

A MATLAB Software Environment for System Identification ^{*}

Adrian Wills^{*} Adam Mills^{*} Brett Ninness^{*}

^{*} School of Electrical Engineering and Computer Science, University of Newcastle, Callaghan, NSW, 2308, Australia (Tel: +61 2 49216028; Corresponding author e-mail: adrian.wills@newcastle.edu.au).

Abstract: This paper describes a Matlab based software environment for the estimation of dynamic systems. It has been developed primarily as a vehicle for profiling novel approaches relative to existing methods within a common software framework in order to streamline comparisons. Key features of the toolbox include simplicity of use (particularly via automated entry of unspecified values) and the support of a wide range of scalar and multivariable model structures. The development of this software is an ongoing project, with earlier progress being reported on previously. This paper details recent advancements, including the provision of a graphical user interface environment.

Keywords: gradient-based search, output-error, Hammerstein, Wiener, black-box

1. INTRODUCTION

This paper details system identification software developed to run under a Matlab Mathworks [2004] environment. It offers a suite of methods which have become standard tools within the system identification community. These principally include least-squares and subspace-based techniques in combination with shift operator transfer function and state space model structures. Both time and frequency domain data can be accommodated in these contexts.

More interestingly, the software implements several new approaches which the authors have found to be effective. These include the Expectation Maximisation (EM) algorithm for computation of Maximum Likelihood estimates Gibson and Ninness [2005], Gibson et al. [2005], the use of an adaptive Jacobian rank algorithm Wills and Ninness [2008] in gradient based search for least squares estimates and, in some cases, the use of a delta operator model Middleton and Goodwin [1990]. As well, a range of non-linear model structures including those of bilinear, and Hammerstein–Wiener type are supported.

Earlier versions of this software have been reported on previously Ninness and Wills [2006]. This paper details new developments. These include the ability to directly estimate continuous time models from irregularly sampled data, the capacity to accommodate “grey-box” and frequency domain multivariable state space models, and the provision of a graphical user interface environment.

Before presenting these new features, an overview of the use of the software and its capabilities will be provided.

2. SOFTWARE OVERVIEW

The software is designed to facilitate the estimation of a wide range of model structures based on either time or frequency domain data.

For this purpose, two essential inputs from the user are a MATLAB structure **Z** specifying the observed data, and structure **M** specifying the required model structure form. The toolbox then returns an estimated model in a MATLAB structure **G** via a call to the `est` function. For example

```
>> Z.y=y; Z.u=u; M.A=4; G = est(Z,M);
```

specifies that the observed input and output come from vectors **u**, **y** in the MATLAB workspace, and that a model of order 4 is required.

In this example, no particular model type is specified. A feature of the software is that, in the interests of streamlining its use, defaults are applied as much as possible in case the user has not made a full specification.

For example, the commands above invoke a default transfer function model structure of general type Box–Jenkins type (**θ** is a parametrizing vector)

$$y_t = G(q, \boldsymbol{\theta})u_t + H(q, \boldsymbol{\theta})e_t = \frac{B(q, \boldsymbol{\theta})}{A(q, \boldsymbol{\theta})}u_t + \frac{C(q, \boldsymbol{\theta})}{D(q, \boldsymbol{\theta})}e_t \quad (1)$$

being employed, but in fact with no noise model $H(q, \boldsymbol{\theta}) = C(q, \boldsymbol{\theta})/D(q, \boldsymbol{\theta})$ so that the default is in fact of Output-Error form. If in fact this full Box–Jenkins structure were to be used with $D(q, \boldsymbol{\theta})$ of (for example) 2nd order, then this can be achieved by

```
>> Z.y=y; Z.u=u; M.A=4; M.D=2; G = est(Z,M);
```

Returning to the previous case, the default OE model choice can be discovered by the user using the `details` command to investigate the returned model **G**:

^{*} This work was supported by the Australian Research Council.

 Details for Estimated Model Structure

Operator used in model = q
 Sampling Period = 1.000000 seconds
 Estimated Innovations Variance = 1.094039e-02
 Model Structure Used = Output Error
 Estimation algorithm = Gauss-Newton search
 Input #1 block type = linear
 Output block type = linear

 Input #1 to Output #1 Estimated T/F model + standard devs:

1 q⁻¹ q⁻² q⁻³ q⁻⁴
 B = 0.0052 -0.0103 0.0054 -0.0051 0.0096
 SD= 0.0048 0.0117 0.0158 0.0142 0.0068

1 q⁻¹ q⁻² q⁻³ q⁻⁴
 A = 1.0000 -2.0384 0.7937 0.6297 -0.3800
 SD= 0 0.5763 1.5343 1.3773 0.4177

delay = 0 samples

Poles at 0.8566*exp(+j0.1329), -0.5693, 0.9097.

This reveals that other default choices have been made, such as the underlying sampling period being $M.T = 1$ seconds, and the delay on the input being $M.delay = 0$ seconds. Of course, if these elements of the model structure M had been set otherwise, they would have overridden the defaults.

Likewise, the model structure type can be set, for example, to estimate an ARX by issuing the commands:

`M.type = 'arx'; G = est(Z,M);`

Other possibilities for `M.type` are `arma`, `armax`, `fir` and `bj` and `ss`. Most of these are self explanatory (`bj` refers to Box-Jenkins) and are special cases of the transfer function structure (21).

However, a specification of `M.type='ss'` will result in the following “innovations form” state space structure being employed

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_t \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{K} \\ \mathbf{I} \end{bmatrix} \mathbf{e}_t. \quad (2)$$

Here, both $\mathbf{u}_t \in \mathbf{R}^m$ and $\mathbf{y}_t \in \mathbf{R}^l$ may be vectors so that the multiple input, multiple output (MIMO) scenario can be accommodated in this way.

In this situation, and in the transfer function cases mentioned earlier, the actual estimate $\hat{\boldsymbol{\theta}}_N$ of the vector $\boldsymbol{\theta}$ parametrizing the model structure is a prediction error estimate defined as the solution to the optimisation problem

$$\hat{\boldsymbol{\theta}}_N \triangleq \arg \min_{\boldsymbol{\theta} \in \mathbf{R}^n} V_N(\boldsymbol{\theta}) \quad (3)$$

involving the cost function ($*$ denotes conjugate transpose)

$$V_N(\boldsymbol{\theta}) \triangleq \text{Trace}\{E(\boldsymbol{\theta})^* E(\boldsymbol{\theta})\} \quad (4)$$

where $E(\boldsymbol{\theta})$ is vector

$$E^T(\boldsymbol{\theta}) \triangleq [\boldsymbol{\varepsilon}_1(\boldsymbol{\theta}), \dots, \boldsymbol{\varepsilon}_N(\boldsymbol{\theta})] \quad (5)$$

with elements $\boldsymbol{\varepsilon}_t(\boldsymbol{\theta})$ being the prediction error

$$\boldsymbol{\varepsilon}_t(\boldsymbol{\theta}) \triangleq \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}(\boldsymbol{\theta}) \quad (6)$$

where $\hat{\mathbf{y}}_{t|t-1}(\boldsymbol{\theta})$ is the (mean square optimal) one step ahead prediction of \mathbf{y}_t conditional on experimental observations up to and including time $t - 1$. For the state space case (2), this predictor is computed via the Kalman Filter,

and in the transfer function case (21), it is computed by the steady state Kalman filter in transfer function form Ljung [1999].

Solving the optimisation problem (3) in order to compute $\hat{\boldsymbol{\theta}}_N$ is achieved via a gradient based search method which is of Gauss-Newton type, and has been presented in detail in Wills and Ninness [2008].

However, in this case of state-space model structure other estimation methods and algorithms may be specified via the inclusion of a third argument to the `est` operation. This argument, in this paper shall be called OPT to recognise it being designed to provide input of optional choices. As an example of its use, the command

`OPT.alg='n4sid'; G=est(Z,M,OPT);`

will change the algorithm used to estimate the state space model structure (2) from the default (3)-(6) to the so-called N4SID subspace based method van Overschee and de Moor [1994]. Furthermore, a specification `OPT.alg='cca'` will result in an alternate canonical-correlation subspace based method Larimore [1990] being used.

Finally, a specification of `OPT.alg='em'` will cause the software to use the Expectation-Maximisation (EM) algorithm described in Gibson and Ninness [2005] to be employed to estimate a state space structure.

Multivariable data can also be addressed using transfer function models which are an extension of (21) to the multiple input, single output (MISO) form

$$y_t = \sum_{k=1}^m G_k(q, \boldsymbol{\theta}) u_t^k + H(q, \boldsymbol{\theta}) e_t. \quad (7)$$

For example, if three input records $u_t^1 - u_t^3$ are collected as three columns of a vector \mathbf{u} which are thought to affect one output record recorded in a vector \mathbf{y} . Then the command

`Z.y=y; Z.u=u; M.A=[4;2;3]; M.D=1; G=est(Z,M);`

will deliver a model estimated using the prediction error criterion (4) and consisting of estimates $G_1(q, \hat{\boldsymbol{\theta}})$, $G_2(q, \hat{\boldsymbol{\theta}})$ and $G_3(q, \hat{\boldsymbol{\theta}})$ being of orders 4, 2 and 3 respectively, with $H(q, \boldsymbol{\theta})$ being first order.

In addition to these linear model structures, the software also supports the estimation of certain non-linear classes. For example, MISO Hammerstein-Wiener models are supported of the following form

$$z_t = \sum_{k=1}^m G_k(q, \boldsymbol{\theta}) X_k(u_t^k, \boldsymbol{\theta}) \quad (8)$$

$$y_t = Z(z_t, \boldsymbol{\theta}) + H(q, \boldsymbol{\theta}) e_t \quad (9)$$

where $X_k(\cdot, \boldsymbol{\theta})$, $Z(\cdot, \boldsymbol{\theta})$ are memoryless non-linearities parametrized by $\boldsymbol{\theta}$. Therefore if, for example, the user wished to progress beyond the previous 3 input MISO transfer function estimation to one that included a dead-zone, saturation, and polynomial memoryless non-linearity X_1, X_2, X_3 on inputs numbered one to three, and also included a piecewise linear memoryless non-linearity Z , then this can be accommodated by augmenting the above model structure as follows:

`M.in(1).type='deadzone'; Min(2).type='saturation';`

```
M.in(3).type='poly'; M.out.type='hinge';
G=est(Z,M);
```

A final class of nonlinear models accommodated by the software is the state-space bilinear extension of the linear structure (2) of the following form Favoreel et al. [1999]

$$\begin{bmatrix} \mathbf{x}_{t+1} \\ \mathbf{y}_t \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{F} & \mathbf{B} \\ \mathbf{C} & \mathbf{G} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \otimes \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{K} \\ \mathbf{I} \end{bmatrix} \boldsymbol{\epsilon}_t \quad (10)$$

where \otimes represents the Kronecker tensor product Brewer [1978]. This structure is employed by setting

```
M.type = 'bilin'
```

and the default estimation method in this case is Gauss-Newton type gradient based search (i.e. `OPT.alg='gn'`) to deliver the prediction error solution (3)- (6). Setting `OPT.alg='em'` in this bilinear case will alter this by using the EM algorithm to compute (under Gaussian assumptions) a Maximum-Likelihood solution. For further details on these methods, the reader is referred to Gibson et al. [2005].

Finally, the software also supports linear model estimation from an observed frequency response. Specifically, if a set of (complex valued) frequency responses $\{Y(\omega_k)\}$ of a linear system to a unit amplitude sinusoidal input at frequencies $\{\omega_k\}$ are stored in vectors \mathbf{y} and \mathbf{w} , then the command

```
Z.y=y; Z.w=w; M.A=4; G=est(Z,M);
```

will deliver a MATLAB structure \mathbf{G} which employs the output-error model structure

$$Y(\omega_k) = G(e^{j\omega_k}, \boldsymbol{\theta}) + e_k = \frac{B(e^{j\omega_k}, \boldsymbol{\theta})}{A(e^{j\omega_k}, \boldsymbol{\theta})} + e_k \quad (11)$$

of order 4. By default, a discrete time model (`M.op='q'`) is assumed so that $\gamma_k = e^{j\omega_k}$ and the frequencies ω_k are assumed to be normalised with respect to the sampling period. However, setting `M.op='s'` will result in $\gamma_k = j\omega_k$ being used in which case the frequencies ω_k are the actual measured ones.

The delivered parameter estimate $\hat{\boldsymbol{\theta}}_N$ in this case is again computed via the Gauss-Newton search developed in Wills and Ninness [2008] to be one that satisfies the non-linear least squares criterion (3)-(5), but in this frequency domain case the estimation error is taken as

$$\varepsilon_k = Y(\omega_k) - G(e^{j\omega_k}, \boldsymbol{\theta}). \quad (12)$$

A state space model structure of the form

$$G(\gamma_k, \boldsymbol{\theta}) = \mathbf{C}(\gamma_k \mathbf{I} - \mathbf{A})^{-1} \mathbf{B} + \mathbf{D} \quad (13)$$

is used in place of the transfer function one (12) by setting `M.type='ss'`. As in the time domain case, when employing a state space structure further estimation algorithms beyond the least squares one (3)-(5) are obtainable by setting `OPT.alg`.

For example, `OPT.alg='sid'`, `G=est(Z,M,OPT)` will result in the estimated state space model being computed using the frequency domain subspace identification methods developed in McKelvey et al. [1996b,a]. Furthermore, a choice of `OPT.alg='em'` will use this subspace estimate as an initial point and then use the EM algorithm to refine it towards a Maximum-Likelihood solution, as developed in Wills et al. [2008].

Finally, while `details(G)` provides one method for assessing an estimated model \mathbf{G} , there are further tools provided by the software. The commands `showbode(G)` and `shownyq(G)` display the frequency response(s) of the model \mathbf{G} using (respectively) Bode and Nyquist plots.

Furthermore, the command `validate(Z,G)` will perform a validation of the model \mathbf{G} against the data \mathbf{Z} , which need not be the data used for estimation of \mathbf{G} . This involves the generation of several measures and associated displays, such as observed measurements versus model \mathbf{G} predictions, the auto-correlation of the residuals $\{\varepsilon_t(\hat{\boldsymbol{\theta}}_N)\}$ together with 95% confidence bounds assuming whiteness, and the cross-correlations between predicted output $\{\hat{\mathbf{y}}_{t|t-1}(\hat{\boldsymbol{\theta}}_N)\}$ and observed input $\{\mathbf{u}_t\}$.

This brief overview has presented only the essentials of using the software suite. Further details of the use of the `est(Z,M,OPT)` function and the options which may be passed to it via \mathbf{M} and `OPT` may be discovered by typing `help est` at the MATLAB command line.

3. NEW DEVELOPMENTS

There have been several significant new developments with the software since it was last reported on in Ninness and Wills [2006] which will now be detailed.

3.1 Irregularly Spaced Samples and Continuous Time Models

In dealing with time domain data, the software description to this point has dealt only with shift operator models that assume data samples collected at regularly spaced sampling instants.

A recent development with the software has been the capability to handle time domain data collected with respect to an arbitrary sampling regime. Specifically, consider the case if observed input $\mathbf{u}(t) \in \mathbf{R}^m$ and output $\mathbf{y}(t) \in \mathbf{R}^\ell$ which are sampled at time points $t = \{t_1, \dots, t_N\}$ to deliver the records $\{\mathbf{u}_k\}, \{\mathbf{y}_k\}$. Suppose these are collected in MATLAB matrices \mathbf{u} and \mathbf{y} , together with a third vector \mathbf{t} specifying the associated sampling points $\{t_1, \dots, t_N\}$.

Then the following commands will estimate a 5'th order model

```
>> Z.t = t; Z.u = u; Z.y = y; M.A=5;
>> G=est(Z,M);
```

which can be examined as usual via the `details(G)` command to deliver:

```
-----
Details for Estimated Model Structure
-----
Operator used in model      = s
Sampling Period            = irregular
Estimated Innovations Variance = 8.617693e-03
Model Structure Used       = State Space
Estimation algorithm       = Gauss-Newton search
Input #1 block type        = linear
Output block type         = linear
-----
```

This reveals that the software handles this irregularly spaced data situation by employing a continuous time (the operator is reported as `s`) state space model which is of the following form:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \quad (14)$$

$$\mathbf{y}_k = \frac{1}{\Delta_k} \int_{t_k}^{t_k + \Delta_k} [\mathbf{C}\mathbf{x}(t) + \mathbf{e}(t)] dt. \quad (15)$$

An essential feature of this continuous time model is that via (15), the measured output sample is assumed to be obtained via integration of the underlying continuous signal $\mathbf{y}(t)$ over a period $\Delta_k \leq t_{k+1} - t_k$. The reasoning underpinning this assumption and its consequences are provided in Ljung and Wills [2008].

Furthermore, the terms $\mathbf{w}(t) \in \mathbf{R}^n$ and $\mathbf{e}(t) \in \mathbf{R}^\ell$ in the model structure (21), (15) are assumed to be continuous-time white noise processes with incremental covariance

$$\text{Cov} \left\{ \begin{bmatrix} d\mathbf{w}(t) \\ d\mathbf{e}(t) \end{bmatrix} \right\} = \begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{R} \end{bmatrix} dt. \quad (16)$$

The estimation of this model structure is achieved by employing the estimation method (3), (5) but with $E(\boldsymbol{\theta})$ modified in order to realise a Maximum-Likelihood criterion under Gaussian assumptions on $\mathbf{w}(t), \mathbf{e}(t)$. The estimate $\hat{\boldsymbol{\theta}}_N$ is again computed by the Gauss-Newton type gradient-based search detailed in Wills and Ninness [2008].

3.2 Multivariable Frequency Domain Modelling

A further new development has been the extension to multivariable modelling of frequency domain data. This involves extending the single input, single output (SISO) state space representation (13) to

$$\mathbf{G}(\gamma_k) = \mathbf{C}(\gamma_k \mathbf{I} - \mathbf{A})^{-1} \mathbf{B} \mathbf{U}_k + \mathbf{D}; \quad \gamma_k = e^{j\omega_k} \text{ or } j\omega_k \quad (17)$$

In this model the quantities

$$\{\mathbf{Y}(\omega_k) \triangleq \mathbf{Y}_k\}_{k=1}^N; \quad \mathbf{Y}_k \in \mathbf{C}^{p \times m} \quad (18)$$

are, as in the SISO case, measurements of a system's frequency response at a set $\{\omega_k\}_{k=1}^N$ of frequencies that need not be equally spaced.

The interpretation of the elements in the matrix \mathbf{Y}_k is that the i, ℓ 'th element is the response at the i 'th output on the ℓ 'th excitation experiment.

Note that relative to the SISO case (13), the multivariable model (17) is augmented by allowing for measurements

$$\{\mathbf{U}(\omega_k) \triangleq \mathbf{U}_k\}_{k=1}^N; \quad \mathbf{U}_k \in \mathbf{C}^{m \times m} \quad (19)$$

of the input spectrum. The interpretation of the elements in the matrix \mathbf{U}_k is that the i, ℓ 'th element is the excitation on the i 'th input during the ℓ 'th experiment [Pintelon and Schoukens, 2001, Section 2.7].

3.3 Grey Box Modeling

The software has now been extended to provide the capability for so-called ‘‘greybox’’ modeling of state-space systems. These are ones that are of the linear form (2), but with an arbitrary mapping

$$\boldsymbol{\theta} \mapsto \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{K}\} \quad (20)$$

from parameter vector $\boldsymbol{\theta}$ to state space system matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{K}\}$.

To achieve this, the user is required to supply a function that performs the above mapping. The name of this function must be specified in `M.t2m`. Similarly, the user

must also provide a function that maps from the system matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{K}\}$ back to $\boldsymbol{\theta}$ and specify the name of this function in `M.m2t`.

By way of example, suppose $\boldsymbol{\theta} \in \mathbf{R}^6$ has the following mapping to $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{K}\}$

$$\mathbf{A} = \begin{bmatrix} \theta_1 & 0 \\ \theta_2 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \theta_3 \\ \theta_4 \end{bmatrix}, \quad \mathbf{C} = [1 \ 0], \quad \mathbf{D} = 0, \quad \mathbf{K} = \begin{bmatrix} \theta_5 \\ \theta_6 \end{bmatrix}. \quad (21)$$

Then the following function definition specifies the ‘‘forward’’ mapping to be specified as `M.t2m='ttom'`

```
function M = ttom(M,theta)
M.ss.A = [theta(1) 1; theta(2) 0];
M.ss.B = [theta(3);theta(4)];
M.ss.C = [1 0];
M.ss.D = [];
M.ss.K = [theta(5);theta(6)];
```

while the ‘‘reverse’’ mapping to be specified as `M.m2t='mtot'` is given by.

```
function theta = mtot(M)
theta(1) = M.ss.A(1,1);
theta(2) = M.ss.A(2,1);
theta(3) = M.ss.B(1);
theta(4) = M.ss.B(2);
theta(5) = M.ss.K(1);
theta(6) = M.ss.K(2);
theta = theta(:);
```

In addition to providing these function names, the user must set `M.type = 'ss'` and `M.par = 'grey'` to specify and state-space model with a user defined parametrization. Assuming a data-set Z and an initial estimate of $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{K}\}$ in `M.ss.A,...,M.ss.K`, then the following commands would estimate $\boldsymbol{\theta}$ for (21)

```
>> M.type = 'ss';
>> M.par = 'grey';
>> M.t2m = 'ttom';
>> M.m2t = 'mtot';
>> G = est(Z,M);
```

The returned estimate is obtained via the prediction error method (4)-(6) computed using the Gauss-Newton gradient based search developed in Wills and Ninness [2008]. Numerical differentiation is used to obtain

$$\frac{\partial \mathbf{A}}{\partial \theta_i}, \dots, \frac{\partial \mathbf{K}}{\partial \theta_i} \quad (22)$$

Note that this method is exact (to machine precision) when the mapping from θ_i to an system matrix element is linear (as in the above example).

3.4 Graphical User Interface (GUI)

A major new feature of the software has been the development of a graphical user interface (GUI) environment. This is designed to streamline the process of handling combinations of different data sets, model structures and estimates.

For reasons of stability and portability it is written in Java rather than using the proprietary MATLAB ‘‘HandleGraphics’’ functionality. It consists of four main tabbed ‘‘panes’’ that lead the user through the four steps of

- (1) Data specification;
- (2) Model structure specification;
- (3) Model estimation, possibly including algorithm choice;
- (4) Model validation and comparison.

The first data specification pane is illustrated in Figure 1. This illustrates (top half of pane) data being loaded in

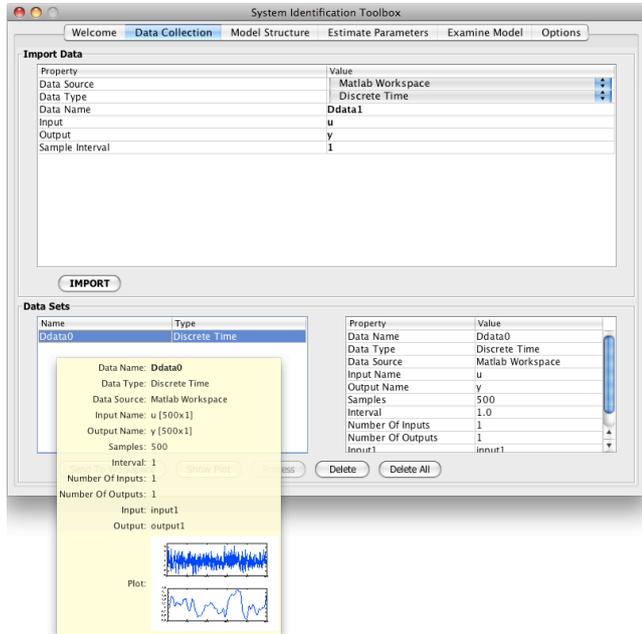


Fig. 1. GUI Data Specification Pane

from MATLAB workspace vectors u and y , and (lower right corner of pane) being given the default name `Data0`. This field is editable to an arbitrary name if desired. The lower left of the pane shows a list of defined data sets (only one has been defined), and the yellow box shown there is a “tooltip” that appears when the mouse hovers over the data. Its purpose is to allow the user to, if necessary, quickly remind themselves as to which data set corresponds to a particular name.

Once one or more data sets have been defined, the user moves to the second model specification pane shown in Figure 2. The specification itself is performed by making selections in the top left of the pane.

As the model structure is altered here, a graphic representation of the model structure is adjusted and presented in the lower half of the pane. Figure 2 illustrates that a Box–Jenkins model has been defined with saturation non-linearity on the input, and piecewise linear map on the output.

The top right of the pane provides a list of defined model structures. Again, as illustrates in Figure 2, mouse hovering over a model structure name raises a tooltip that provides the user with a quick reminder of the details of a model structure corresponding to that name.

The model structure name is chosen automatically in a manner to reflect the details of that structure. For example, as illustrated in Figure 2, the name `BJ_nlinout_55220` has been chosen to indicate that it pertains to a Box–Jenkins structure with non-linearities on both input and output, and with model orders for A, B, C, D transfer

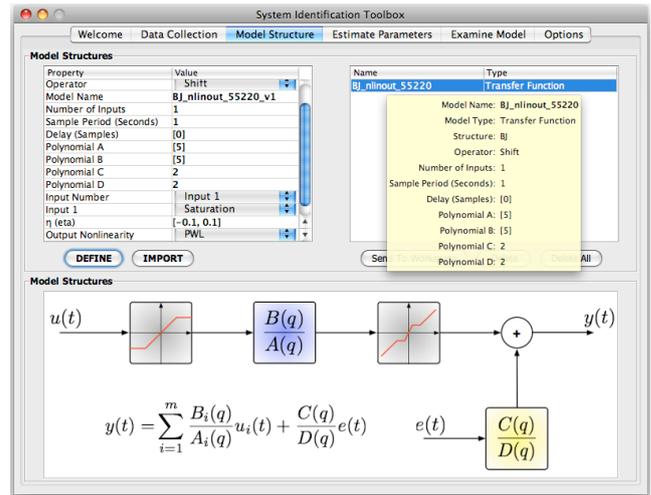


Fig. 2. GUI Model Structure Specification Pane

function of (respectively) 5,5,2 and 2, and with zero delay on the input.

However, the field where this appears in the top left pane is editable to an arbitrary name if desired. The fact that it is editable is indicated by its bold font. A general principle with the GUI is that all text appearing in bold is editable, with all other text non-editable.

Once one or more model structures have been defined, the user progresses to the estimation pane illustrated in Figure 3.

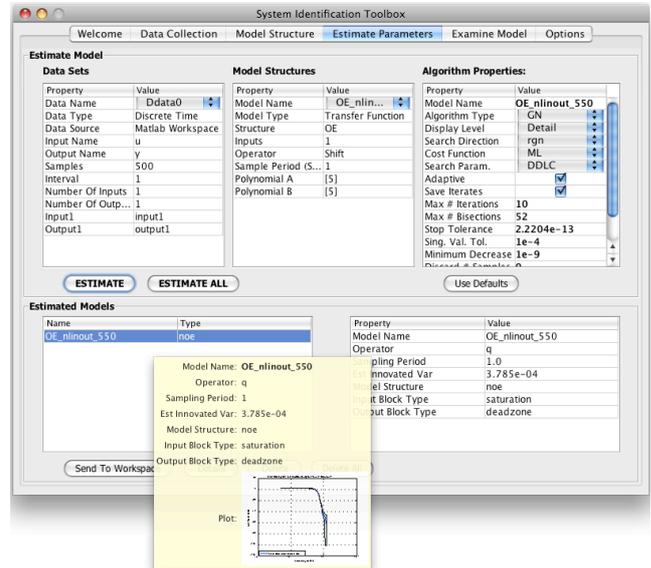


Fig. 3. GUI Model Estimation Pane

All model structures specified in the previous pane are available to be chosen from a drop down list at the top of the middle column in the upper half of the pane.

Hitting the “Estimate” button at the middle left of the pane results in this model structure being estimated. The data set used for estimation is specified by the drop down list at the top of the left column in the upper half of the pane. The algorithm used and any parameters associated

with it are specified by drop down lists and editable fields in the right column in the top half of the pane.

At the completion of an estimation step, the resulting estimated model populates a list in the lower left of the pane. Figure 3 illustrates a tooltip generated by mouse hovering over one such estimated model.

A final important feature of this pane is that if multiple model structures were defined on the previous pane, then the “Estimate All” button in the middle of the pane may be used to estimate all of these structures using the data set specified in the left top column.

With models estimated, the user now progresses to the final pane supporting the evaluation, validation and comparison of them. This is illustrated in Figure 4. Each model

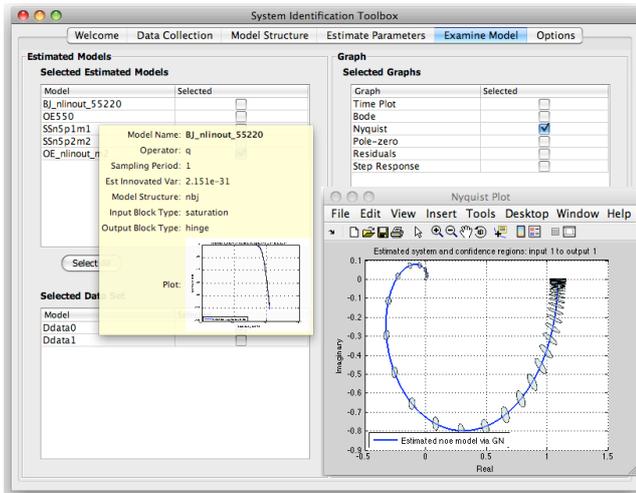


Fig. 4. GUI Model Validation and Comparison Pane

estimated on the previous pane appears in a list in the upper left quarter of the pane. In Figure 4, a tooltip is being illustrated by mouse hover over a model in this list.

If the user wishes to validate one or more of these models, then they select via check box the models to be validate in the upper left pane quarter. They then select the data to be used for validation, again via check box in the lower left pane quarter. Going to the upper right pane quarter and using a check box to select “Time Plot” will then show the results of the `validate` function described earlier.

Other plots may also be generated for any model checked in the upper left pane, by clicking the appropriate check box in the upper right pane. Figure 4 is illustrating the simple case of one Nyquist plot (with 95% confidence regions) for one estimated mode being selected and displayed.

Further features of the GUI are the ability to export estimated models to the MATLAB workspace, and the availability of a help system to guide the user through using the four panes of the GUI.

REFERENCES

John W. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems*, 25(9):772–781, September 1978.

Wouter Favoreel, Bart De Moor, and Peter Van Overschee. Subspace identification of bilinear systems subject to white inputs. *IEEE Trans. Automat. Control*, 44(6):1157–1165, 1999. ISSN 0018-9286.

Stuart Gibson and Brett Ninness. Robust maximum-likelihood estimation of multivariable dynamic systems. *Automatica*, 41(10):1667–1682, 2005.

Stuart Gibson, Adrian Wills, and Brett Ninness. Maximum-likelihood parameter estimation of bilinear systems. *IEEE Transactions on Automatic Control*, 50(10):1581–1596, 2005.

W. Larimore. Canonical variate analysis in identification, filtering and adaptive control. In *Proceedings of the 29th IEEE Conference on Decision and Control, Hawaii*, pages 596–604, 1990.

L. Ljung and A. G. Wills. Issues in sampling and estimating continuous-time models with stochastic disturbances. In *In proceedings of the 17th IFAC World Congress on Automatic Control*, Seoul, Korea, July 6–11 2008.

Lennart Ljung. *System Identification: Theory for the User, (2nd edition)*. Prentice-Hall, Inc., New Jersey, 1999.

Mathworks. *MATLAB Users Guide, Version 7*. The Mathworks, 2004.

Tomas McKelvey, Hüseyin Akçay, and Lennart Ljung. Subspace-based identification of infinite-dimensional multivariable systems from frequency-response data. *Automatica J. IFAC*, 32(6):885–902, 1996a. ISSN 0005-1098.

Tomas McKelvey, Hüseyin Akçay, and Lennart Ljung. Subspace-based multivariable system identification from frequency response data. *IEEE Transactions on Automatic Control*, 41:960–979, 1996b.

R.H. Middleton and G.C. Goodwin. *Digital Estimation and Control: A Unified Approach*. Prentice-Hall, Inc., New Jersey, 1990.

Brett Ninness and Adrian Wills. An identification toolbox for profiling novel techniques. In *14th IFAC Symposium on System Identification*, pages 301–307, mar 2006.

R. Pintelon and J. Schoukens. *System Identification: A Frequency Domain Approach*. IEEE Press, 2001.

Peter van Overschee and Bart de Moor. N4SID:Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.

Adrian Wills and Brett Ninness. On gradient-based search for multivariable system estimates. *IEEE Trans. Automat. Control*, 53(1):298–306, 2008. ISSN 0018-9286.

Adrian Wills and Brett Ninness. On gradient-based search for multivariable system estimates. In *Accepted for publication in the Proceedings of the IFAC World Congress, Prague*, July 2005.

Adrian Wills, Brett Ninness, and Stuart Gibson. Maximum likelihood estimation of state space models from frequency domain data. *To appear, IEEE Transactions on Automatic Control*, 2008.