IFAC Conference on System Identification, SYSID'03* (P. van den Hof, Ed.). Rotterdam, The Netherlands.
- Ljung, L. and T. Glad (1994). *Modeling of Dynamic Systems*. Prentice Hall, Englewood Cliffs.
- Van Overschee, P., B. De Moor, H. Aling, R. L. Kosut and S. Boyd (1994). *Xmath interactive system identification module*. Integrated Systems, Inc., Santa Clara, CA.

Abstract

Keywords:

Avdelning, Institution

Division, Department

Division of Automatic Control

Department of Electrical Engineering

Datum

Date

2003-12-03

Språk

Language

- ☐ Svenska/Swedish
☒ Engelska/English

☐

Rapporttyp

Report category

- ☐ Licentiatavhandling
☐ Examensarbete
☐ C-uppsats
☐ D-uppsats
☒ Övrig rapport
☐

ISBN

.....

ISRN**Serietitel och serienummer**

Title of series, numbering

ISSN

1400-3902

LiTH-ISY-R-2562

URL för elektronisk version<http://www.control.isy.liu.se>**Titel**

Title

Educational Aspects of Identification Software User Interfaces

Författare

Author

Lennart Ljung, ,

Sammanfattning

Abstract

Nyckelord

Keywords