AN IDENTIFICATION TOOLBOX FOR PROFILING NOVEL TECHNIQUES

Brett Ninness [*,1] **Adrian Wills** [*]

[*] *School of Elec. Eng. & Comp. Sci. University of Newcastle, Australia. email:*brett@ee.newcastle.edu.au

Abstract: This paper describes a Matlab (or Octave) based software package 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 that streamlines comparisons. Key features of the toolbox include simplicity of use (particularly via automated entry of unspecified values), the support of a wide range of scalar and multivariable model structures which include certain nonlinear classes such as bilinear and Hammerstein–Wiener, the ability to handle both time and frequency domain data, the hand optimisation of certain key routines compiled against ATLAS libraries for optimum speed, the use of non-standard optimisation methods based on adaptive subspace gradient search and the Expectation-Maximisation method, and the fact that the toolbox is freely available from `http://sigpromu.org` for non-commercial use.

Keywords: Parameter Estimation, System Identification, Software.

1. INTRODUCTION

This paper describes a MATLAB/Octave-based toolbox for the identification of dynamic systems on the basis of experimental observations.

While it began as a self-education exercise for the authors, it has developed into a useful platform for profiling new methods relative to existing ones, and for robust transfer and evaluation of these new methods to research colleagues.

In the hope that a wider community might find these new techniques and the underlying platform both interesting and useful, as first announced in (Ninness *et al.*, 2005) and expanded upon here the toolbox is freely available for non-commercial use via download at `http://sigpromu.org`.

Key aspects of the toolbox are

- It is MATLAB (Mathworks, 2004) based, with an Octave (`www.octave.org`) port also available;
- Simplicity of use is emphasised, with all estimation methods and model structures encapsulated in one command:
  `g=est(z,m,o)`
  with automatic entry (to default values) of unspecified entries;
- A wide range of model structures are accommodated, from standard SISO/MISO polynomial ones, through fully-parametrized MIMO state-space cases, both in linear and bilinear forms, and together with memoryless non-linearities;
- Both time and frequency domain data are handled, and in the time-domain case non-regular sample instants may be accommodated via the estimation of continuous time (presently only state-space) model structures;
- (MATLAB port only) Key routines are hand-coded in C and compiled to `mex` files linked against platform optimised ATLAS (Netlib, 2004) libraries for maximum speed;
- Robust (Square root) Kalman Filtering and Smoothing routines for linear state estimation are provided, together with particle filtering routines for non-linear state estimation.

[1] This work was supported by the Australian Research Council.

Essential innovative features include

- The above-mentioned ability to accommodate non-linear model structures of bilinear and Hammerstein–Wiener type;
- The ability to employ, as an option, the Expectation-Maximisation method for computing Maximum-Likelihood estimates;
- The inclusion, as standard, of a robust version of Gauss–Newton gradient-based search that has been found particularly effective for MIMO system estimation (Wills and Ninness, 2004);
- The facility to employ δ-operator (Middleton and Goodwin, 1990) parametrizations for polynomial type model structures.

In order to profile these points, the paper will proceed by providing further detail on the model structures supported, the estimation methods and optimisation techniques used, and then conclude by illustrating the use of the toolbox on some simple examples.

2. SUPPORTED MODEL STRUCTURES

To begin with, in the case of time domain data, the toolbox supports standard polynomial/transfer function models of the following form

$$z_t = \sum_{i=1}^{m} G_i(\rho,\theta) X_i(u_t^i,\theta) \tag{1}$$

$$y_t = Z(z_t,\theta) + H(\rho,\theta)e_t \tag{2}$$

where $u_t^1, \cdots, u_t^m$ are each scalar measured inputs, y_t is a scalar measured output, e_t is a scalar zero mean i.i.d. process of variance $\mathbf{E}\left\{e_t^2\right\} = \sigma^2 < \infty$, and $\theta \in \mathbf{R}^n$ is a vector specifying the model parameters.

That is, in the transfer function case, multiple input, single output (MISO) model structures are supported. To be completely clear, the elements $G_i(\rho,\theta)$ and $H(\rho,\theta)$ are rational according to

$$G_i(\rho,\theta) = q^{-k_i}\frac{B_i(\rho,\theta)}{A_i(\rho,\theta)}, \quad H(\rho,\theta) = \frac{C(\rho,\theta)}{D(\rho,\theta)} \tag{3}$$

$$A_i(\rho,\theta) = 1 + a_1\rho + a_2\rho^2 + \cdots + a_{m_a}\rho^{m_a^i}, \tag{4}$$

$$B_i(\rho,\theta) = b_0 + b_1\rho + b_2\rho^2 + \cdots + b_{m_b}\rho^{m_b^i}, \tag{5}$$

$$D(\rho,\theta) = 1 + d_1\rho + d_2\rho^2 + \cdots + d_{m_d}\rho^{m_d}, \tag{6}$$

$$C(\rho,\theta) = 1 + c_1\rho + c_2\rho^2 + \cdots + c_{m_c}\rho^{m_c}, \tag{7}$$

and k_i is a specified (i.e. not estimated) delay acting on the i'th input.

Since any of the orders m_a^i, m_b^i, m_c, m_d may be set to zero, then the toolbox can implement any of the common FIR, ARX, ARMAX, ARMA, Output-Error and Box–Jenkins model structures(Ljung, 1999).

The symbol ρ above is a time domain operator which may be set to

$$\rho = q^{-1} \tag{8}$$

the standard (backward) shift operator, or

$$\rho = \delta \triangleq \frac{q-1}{\Delta} \tag{9}$$

the Euler difference, or "Delta" operator(Middleton and Goodwin, 1990) with Δ being the underlying sampling period.

In (1), each of the $X_i(u_t^i,\theta)$ are (possible) memoryless nonlinearities, as is $Z(z_t,\theta)$ in (2). That is, the toolbox supports the estimation of Hammerstein–Wiener type non-linear systems, with the following possibilities for the types of memoryless functions X_i,Z: Saturation, Deadzone, Polynomial and Hinging-Hyperplane (Piecewise Linear).

State space model structures are also supported, beginning in the linear case with the form

$$\begin{bmatrix} x_{t+1} \\ y_t \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} w_t \\ e_t \end{bmatrix}, \begin{bmatrix} K \\ I \end{bmatrix}\epsilon_t \tag{10}$$

Here, both u_t and y_t may be vectors so that the multiple input, multiple output (MIMO) scenario can be accommodated. Note that in (10) two forms for the estimation and state noise are indicated as being possible. One is the well known situation of separate state noise w_t and measurement noise e_t which are zero mean i.i.d. processes with

$$\mathrm{Cov}\left\{\begin{bmatrix} w_t \\ e_t \end{bmatrix}\right\} = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix}. \tag{11}$$

The other, is the innovations form, with ϵ_t being an i.i.d. zero mean process with

$$\mathrm{Cov}\left\{\epsilon_t\right\} = \overline{R}. \tag{12}$$

In the Gaussian case, K and $\overline{R}$ in (10),(12) follow from the solution of the associate discrete algebraic Riccati equation specified by A, C, Q, S, R in (10),(11).

These two cases are parametrized (respectively) in the toolbox according to

$$\theta^T \triangleq \Big[\mathrm{vec}\left\{A\right\}^T, \cdots, \mathrm{vec}\left\{D\right\}^T, \mathrm{vec}\left\{Q\right\}^T, \mathrm{vec}\left\{R\right\}^T, \mathrm{vec}\left\{S\right\}^T\Big], \tag{13}$$

and

$$\theta^T \triangleq \Big[\mathrm{vec}\left\{A\right\}^T, \cdots, \mathrm{vec}\left\{K\right\}^T, \mathrm{vec}\left\{\overline{R}\right\}^T\Big] \tag{14}$$

where the $\mathrm{vec}\left\{\cdot\right\}$ operator is one that forms a vector from a matrix by stacking the columns of the matrix on top of one another. That is, the parametrizations in both cases are full (non minimal) ones in which every element in every matrix is a parameter to be estimated.

The toolbox also supports an extended version of this MIMO state-space model that encompasses nonlinear systems of bilinear form according to

$$\begin{bmatrix} x_{t+1} \\ y_t \end{bmatrix} = \begin{bmatrix} A & F & B \\ C & G & D \end{bmatrix} \begin{bmatrix} x_t \\ u_t \otimes x_t \\ u_t \end{bmatrix} + \begin{bmatrix} w_t \\ v_t \end{bmatrix}, \begin{bmatrix} K \\ I \end{bmatrix} \epsilon_t$$

where $\otimes$ represents the Kronecker tensor product (Brewer, 1978), the noise modelling is as presented in (11), (12) and the parametrizations are again full ones according to either

$$\theta^T \triangleq \Big[\mathrm{vec}\,\{A\}^T, \cdots, \mathrm{vec}\,\{G\}^T, \mathrm{vec}\,\{Q\}^T, \mathrm{vec}\,\{R\}^T, \mathrm{vec}\,\{S\}^T\Big],$$

or

$$\theta^T \triangleq \Big[\mathrm{vec}\,\{A\}^T, \cdots, \mathrm{vec}\,\{G\}^T, \mathrm{vec}\,\{K\}^T, \mathrm{vec}\,\{\overline{R}\}^T\Big]$$

This covers all available model structures in the case of time domain data, although before progressing it should be noted that while the notation has suggested that all samples (for example u_t, y_t are obtained at regular sampling points, in the case of the state space model structure (10), the toolbox can accommodate the samples occurring at arbitrarily distributed instants. It remains for future work to accommodate this embellishment in other (for example, polynomial) model structures.

In the case of frequency domain data, two model structure types are supported. The first is the (SISO) transfer function form

$$Y(\omega_k) = G(\gamma_k, \theta) + H(\gamma_k, \theta)E(\omega_k) \tag{15}$$

with G and H being as described in (3)-(7).

The other is the state-space form

$$Y(\omega_k) = G(\gamma_k, \theta) + E(\omega_k), \tag{16}$$

$$G(\gamma, \theta) = C(\gamma I - A)^{-1}B + D \tag{17}$$

with matrices according to (10) and

$$\theta^T \triangleq \Big[\mathrm{vec}\,\{A\}^T, \cdots, , \mathrm{vec}\,\{D\}^T\Big]. \tag{18}$$

In both cases, the operator γ_k may be set to either

$$\gamma_k = e^{j\omega_k \Delta} \tag{19}$$

if a shift operator discrete time model is required, or to

$$\gamma_k = j\omega_k \tag{20}$$

if a continuous time (Laplace s operator) model is of interest. In both cases, ω_k is the radian per second value at which the frequency response $Y(\omega_k)$ has been measured.

Finally, aside from the parametrized model structures presented here, the toolbox also supports non-parametrized models estimated via either Empirical Transfer Function Estimates (ETFE) or Blackman–Tukey procedures (Ljung, 1999).

3. ESTIMATION METHODS

In the toolbox, the primary means for parameter in the case of time domain experimental data involves solution of the following optimisation problem

$$\widehat{\theta}_N \triangleq \arg\min_{\theta \in \mathbf{R}^n} V_N(\theta). \tag{21}$$

This is of a least-squares sort in that

$$V_N(\theta) \triangleq E(\theta)^T E(\theta) \tag{22}$$

where

$$E^T(\theta) \triangleq [\varepsilon_1^T(\theta), \cdots, \varepsilon_N^T(\theta)] \tag{23}$$

with

$$\varepsilon_t(\theta) \triangleq y_t - \widehat{y}_{t|t-1}(\theta). \tag{24}$$

Here $y_{t|t-1}(\theta)$ is the (mean square) optimal one step ahead prediction of y_t conditional on experimental observations up to and including time $t-1$ and based on a model parametrized by θ.

In the case of state-space modelling (including the bilinear case) it is computed by (a square root implementation of) a Kalman filter, while in the polynomial case it is computed via the steady state Kalman filter in transfer function form; viz.

$$\widehat{y}_{t|t-1}(\theta) = H^{-1}(q, \theta) \sum_{i=1}^{m} G_i(q, \theta) u_t^i + \left[1 - H^{-1}(q, \theta)\right] y_t \tag{25}$$

In the situation of Gaussian distributed noise, the toolbox also supports estimation according to the Maximum-Likelihood criterion

$$\widehat{\theta}_N \triangleq \arg\min_{\theta \in \mathbf{R}^n} L(\theta), \tag{26}$$

$$L(\theta) \triangleq -\log p_\theta(y_1, \cdots, y_N) \tag{27}$$

where $p_\theta(y_1, \cdots, y_N)$ is the joint probability density functions of the observations $y_1, \cdots, y_N$ which depend on a model structure defined by the parameters θ.

Finally, in the case of estimation from frequency domain data, the toolbox employs a minimum norm method in which a solution $\widehat{\theta}_N$ according to (21) is sought where ($\star$ denotes conjugate transpose)

$$V_N(\theta) \triangleq E(\theta)^\star E(\theta) \tag{28}$$

with $E(\theta)$ being as in (23) but with

$$\varepsilon_k(\theta) \triangleq \left[Y(\omega_k) - G(\gamma_k, \theta)\right] H(\gamma_k, \theta). \tag{29}$$

4. ALGORITHMS

The toolbox profiled in this paper offers a number of algorithms to find a parameter estimate $\widehat{\theta}_N$ as solution of either (21) or (26).

Firstly, with $\cdot'$ denoting differentiation with respect to θ so that $E'(\theta)$ is the Jacobian Matrix associated with the estimation error vector (23), the toolbox can implement a standard Gauss–Newton based search for $\widehat{\theta}_N$ satisfying (21) by finding iterations starting from an initial guess θ_0 which is then refined according to

$$\theta_{k+1} = \theta_k + \mu\, p \tag{30}$$

where the search direction p is any solution to

$$\left[E'(\theta_k)^T E'(\theta_k)\right] p = -E'(\theta_k)^T E(\theta_k). \tag{31}$$

Here, μ is step length, that while illustrated in (30) as fixed for simplicity of notation, in fact is not. At each iteration k, an initialisation of $\mu = 1$ is set. Then

$$V_N(\theta_{k+1}) < V_N(\theta_k) \tag{32}$$

is tested. If this test fails, $\mu := \mu/2$ is set and the update (30) is recomputed followed by the test (32) until this test succeeds, or a maximum number of bisections is reached and the search terminates.

Note that in the case of state-space modelling with full parametrizations, the ensuing over-parametrization implies rank deficiency of the Jacobian $E'(\theta)$ so that the solution to (31) is not unique.

In this case, the toolbox first employs a Data Driven Local Co-Ordinate (McKelvey *et al.*, 2004) re-parametrization in which at iteration k, a matrix P_k with columns orthogonal to the null space of $E'(\theta)$ is found.

This implies a search direction λ in a reduced dimension parameter space defined as a solution to

$$\left[P_k^T E'(\theta_k)^T E'(\theta_k) P_k\right] \lambda = -P_k^T E'(\theta_k)^T E(\theta_k) \tag{33}$$

which then defines a modified Gauss–Newton update strategy of

$$\theta_{k+1} = \theta_k + \mu\, P_k \lambda. \tag{34}$$

However, in cases of poor input excitation, once again, the solution to (33) may not be unique. Indeed, due to this reason or (for example) due to over-modelling, this problem may also occur in the case of polynomial model structures, even though (for the given model order) the parametrizations employed there are minimal.

To deal with these situations, the toolbox employs the update strategy (30) with search direction p given by

$$p = -V_1 S_1^{-1} U_1^T E(\theta_k) \tag{35}$$

where the Jacobian $E'(\theta)$ has singular value decomposition

$$E'(\theta) = USV^T = [U_1, U_2] \begin{bmatrix} S_1 & \varnothing \\ \varnothing & \varnothing \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}. \tag{36}$$

The dimension of S_1 is chosen adaptively, which is coupled to the choice of step length μ in a manner which cannot be described here due to space restrictions, but is fully profiled in(Wills and Ninness, 2004).

Moving on, in the case of the Maximum-Likelihood estimation method (26), the toolbox employs the Expectation-Maximisation (EM) algorithm which only handles states spaced structures of the form (10) and involves iterations

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{Q}(\theta, \theta_k) \tag{37}$$

where

$$\mathcal{Q}(\theta, \theta_k) \triangleq \mathbf{E}_{\theta_k}\{\log p_\theta(X_N, Y_N) \mid Y_N\} \tag{38}$$

$$X_N \triangleq x_1, \cdots, x_N, \quad Y_N \triangleq y_1, \cdots, y_N. \tag{39}$$

In the case of Gaussian noise (modulo initial condition effects) the estimation methods (21) and (26) co-incide so that this EM algorithm offers an alternative to gradient based search for the computation of (21) which, as examined in detail in (Gibson and Ninness, 2005),(Gibson *et al.*, 2005) offers a particularly robust and reliable alternative, hence the inclusion of the method in the toolbox.

Finally, although no underlying optimisation problem is explicit, the toolbox also implements so-called state-space subspace-based estimation methods. These include the N4SID, MOESP and Canonical Variate types (Ljung, 1999).

5. SOFTWARE DESCRIPTION

For the purposes of finding a system estimate from available data, a user of the toolbox need know only one command:

```
g=est(z,m,opt);
```

All arguments, and the returned output `g` are structure variables, which (typically) contain individual elements of mixed type.

To be more specific, the structure `z` defines the observed data, and must have element `z.y` and `z.u` which (respectively) are matrices of output and input data, with each column representing a different input or output, and each row a different time sample.

The structure `m` defines the model structure to be used. At a minimum, it must contain the entry `m.A` which, if an integer, defines a model order, and if a matrix/vector defines an estimate. The structure `opt`, defines optional specifications pertaining to the algorithm being used. If there are none the

user wishes to dictate, then `g=est(z,m)` is a legal command.

To make this concrete, consider the data set shown in figure 1 with measure input stored in vector `u` and output in vector `y`. Then the following

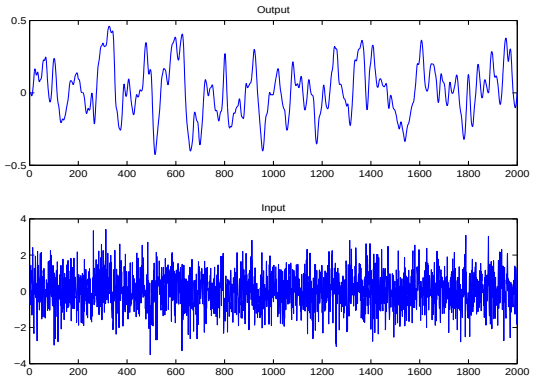


Fig. 1. *Observed Input/Output Data.*

sequence of commands will estimate a fifth order model from the data.

```
>> z.y=y; z.u=u; m.A=5; g=est(z,m);
```

As mentioned, `g` is a structure, and in the above example it will contain 26 elements. In order to extract a summary of the properties of the estimate, the `details` command provides

```
>> details(g)
------------------------------------
Details for Estimated Model Structure
------------------------------------
Operator used in model          = q
Sampling Period                 = 1.000000 seconds
Estimated Innovations Variance = 5.059506e-03
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    q^-5
B =  0.0011 -0.0019  0.0031 -0.0005 -0.0029  0.0081
SD=  0.0016  0.0029  0.0029  0.0033  0.0033  0.0024

     1       q^-1    q^-2    q^-3    q^-4    q^-5
A =  1.0000 -1.4324 -0.5590  1.4602 -0.3570 -0.1047
SD=  0       0.2955  0.4923  0.1017  0.4801  0.2049

delay = 0 samples

Poles at 0.8372*exp(+-j0.0797), -0.9698, -0.1704, 0.9035.
```

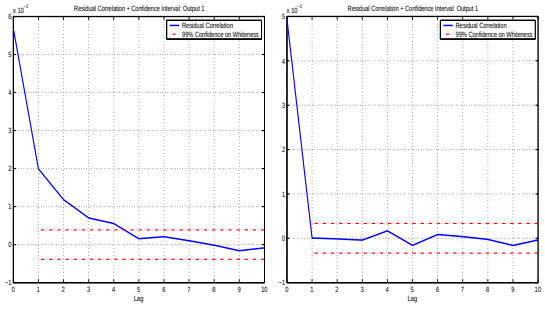
This illustrates a main point. A philosophy underlying the toolbox is that, in order to maximise utility for the inexperienced, defaults are used as opposed to issuing error messages. In particular, the above indicates that since only an order was specified, the operator type (q), sampling period, and model order type (Output Error) have all been set as defaults.

In order to assess the quality of this model, a standard model validation test may be performed

```
validate(z,g);
```

This provides the sample-correlation of the error residuals as shown in figure 2(a). Clearly, there appears to be some undermodelling. In order to illustrate the specification of a Box–Jenkins structure, the following commands specify that a first order noise model should be added to the model structure (`m.D=1`), that the progress of the Gauss–Newton iterations should be displayed (`opt.dsp=1`) and that this new model should also be validated

```
>> m.D=1; opt.dsp=1; g1=est(z,m,opt);
Finding initial dynamics model via Steiglitz-McBride...
Finding initial noise Model by Hannan-Rissanen...
bisec# = 2,cost = 4.9459e-03,gn norm =4.9212e+00:G-N Direction
bisec# = 6,cost = 4.9448e-03,gn norm =2.2722e+01:G-N Direction
bisec# = 3,cost = 4.9392e-03,gn norm =1.1786e+00:G-N Direction
bisec# = 3,cost = 4.9374e-03,gn norm =7.0141e-02:G-N Direction
bisec# = 1,cost = 4.9361e-03,gn norm =5.4293e-02:G-N Direction
bisec# = 2,cost = 4.9358e-03,gn norm =9.6350e-01:G-N Direction
-----------------------------------------------------
Termination due to relative cost decrease < OPT.mdec
-----------------------------------------------------
>> validate(z,g1);
```



(a) OE Model (b) BJ Model

Fig. 2. *Model Validation*

The ensuing validation results are shown in figure 2(b). Of course, in practise, data other than that used for estimation should be employed for this validation purpose, but we ignore this in order to streamline the presentation.

To illustrate a more complex example, consider the MISO data shown in figure 3. This comes

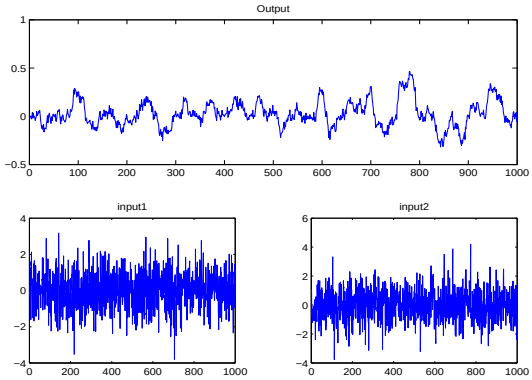


Fig. 3. *Observed MISO Input/Output Data from Hammerstein System.*

from a system with saturation and deadzone non-linearities on inputs 1 and 2 respectively, that then pass through linear systems of order 4 and 3 respectively with the output then passing through a further deadzone non-linearity.

In order to estimate a model structure of this form we proceed as follows

```
>> z.y=y; z.u=u;
>> m.A=[4;3]; m.B=[3;2]; m.delay=[1;1];
>> m.in(1).type='hinge';
>> m.in(2).type='deadzone';
>> m.out.type='deadzone';
```

```
>> m.out.upper=0.1; m.out.lower=-0.1;
>> g=est(z,m);
```

Note that now two model orders are specified, which are different, for the two linear components associated with the two inputs. It is important that the different orders are specified as a column vector. If they had been a row vector, the toolbox would interpret them not as orders, but an initial estimate of a polynomial denominator and/or numerator.

In the above, a time delay of one sample is specified for both of the inputs. Furthermore, it is specified that a piecewise linear structure should be used to model the non-linearity on the first input, while for the second, prior knowledge that it is of deadzone type is employed. Finally, a deadzone non-linearity is specified for the output, with initial estimates for the deadzone region as $[-0.1, 0.1]$.

As before, the results of this estimation experiment can be summarised via use of the `details` command (to save space, output related to the linear component of the model structure has been deleted).

```
>> details(g)
------------------------------------
Details for Estimated Model Structure
------------------------------------
Operator used in model         = q
Sampling Period                = 1.000000 seconds
Estimated Innovations Variance = 1.069568e-04
Model Structure Used           = Output Error
Estimation algorithm           = Gauss-Newton search
Input #1 block type            = hinge
Input #2 block type            = deadzone
Output block type              = deadzone
------------------------------------
------------------------------------------------------------
Input Non-linearity Parameters and standard deviations:
------------------------------------------------------------
Input block #1 of type hinge has estimates and standard dev:

eta =

   -6.1438    0.1417    7.5736  -12.8909    6.3845   12.8978
    0.1249    0.0875    0.1424    0.1980    0.1265    0.2156

Input block #2 of type deadzone has estimates and standard dev:

upper limit = 0.6064, sd = 0.0058
lower limit = -0.4900, sd = 0.0051

------------------------------------------------------------
Output Non-linearity Parameters and standard deviations:
------------------------------------------------------------
Output block of type deadzone has estimates and standard dev:

upper limit = 0.0618, sd = 0.0014
lower limit = -0.0398, sd = 0.0015
```

By way of information, the true noise variance was $\sigma^2 = 10^{-4}$, the true deadzone region on input 2 was (lower, then upper limit) $[-0.5, 0.6]$. Furthermore, the above hinge parametrization corresponds to a deadzone with limits $[-0.49, 0.59]$ while the underlying true one was $[-0.5, 0.6]$. There was no output non-linearity in the true system, which has essentially also been estimated in the model, even though it was allowed for.

There are further features of the toolbox that have not been profiled by these brief examples but are still worth mentioning.

- An estimated model structure `g` may be used to specify the model structure `m` in cascaded estimation experiments;
- It is not necessary to use the `details` command to assess the features of an estimated model `g`, since the latter is simply a structure, whose elements will be echoed to the screen by typing it at the `>>` prompt;
- Frequency domain plots, complete with error bounds may be simply generated via the `showbode(g)` and `shownyq(g)` commands;
- A model structure may be forced by setting `m.type` to `fir`,`arx`,`armax`,`oe`, `bj` or `ss`;
- A delta operator parametrization may be specified by setting `m.op='d'`;
- The EM algorithm may be specified by setting `opt.alg='em'`. This and `m.type='ss'` will be used as a default if `z.y` contains multiple columns (outputs);
- Finally, online help for all functions is available. In particular `help est` gives a catalogue of all options available.

6. CONCLUSION

The software profiled here is intended as an open source platform to support further research developments in the science of system identification. It will co-exist without conflict in a MATLAB environment where the propriety system identification toolbox available for that product (Ljung, 2004) is installed. Nevertheless, the toolbox profiled here is intended mainly for researchers and experienced practitioners. It does not pretend to offer the same level of on-line help, error-checking and user guidance (for example) that are necessary in a package suitable for widespread industrial use.

REFERENCES

Brewer, J. W. (1978). 'Kronecker products and matrix calculus in system theory'. *IEEE Trans. Circ. Sys.* **25**(9), 772–781.

Gibson, S., Adrian Wills and Brett Ninness (2005). 'Maximum-likelihood parameter estimation of bilinear systems'. *IEEE Trans. Auto. Cont.* **50**(10), 1581–1596.

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

Ljung, L. (1999). *System Identification: Theory for the User*. Prentice-Hall, New Jersey.

Ljung, L. (2004). *MATLAB System Ident. T'box Users Guide, Version 6*. The Mathworks.

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

McKelvey, T., A. Helmersson and T. Ribarits (2004). 'Data driven local co-ord's for MIMO linear systems & application to system identification'. *Automatica* **40**(9), 1629–1635.

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

Netlib (2004). 'The Netlib Repository'. `www.netlib.org`.

Ninness, B., Adrian Wills and Stuart Gibson (2005). The Uni. Newc. Id. T'box (UNIT). In 'Proc. IFAC World Congress, Prague'.

Wills, A. and Brett Ninness (2004). On gradient-based search for multivariable system estimates. In 'Proc. IFAC World Congress, Prague'.