9238

An Interactive System Identification Module for Xmath (ISID)

H. Aling , U. Milletti, R.L. Kosut, M. P. Mesaros
Integrated Systems Inc., 3260 Jay St., Santa Clara, CA 95054

P. Van Overschee†, B. De Moor‡
Dept. of Electrical Engineering
Katholieke Universiteit Leuven
Kardinaal Mercierlaan 94
B-3001 Leuven (Heverlee), Belgium

Abstract

System Identification is intrinsically visualization-intensive, given the large amounts of information that need to be processed. The Interactive System IDentification (ISID) Module allows the user to visualize data and make decisions graphically by moving sliders, pushing buttons and dragging weight functions. This paper also reviews the various algorithms implemented in ISID, including state space subspace identification methods.

INTRODUCTION

The module includes algorithms that fall in two main categories. First, there are the standard classical system identification tools like prediction error methods, instrumental variables, frequency domain fit algorithms, signal analysis and model reduction.

The second main category is subspace identification algorithms. The "subspace" terminology refers to the intermediate computation of the state space as an intersection of past and future subspaces which are spanned by data regressions. The main advantage of these algorithms are robustness, ease of use and numerical reliability.

Algorithms in both categories can be used in their own right and lead to a successful identification. In addition, models obtained by the subspace methodology can be used as initial models for prediction error and maximum likelihood methods. Together the algorithms provide the numerical basis for the most powerful identification software that is currently available. The software is entirely multivariable (multiple input, multiple output).

Perhaps the most innovative component of ISID is its interactive nature. Using Xmath™'s programmable Graphical User Interface (GUI), graphical environments have been developed as interfaces to the algorithms described below. These interfaces free the user from the details of the algorithm (syntax, input types...) and allow him/her to interact graphically with the specifications and results of the identification process. In our experience this graphical environment provides a new level of productivity that cannot be obtained, regardless of how complete the function library is, from the traditional "create data->invoke function->plot results" loop. Incidentally, since the interfaces were built using the programmable GUI, source is available for all the windows and the user can choose to customize the interfaces as well as creating new ones, without having to write one line of C or C++ code.

Background on Xmath and the programmable GUI

For those not familiar with Xmath and its GUI building environment, a brief description follows.

Xmath is being developed at Integrated System Inc. (ISI) using object-oriented programming (C++) and the windowing standards of X-Windows and Motif™. It overcomes many of the problems intrinsic to Matlab-generation tools ($MATRIX_X$™ , Pro-Matlab™ ...), while building on top of the experience and routines (LINPACK and EISPACK, for example) acquired during the years.

The user commands are interpreted by a high-performance interpreted parser that executes user commands and external functions. Much attention has been paid to the user interface which is based on the Motif™ X-Windows toolkit. The management and creation of objects can be executed through forms and spreadsheets. The creation of 2D and 3D plots is mouse-driven and highly interactive, with arbitrary text placement, rotation of the plots, shading, and so on. A hypertext system is used to provide on-line help.

Xmath's object oriented architecture provides support for a variety of data types, and permits algorithm developers to take advantage of such object-oriented paradigms such as operator overload, data abstraction, and keywording.

The *programmable GUI* allows Xmath users to take full advantage of the Motif widget set from the MathScript command level (MathScript is Xmath's object-oriented scripting language). This capability makes it possible to develop and run a new class of interactive engineering design tools which "bind" graphical elements to numeric variables and algorithms.

† Research Assistant of the Belgian National Fund for Scientific Research

‡ Research Associate of the Belgian National Fund for Scientific Research

Aling H., Miletti U., Kosut R.L., Mesaros M.P., Van Overschee P., De Moor B., ``An interactive system identification module for Xmath", in *Proc. of the American Control Conference (ACC'93)*, San Francisco, California, Jun. 1993, pp. 3071-3075.,

There are two basic concepts behind the programmable GUI: *bindings* and *callbacks*. Bindings describe how graphical elements, for example sliders, push-buttons or plotted lines, are logically connected to Xmath variables. Callbacks are the mechanism by which particular portions of MathScript code are executed when the user interacts with a graphical element, for example using the mouse to modify the value of a constraint curve. The definition of the window layout, its bindings and callbacks is done completely at the scripting level, without any need for compilation.

The "classical" identification algorithms

The classical identification library is centered around parametric identification methods. Below we provide an overview of the main algorithms (with their Xmath acronyms in boldface):

- Batch Least Squares (**bls**): One of the oldest and perhaps the most robust method in system identification. All models up to a user-specified maximal order are identified and (cross-)validated, using the square root variance matrix of regressions. The parameter variance is easily obtained and converted to frequency response model uncertainty.

- Recursive Prediction Error Method (**rpem**): The implementation follows the general framework established by Ljung [1], and has been implemented for state space systems.

- Pseudo Linear Regression (**plr**): By using a sequence of least squares and linearization steps iteratively, polynomial model structures such as Box-Jenkins and ARMAX are identified.

- Empirical Transfer Function Estimation (**etfe**): The empirical transfer function estimate (a frequency response) is obtained via the spectral analysis of the system input and output time histories. The spectral density function of additive noise can also be computed.

- Generalized Instrumental Variables (**giv**): This algorithm is not based on a criterion, but on correlation of inputs and outputs with the instrumental variable. The use of an instrumental variable is particularly useful in closed loop situations, where an external reference is available.

- Impulse Response-based Realization (**imprea**): Since any system has an associated impulse response, realization based on impulse responses is extremely useful. Examples are impulse responses of high order least squares models, or impulse responses obtained as the inverse Fourier transform of an empirical transfer function estimate.

- Frequency-Weighted Least Squares (**fwls**): For problems where a model is known to be accurate only within a certain frequency range, model reduction is preferably done in the frequency domain. The standard least squares estimate is used to make a fit to the frequency response of the system.

- Maximum Likelihood (**maxlike**): In many cases, the restriction that the model has to be linear, or discrete-time, is too stringent. `maxlike` offers a general prediction error-based parameter estimation method, where the user specifies the model prediction by creating a function. Even parameterized non-linear (SystemBuild™) models can be invoked, which makes this method an important tool for the estimation of physical models which are usually nonlinear, and defined in continuous-time.

In addition to these parametric methods we have added a set of FFT-based signal analysis tools and tools for preprocessing, validation and model manipulation. Model uncertainty can be calculated with algorithms based on the estimated parameter variance. This variance is then converted to time or frequency domain responses. To obtain a more accurate model uncertainty, an asymptotic computation can also be used.

Subspace Identification Algorithms

The Numerical algorithms for State Space Subspace System IDentification (N4SID), exploit both the numerical robustness of algorithms such as the QR or the singular value decomposition, and geometrical concepts such as subspaces, projections, angles, etc., to identify state space systems. One of the central ideas of these algorithms is to reevaluate the concept of a "state" in system identification. An estimate of the state is calculated as an intersection (or a projection) between "past" and "future" input-output data [3, 5, 7, 8, 9, 10]. Afterwards, these state estimates are used to identify a state space model through a least squares solution of a set of linear equations.

The differences between the subspace approach and existing identification techniques (e.g. [4, 6]) can be grouped under three areas:

- **Conceptual importance.** Models represented in state space form are more amenable to manipulation by numerical linear algebraic algorithms as compared to their polynomial counterparts (matrix fraction descriptions, or coprime factorizations, etc.). One of the main advantages of N4SID methods is that they deliver state space models without the need for an explicit parameterization.

- **Robustness and sensitivity.** The *subspace* idea is directly related to the fact that no explicit parameterization is required. By not insisting on a specific parameterization, we obtain state space models that are more robust to inaccuracies on the data (noise, finite precision arithmetic) than parameterized 'canonical' models. A disadvantage is that, if the exact structure of the state space matrices is (partially) known, then this knowledge can not be incorporated as a constraint in the N4SID algorithms.

- **Computational issues.** The subspace identification algorithms use "tall but skinny" matrices. Through a recursive QR decomposition, the tall dimension is eliminated immediately (in a square root, numerically reliable way). Since the dynamics of systems can be determined from certain subspaces, algorithms such as QR and the singular value decomposition are the ideal natural tools to characterize these subspaces numerically. N4SID methods thus avoid explicit squaring up of the problem. They are essentially data driven without intermediate, possibly ill-conditioned numerical steps. Finally, N4SID methods are not based on possible nonlinear optimization approaches. In this way, they are always bound to give results without the danger of convergence problems or slow convergence rates. They are thus very well suitable to

generate initial models for more sophisticated "optimal" estimators like prediction error methods.

There are two *Subspace Algorithms* implemented in ISID:

- Subspace Deterministic-Stochastic system identification (sds): Identification of general combined deterministic-stochastic state-space system from input-output data.

- Subspace Stochastic system identification (sst): Identification of stochastic state-space systems from output data (time-series analysis). The observed input u is not present.

Both algorithms can be set to be either sensitive to the scaling of the input-output data (energy based) [9, 10], or insensitive to scaling (angle and direction based) [3,7,8]. No *a priori* knowledge of the system order and/or system parameterization is needed. The system order is determined through the inspection of dominant singular values (scaling sensitive) or principal angles (scaling insensitive).

A "software road map" of the System Identification Module is shown in figure 1. Objects have been added to those already pre-defined in Xmath (see [2]) to provide a familiar environment for the user. These identification objects can be categorized in four different levels, from top to bottom: identification data sets, intermediate results, models and validation results. To keep the picture simple, we have only displayed the main commands.

Several algorithms require the computation of an intermediate result which is computationally intensive. From the intermediate, a set of models up to a certain order can be extracted by simple matrix transformations. This is particularly useful for large data sets and high model orders where re-computation of the intermediate results is prohibitive. An example of intermediate result is the square root of the data regression variance matrix which forms the basis for a set of ARX models. This square root can also be passed as an input to sds. Square root information from several data sets can also easily be combined,

SYSTEM OBJECTS AND COMMAND STRUCTURE

One advantage of having an object-oriented system identification environment is that it is much easier to define and store different model types than what can be stored in a matrix. One example is a frequency response that would be stored as 2 matrices (frequency range vector and frequency response matrix) in Matlab-generation packages. In Xmath frequency responses are stored in one object, the Parameter Dependent Matrix (PDM). The same object can also store input/output name labels, and a label for the domain (frequency). Operations on PDMs are also greatly simplified; for example, inverting a frequency response is a simple as invoking `inv(fresp)` (in Matlab-generation packages one would have to carefully write for loops). The use of objects leads to an impressive reduction in code size and complexity, as well as faster execution.

ISID defines some new objects for System Identification. It uses four different object classes for system representation:

- **Standard systems vs. innovations models.** The standard system type represents a single input/output relationship

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

$$y_k = Cx_k + Du_k$$

Innovations models contain an additive filtered noise component:

$$x_{k+1} = Ax_k + Bu_k + Ke_k \tag{2}$$

$$y_k = Cx_k + Du_k + e_k$$

$$Ee_k e_l = \delta_{k,l}\,\Sigma$$

Here, e_k is a white noise sequence of which the variance Σ is also part of the prediction model. In addition to the I/O behavior, this model representation describes the second order noise properties. Innovations models are directly related to Kalman filters. For notational convenience, we will refer to (1) as the "deterministic" and to (A,K,C,Σ) as the "stochastic" part.

- **Polynomial systems vs. state space systems.** Many identification algorithms can more easily work with polynomial models than state space. In particular, prediction based methods like Batch Least Squares (BLS), Pseudo Linear Regression (PLR) and Instrumental Variables (IV) fall under this category. The basic polynomial system is of the form:

$$A(z)\,y_k = B(z)\,u_k \tag{3}$$

where $A(\cdot)$ and $B(\cdot)$ are polynomial matrices in the backward shift operator. The relationship between model representations (1) and (3) is well understood [11, 12, 13]. A $C(z)e_k$ term can be added to (3) to have the polynomial equivalent of (2).

The advantages of using these system objects are the same as those outlined earlier for PDMs. The standard state-space system model (1) will, for examples, contain matrices A,B,C,D, as well as optional fields such as initial conditions X0, input/output/state name labels and sampling time dt (0 means the system is continuous-time). Objects greatly facilitate data management (self-documenting, fewer variables) and substantially compact code (if `A` and `B` are state-space systems, an expression like `A*B/(A+B)` computes just what one would expect).

FUNCTIONS

The main ISID algorithms can be invoked with the {gui} keyword to request the interactive version of the function. The interactive interface consists of one or more windows which display results like frequency response or error norms, and pulldown menus that give access to various options. As an example of how Xmath's programmable GUI has been used to make an interactive graphical system identification tool, consider the batch least squares (`bls`) identification tool shown in figure 2.

At the top, there is a menu bar with hierarchically organized pulldown menus which is laid out consistently across the

various interactive tools. The File pull-down allows the user to exchange data between the interactive tool and the Xmath stack. Examples are saving plots and identified models to Xmath, and loading reference ('true') models or data to use as a benchmark for the identified model. The Algorithm pull-down menu contains options that are specific to the tool, for example display of error norm or singular value plots, and invocation of the mouse-driven frequency weighting specification. The Validation pull-down lets the user select how to view of the data, including input-output data, predicted output data, various forms of prediction errors, frequency response, impulse response, pole-zero location and error bounds. The appearance of the graphics area can be controlled through the Plot menu, and extensive on-line help is provided with each interactive tool, and can be accessed through the Help pull-down.

Below the plotting area the user can find identification 'control knobs', including system order and structure, algorithm control and data channel selection. The sampling interval, Nyquist frequency and number of data points are always shown at the bottom of the window.

The interactive nature of the ISID tools does not end at pull-down menus and push-buttons, but also extends to the graphics area. A mouse-controlled magnifying glass can be used to examine the data more in detail, and specific values can be read off the curves by a right mouse click and hold. Mouse-driven zooming can also be invoked by lassoing the area of interest (axes ranges are dynamically updated).

CONCLUSIONS

ISID brings a new dimension to System Identification software. Its complete library of functions (including both "classical" and subspace algorithms) and its revolutionary interactive nature create a powerful combination of ease-of-use and completeness. We hope other researchers will be inspired by our work and produce many other interactive environments for disparate applications.

ACKNOWLEDGEMENTS

We would like to thank Prof. Stephen Boyd for his encouragement and suggestions and Dr. Craig Barratt for his expert technical advice.

$MATRIX_X$,SystemBuild, and Xmath are trademarks of Integrated Systems, Inc. Pro-Matlab is a trademark of The Mathworks. Motif is a trademark of the Open Software Foundation.

REFERENCES

[1] Ljung L., Söderström T. *Theory and practice of recursive identification*, MIT Press, 1983

[2] Floyd M.A., Dawes P.J., Milletti U. *Xmath: A new generation of Object-Oriented CACSD tools.* Proceedings of the first European Control Conference, Grenoble, France, July 1991, pp. 2232-2237

[3] Larimore W. *Canonical Variate Analysis in Identification, Filtering, and Adaptive Control.* 29th IEEE Conference On Decision and Control, Honolulu, Hawaii, December 1990, pp. 596-604.

[4] Ljung L. *System identification; Theory for the user.* Prentice-Hall Information and System Sciences Series, 1987.

[5] Moonen M., De Moor B., Vandenberghe L., Vandewalle J. *On- and off-line identification of linear state space models.* International Journal of Control, Vol. 49, No. 1, January 1989, pp. 219-232.

[6] Söderström T., Stoica P. *Instrumental Variable Methods for System Identification.* Lecture Notes in Control and Information Sciences, Springer-Verlag, New York, 1983.

[7] Van Overschee P., De Moor B. *Subspace Algorithms for the Stochastic Identification Problem.* Proceedings CDC 1991 (pp 1321-1326), Brighton (Eng land), December 11-13, 1991.

[8] Van Overschee P., De Moor B. *Subspace Algorithms for the Stochastic Identification Problem.* ESAT-SISTA Report 1991-29, Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium. Accepted for publication in Automatica.

[9] Van Overschee P., De Moor B. *N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems.* Proceedings CDC 1992, Tucson Arizona (USA), December 16-18, 1992.

[10] Van Overschee P., De Moor B. *Two Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems.* ESAT-SISTA Report 1992-34, Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium. Accepted for publication in Automatica, Special Issue on Statistical Processing and Control.

[11] Janssen P.H.M. *General results on the McMillan degree and the Kronecker indeces of ARMA and MFD models,* Int. J. Control, vol. 48 no. 2, pp. 591-608, 1988

[12] Kailath T. *Linear Systems,* Prentice-Hall, 1980

[13] Gevers M.R. *ARMA models, their Kronecker indeces and their McMillan degree,* Int. J. Control, vol 43 no. 6, pp. 1745-1761, 1986

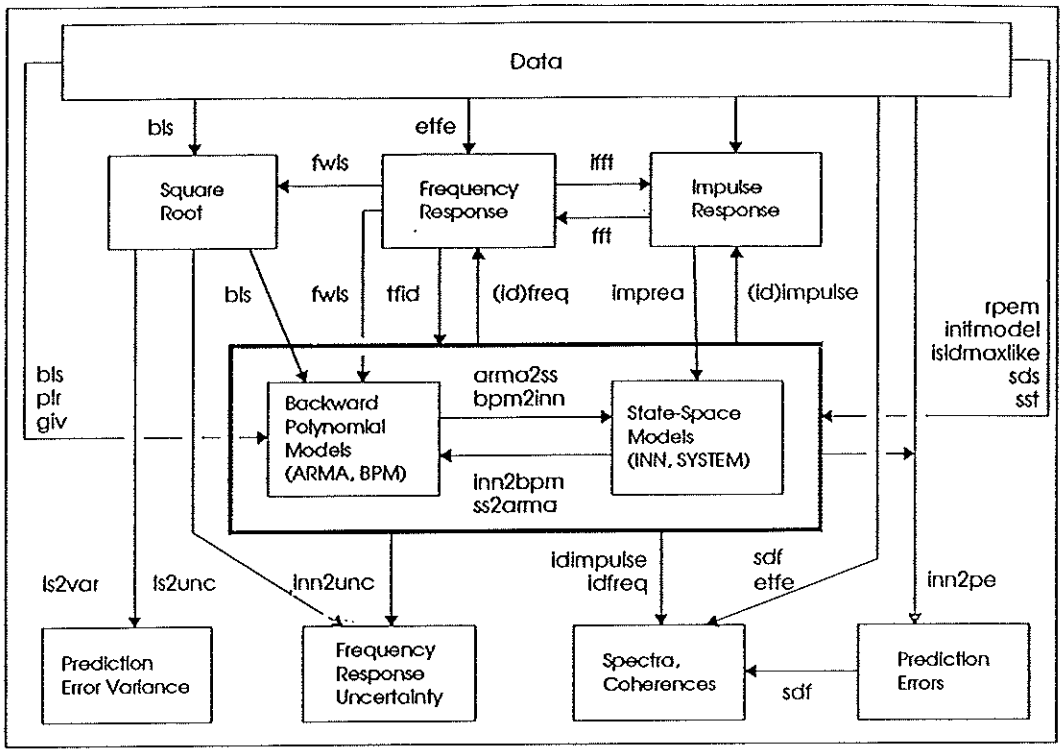

Figure 1. Software Road Map

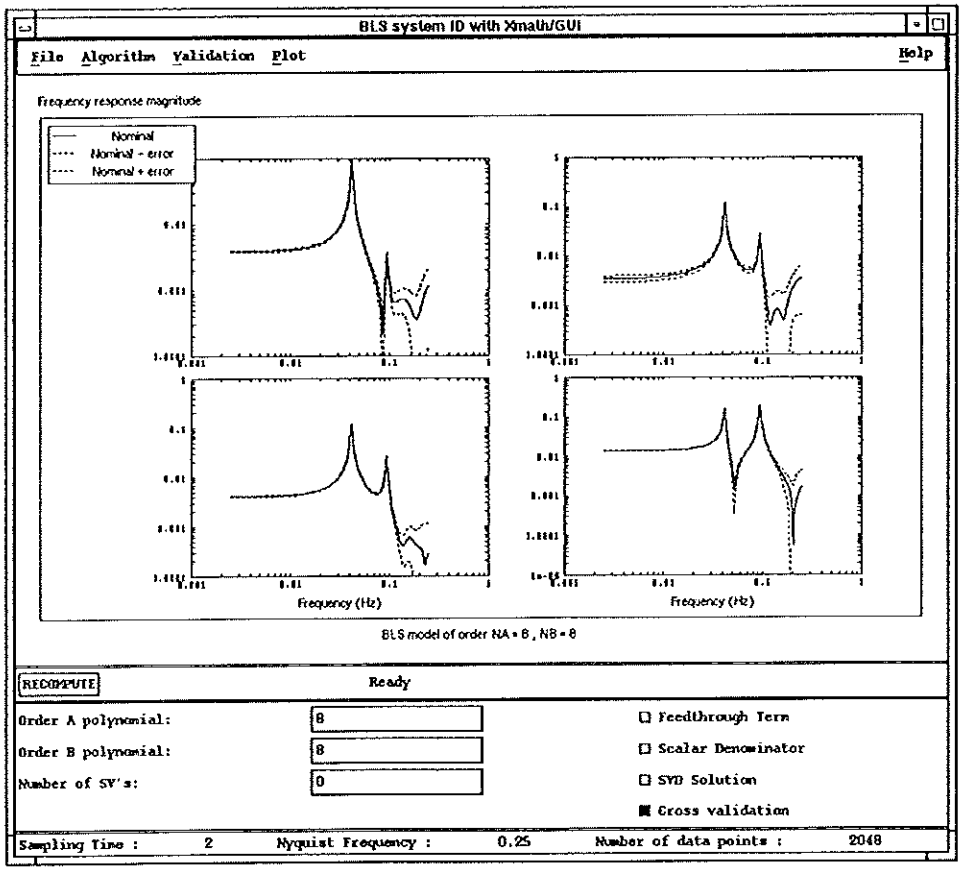

Figure 2. BLS interactive tool identifying an 8th-order Model and displaying error bounds