Control Engineering Practice 21 (2013) 631–644

Contents lists available at SciVerse ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

UNIT: A freely available system identification toolbox ☆

Brett Ninness *, Adrian Wills, Adam Mills

School of Electrical Engineering and Computer Science, The University of Newcastle, Australia

ARTICLE INFO

Article history:
Received 1 March 2012
Accepted 27 October 2012
Available online 26 February 2013

Keywords:
Parameter estimation
System identification
Maximum Likelihood
Nonlinear system identification
Software

ABSTRACT

This paper presents a Matlab-based software package for the estimation of dynamic systems. It has been developed primarily as a platform to support the objective evaluation of novel approaches relative to existing methods within a common software framework. This is designed to streamline comparisons. The work here provides an explanation of the toolbox's design and use together with an overview of the underlying supported models and algorithms. An application study involving a lightly damped resonant system and a simulation example involving two non-linear systems are also presented in order to illustrate the use and capabilities of the toolbox.

1. Introduction

This paper details the "*University of Newcastle Identification Toolbox*" (UNIT) that has been developed to run under the Matlab (Mathworks, 2012) environment. The package is freely available (www.sigpromu.org/staff/idtoolbox) and offers a comprehensive 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 toolbox 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 & Ninness, 2005; Gibson, Wills, & Ninness, 2005; Wills, Ninness, & Gibson, 2009).
- Use of an adaptive Jacobian rank algorithm (Wills & Ninness, 2008) for gradient based search of least squares estimates.
- Accommodation of data collected at non-regularly spaced sampling instants via new continuous time modelling methods (Ljung & Wills, 2010).
- Support for δ-operator based discrete time models (Middleton & Goodwin, 1990).
- Use of Markov-Chain Monte–Carlo (MCMC) methods for computing Bayesian estimates and associated posterior densities (Ninness & Henriksen, 2010).
- The employment of particle filtering techniques for non-linear system estimation (Schön, Wills, & Ninness, 2011).

With regard to these non-standard aspects, the motivation for developing the software suite described here is to provide a platform for efficiently and objectively evaluating the performance of new methods relative to existing ones, and also as a means for effectively disseminating new methods to other researchers. In order to present the toolbox, the paper will provide details of the model structures supported, the estimation methods implemented, and the algorithms employed.

There exists a range of other actively maintained toolboxes for dynamic system identification. These include commercially licensed ones such as the comprehensive Mathworks System Identification Toolbox (Ljung, 2012), the Gamax Frequency Domain System Identification Toolbox (Kollár, Pintelon, & Schoukens, 2006) and the Adaptx subspace-based software (Larimore, 2000) for estimation of state-space models. Non-commercial and freely available packages include the CONTSID toolbox (Garnier, Gilson, & Laurain, 2012) which concentrates on estimation of continuous time models, the CAPTAIN toolbox (Young & Taylor, 2012) which provided a platform supporting the "refined instrumental variable" (RIV) algorithm (Young, 2011) and the ITSIE toolbox (Guzmán, Rivera, Dormido, & Berenguel, 2012) which has an emphasis on education and training in system identification principles.

What distinguishes the toolbox reported here from these other packages is that it is both freely available and supports a wide range of model structures, estimation methods and algorithms. It is intended as an open platform facilitating objective comparison of new methods (such as EM and MCMC-based methods) relative to existing approaches.

2. Fundamentals

The toolbox revolves around the use of the fundamental estimation function `est` which is deployed according to a Matlab

* Corresponding author.
E-mail addresses: brett.ninness@newcastle.edu.au (B. Ninness), adrian.wills@newcastle.edu.au (A. Wills), adam.mills@newcastle.edu.au (A. Mills).

☆This work was supported by the Australian Research Council.

0967-0661/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.conengprac.2012.10.007

function call

$$\gg \texttt{g} = \texttt{est}(\texttt{z},\texttt{m},\texttt{opt}); \tag{1}$$

Each of the elements `g`, `z`, `m` and `opt` are Matlab structures made up of components. The components of the structure `g` specify an estimated model. This is computed by `est` on the basis of observed measurements that are the components of `z`. The type of model that `est` computes based on the observations `z` is specified by the components of the model structure `m`. Optional aspects of how `g` is computed, such as the exact estimation algorithm used, and parameters that tune this algorithm are contained in the structure `opt`.

The sections immediately following detail the capabilities of the toolbox that can be accessed via these structures.

3. Data formats

The toolbox is able to perform identification from either time domain or frequency domain data. In both situations, the toolbox accepts this data via a structure `z`.

In the time domain case the element `z.y` is required, and is interpreted as the observed system output response. In many cases, this is due to a known excitation, which may be specified as `z.u`. For example

$$\gg \texttt{z.y} = \texttt{Y}; \ \texttt{z.u} = \texttt{U}; \tag{2}$$

where `Y` is $N \times p$ and `U` is $N \times \ell$ is interpreted as the specification of N regularly spaced samples of a multivariable system with p outputs and ℓ inputs. By default, it is assumed these samples are collected at regularly spaced intervals of $T=1$ s.

If the samples are not obtained at regularly spaced intervals, this is specified according to

$$\gg \texttt{z.t} = \texttt{T}; \tag{3}$$

where $\texttt{T} = [t_1, t_2, \ldots, t_N]$ is a length N vector of "time stamps" at which the corresponding rows of `z.y` and `z.u` were sampled.

In the frequency domain case, `z.y` is expected to contain the measured frequency response of the system. For example

$$\gg \texttt{z.y} = \texttt{F}; \ \texttt{z.w} = \texttt{W}; \tag{4}$$

where `F` is $M \times p$ and `W` is $M \times 1$ is interpreted as the specification of M frequency response measurements of a p-output system, with the kth row of `z.y=F` being the response at radian per second frequency given in the kth row of `z.w=W`. If `z.w` is not specified, it defaults to M regularly spaced points from 0 to the normalised folding frequency π (minus π/M).

Finally, in some areas (such as vibration analysis) it is common to conduct multiple experiments, each of which measures M frequency points over p outputs, but with each experiment probing ℓ different points in the structure. A novel feature of the toolbox is that it accommodates this case via `z.y` being a three-dimensional matrix of ℓ pages, each of which is the $M \times p$ observed response across M frequencies and p outputs.

4. Model structures

Depending on the data format, a range of model structures both linear and non-linear are accommodated. In all cases, the required model type is specified by user specified elements of a Matlab structure `m`.

4.1. Linear time invariant transfer function

The toolbox supports standard time domain transfer function models of the following form

$$y_t = \sum_{i=1}^{\ell} G^i(\rho,\theta)u_t^i + H(\rho,\theta)e_t \tag{5}$$

where $u_t^1, \ldots, u_t^\ell$ 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 $\mathbf{E}\{e_t^2\} = \sigma^2 < \infty$, and $\theta \in \mathbf{R}^n$ is a vector specifying the model parameters.

The operator ρ in (5) may be chosen as the standard shift operator q by setting `m.op = 'q'` (default), or the discrete time delta difference operator (Middleton & Goodwin, 1990) $\delta = (q-1)/T$ by setting `m.op = 'd'`. In this latter case, the sampling period T in seconds must also be specified by setting `m.T` to an appropriate value (default is $T=1$ s).

Therefore, multiple input, single output (MISO) transfer function model structures are supported, where the elements $G^i(\rho,\theta)$ and $H(\rho,\theta)$ are rational according to

$$G^i(\rho,\theta) = q^{-k_i}\frac{B^i(\rho,\theta)}{A^i(\rho,\theta)}, \quad H(\rho,\theta) = \frac{C(\rho,\theta)}{D(\rho,\theta)}, \tag{6}$$

$$A^i(\rho,\theta) = 1 + a_1^i\rho^{-1} + a_2^i\rho^{-2} + \cdots + a_{n_a^i}^i\rho^{-n_a^i}, \tag{7}$$

$$B^i(\rho,\theta) = b_0 + b_1^i\rho^{-1} + b_2^i\rho^{-2} + \cdots + b_{n_b^i}^i\rho^{-n_b^i}, \tag{8}$$

$$D(\rho,\theta) = 1 + d_1\rho^{-1} + d_2\rho^{-2} + \cdots + d_{n_d}\rho^{-n_d}, \tag{9}$$

$$C(\rho,\theta) = 1 + c_1\rho^{-1} + c_2\rho^{-2} + \cdots + c_{n_c}\rho^{-n_c}, \tag{10}$$

and k_i is a given (i.e. not estimated) delay acting on the ith input. These are specified by the elements of a vector `m.delay`, with the ith element being the delay, in samples, on the ith input. The default is to assume no delays.

Since any of the orders n_a^i, n_b^i, n_c, n_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). These can be specified by setting the element `m.type`, respectively, to one of the text strings `'fir'`, `'arx'`,`'armax'`, `'arma'`, `'oe'` or `'bj'`. For example

$$\gg \texttt{m.type} = \texttt{'bj'}; \tag{11}$$

specifies a Box–Jenkins model structure in which both $G(\rho,\theta)$ and $H(\rho,\theta)$ are present.

No model orders are specified in this case, and so defaults will be applied, which are fifth order dynamics for as many $G^i(\rho,\theta)$ as there are columns in `z.u` (i.e. $n_a^i = n_b^i = 5$, $i = 1, \ldots, m$) and first order dynamics for $H(\rho,\theta)$ (i.e. $n_c = n_d = 1$).

These defaults may be over-ridden by specifying the elements `m.A,m.B,m.C,m.D` to be vectors of integers, with the i'th row corresponding to the orders of $G^i(\rho,\theta)$ For example

$$\texttt{m.A} = [5\ ;\ 3], \texttt{m.B} = [4\ ;\ 4], \texttt{m.C} = 2\ ;\ \texttt{m.D} = 2\ ; \tag{12}$$

also results in a Box–Jenkins structure, this time with orders

$$\begin{bmatrix} n_a^1 \\ n_a^2 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \quad \begin{bmatrix} n_b^1 \\ n_b^2 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \quad n_c = 2, \quad n_d = 2. \tag{13}$$

Furthermore, initial estimates may be specified, which then also imply a model order. This can be achieved by setting an appropriate row of `m.A,m.B,m.C,m.D` to be a vector of real values. For example

$$\texttt{m.A} = [1, -1.8, 0.81]; \texttt{m.B} = [0.1, 0.5]; , \texttt{m.op} = \texttt{'q'}; \tag{14}$$

specifies a single-input, single-output (SISO) Output-Error structure with $n_a = 2, n_b = 1$ and for which any iterative search for a parameter

estimate is initialised at

$$G(q,\theta)=\frac{0.1+0.5q^{-1}}{1-1.8q^{-1}+0.81q^{-2}}. \tag{15}$$

In the case of frequency domain data, single-input single-output (SISO) transfer function structures of the following form are supported:

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

with G as described in (6)–(8) and ε_k in (16) accounting for modelling errors.

Here, ω_k is the radian per second value at which the frequency response $Y(\omega_k)$ has been measured, and γ_k is one of

$$\gamma_k=e^{j\omega_k T} \quad \text{or} \quad \gamma_k=j\omega_k \tag{17}$$

depending on whether the corresponding underlying operator ρ is taken as the shift q or continuous time derivative $\mathrm{d}/\mathrm{d}t$ operator. This choice is specified by setting the element `m.op` to be either the character `'q'` or `'s'`, respectively. If not specified, the default is `m.op='q'`.

Specification of the order of numerator and denominator in $G(\gamma,\theta)$, or initialisations for values therein proceeds as per the time domain case just discussed.

4.2. Linear time invariant state space

State space model structures are also supported, beginning in the linear time domain case with the representation

$$\begin{bmatrix}\rho x_t\\ y_t\end{bmatrix}=\begin{bmatrix}A & B\\ C & D\end{bmatrix}\begin{bmatrix}x_t\\ u_t\end{bmatrix}+\begin{bmatrix}w_t\\ e_t\end{bmatrix}, \tag{18}$$

or its "innovations" form equivalent

$$\begin{bmatrix}\rho x_t\\ y_t\end{bmatrix}=\begin{bmatrix}A & B\\ C & D\end{bmatrix}\begin{bmatrix}x_t\\ u_t\end{bmatrix}+\begin{bmatrix}K\\ I\end{bmatrix}\epsilon_t. \tag{19}$$

Here, both $u_t\in\mathbf{R}^{\ell}$ and $y_t\in\mathbf{R}^{p}$ may be vectors so that the multiple input, multiple output (MIMO) scenario can be accommodated, and u_t need not be specified in the case of time series modelling, in which case B and D are not included in the model structure.

The elements w_t,e_t and ϵ_t in (18), (19) are i.i.d. sequences of zero mean random variables with covariance structure

$$\mathrm{Cov}\left\{\begin{bmatrix}w_t\\ e_t\end{bmatrix}\right\}=\begin{bmatrix}Q & S^T\\ S & R\end{bmatrix}, \quad \mathrm{Cov}\{\epsilon_t\}=\Sigma. \tag{20}$$

The operator ρ may be specified as either the forward shift operator q by setting `m.op='q'` (default for regularly spaced samples) or the continuous time derivative $\mathrm{d}/\mathrm{d}t$ by setting `m.op='s'`, which is the default when the vector `z.t` is specified with irregularly spaced time stamps.

In this latter case, the toolbox formulates an associated time-varying linear discrete time model

$$x(t_{k+1})=F_k x(t_k)+G_k u(t_k)+\tilde{w}(t_k), \tag{21}$$

$$y(t_k)=H_k x(t_k)+D_k u(t_k)+e(t_k), \tag{22}$$

using the methods described in Ljung & Wills (2010) and for the purpose of likelihood or prediction error cost computation using a Kalman filter.

Employing state-space structures is achieved by setting

```
m.type='ss';
```
(23)

Which of the two possibilities (18), or its "innovations form" (19) is then employed depends on the algorithm used for computing the system estimate, as will be detailed shortly. In all cases, models (18), (19) are parametrized by a vector θ which contains all entries of all the system matrices.

With no model order specified, a default system with state dimension $\dim\{x_t\}=n_x=5$ is used. This may be altered. For example, the assignment

```
m.ss.A=9;
```
(24)

specifies a state space model of order n_x=9, with no need to also set `m.type='ss'`. Values for θ that are to be used as initial points in an iterative search for final estimates may be set by further specification in `m.ss`. For example, the declaration

```
m.ss.A=[1.8 -0.81; 1,0],m.ss.B=[1; 0];
```
(25)

```
m.ss.C=[0.1,0.5],m.ss.D=0;
```
(26)

specifies an n_x=2nd order model with evident initial state space entries.

A state space model structure can also be employed for estimation from frequency domain observations $\{Y(\omega_k)\}$. In this case the relationship (16), (17) still holds but $G(\gamma_k,\theta)$ with γ_k given by (17) is parametrized according to

$$G(\gamma_k,\theta)=C(\gamma_k I-A)^{-1}B+D. \tag{27}$$

Again, this may be achieved by setting `m.type='ss'`. Alternatively, it is the default if the data `z.y` has more than one column, indicating measurements from a multivariable system.

4.3. Nonlinear models

In addition to the linear model structures just described, the toolbox also supports important classes of non-linear model structures.

4.3.1. Hammerstein–Wiener

This model structure involves linear time invariant dynamics "sandwiched" between time invariant memory-less non-linearities (Bai, 1998). That is, it is an extension of (5) which can be described by

$$z_t=\sum_{i=1}^{\ell}G^i(q,\theta)X^i(u_t^i,\theta), \tag{28}$$

$$y_t=Z(z_t,\theta)+H(q,\theta)e_t, \tag{29}$$

where $G^i(q,\theta)$, $H(q,\theta)$ and the methods by which their orders or initial values are specified are all as per the previous discussion in Section 4.1.

Furthermore, in (28), each of the $X^i(u_t^i,\theta)$ are memory-less non-linearities, as is $Z(z_t,\theta)$ in (29). The parameters describing them are included in the vector θ that is estimated.

If the component of θ containing this description of the non-linearities is labelled η, then the memory-less functions $X(\cdot,\eta)$ supported by the toolbox (which are identical to the supported $Z(\cdot,\eta)$ functions) may be described as follows.

Polynomial

$$X(u_t,\eta)=\eta_1 u_t+\eta_2 u_t^2+\cdots+\eta_\kappa u_t^\kappa; \quad \eta=[\eta_1,\ldots,\eta_\kappa]^T \tag{30}$$

Saturation

$$X(u_t,\eta)=\begin{cases}\eta_3 u_t; & u_t\in[\eta_1,\eta_2]\\ \eta_2\eta_3; & u_t>\eta_2\\ \eta_1\eta_3; & u_t<\eta_1;\end{cases} \qquad \eta=[\eta_1,\eta_2,\eta_3]^T \tag{31}$$

Deadzone

$$X(u_t,\eta)=\begin{cases} u_t-\eta_2; & u_t>\eta_2 \\ u_t-\eta_1; & u_t<\eta_1 \\ 0; & u_t\in[\eta_1,\eta_2], \end{cases} \qquad \eta=[\eta_1,\eta_2]^T \tag{32}$$

Note that since the linear dynamics in a Hammerstein–Wiener model can accommodate overall gain, there is no loss of generality by assuming it to be one in the linear regions outside the deadzone. However, this is not the case in the saturation situation, which explains the need for the η_3 component in (31).

Piecewises Linear (*Hinging Hyperplane*) This non-linearity is described by a set of "hinge" functions of the form

$$X_k(u_t,\eta_k)=\begin{cases} \eta_{1,k}+\eta_{2,k}u_t; & u_t>-\eta_{1,k}/\eta_{2,k}, \\ 0; & \text{Otherwise}, \end{cases}$$

which are zero up to a "breakpoint", and linear thereafter. The overall piecewise linear function is then the superposition of (say κ of) these hinges:

$$X(u_t,\eta)=\eta_0+\eta_1u_t+\sum_{k=1}^{\kappa}X_k(u_t,\eta_k), \tag{33}$$

$$\eta=[\eta_0,\eta_1,\eta_{1,1},\eta_{2,1},\ldots,\eta_{1,\kappa},\eta_{2,\kappa}]^T. \tag{34}$$

In order to employ a Hammerstein–Wiener type structure, the user first specifies the type and order of the linear transfer function via the methods described in Section 4.1.

Considering first the SISO situation of input dimension $\ell=1$, any memoryless non-linear components on input and output in the model structure (28), (29) are specified by setting the model structure components `m.in.type` and `m.out.type` to be one of the strings `'polynomial'`, `'saturation'`, `'deadzone'` or `'hinge'` according to which of the previously described non-linearities are, respectively, required.

The linear case of $X(u_t,\theta)=u_t$ on input and $Z(z_t,\theta)=z_t$ on output, which corresponds to the settings `m.in.type='linear'`, `m.out.type='linear'` is the default used in the toolbox unless specified otherwise.

Depending on the type of non-linearity, various parameters describing it can also be specified. In the case of deadzone, two extra parameters `m.in.lower` and `m.in.upper` corresponding to η_1 and η_2 in (32) may be given initial estimates, and if not default to values $-\eta_1=\eta_2=0.1$. In the case of saturation, these parameters are also available for pre-specification, as is a third `m.in.gain`.

For both the polynomial and hinging hyperplane cases, a vector `m.in.eta` corresponding to an initial estimate may be specified. If not given then (respectively) a 6th order polynomial or a four section piecewise linear function are taken as defaults.

Therefore, as an example, the commands

```
>> m.type='oe'; m.A=3;
>> m.in.type='deadzone',
>> m.in.upper=0.4;
>> m.in.lower=-0.1;
>> m.out.type='polynomial',
>> m.out.eta=[0,1];
```

specify a Hammerstein–Wiener structure consisting of a 3rd order output-error transfer function structure sandwiched between a deadzone that nulls inputs of size $-0.1<u_t<0.4$ and a polynomial that squares the output.

4.3.2. Bilinear

A further class of non-linear models accommodated by the toolbox are MIMO bilinear ones of the following form (Favoreel, De Moor, & Van Overschee, 1999):

$$\begin{bmatrix} x_{t+1} \\ y_t \end{bmatrix}=\begin{bmatrix} A & F & B \\ C & G & D \end{bmatrix}\begin{bmatrix} x_t \\ u_t\otimes x_t \\ u_t \end{bmatrix}+\begin{bmatrix} w_t \\ v_t \end{bmatrix},\begin{bmatrix} K \\ I \end{bmatrix}\epsilon_t \tag{35}$$

where $\otimes$ represents the Kronecker tensor product (Brewer, 1978), the noise modelling may be of either form discussed in relation to (18) or (19), and $y_t\in\mathbf{R}^p$, $u_t\in\mathbf{R}^\ell$ may be multivariable with arbitrary $p,\ell\geq 1$.

This structure is invoked by the simple specification

```
>> m.type='bilin';
```

with the orders for the system matrices, or initial guesses for them, being specified as per the linear state space case explained in (24)–(26).

4.3.3. General non-linear

A final class of non-linear models is the general state space structure

$$x_{t+1}=f_t(x_t,u_t,w_t,\theta), \tag{36}$$

$$y_t=h_t(x_t,u_t,e_t,\theta), \tag{37}$$

where the mappings f_t and h_t may be quite arbitrary and possibly time varying. As in the previous cases, the output $y_t\in\mathbf{R}^p$ and input $u_t\in\mathbf{R}^\ell$ may be multivariable with arbitrary $p,\ell\geq 1$. Additionally, the stochastic disturbances w_t and e_t may have arbitrary distributions, but the form of the associated densities $p_w(\cdot)$ and $p_e(\cdot)$ must be known.

This structure is employed by the specification

```
>> m.type='nlss';
```

The form of the mappings f_t, h_t and the densities p_w, p_e are embodied in three functions that must also be specified, and which will be illustrated in the example Section 8 following.

5. Estimation methods and algorithms

With the model structures supported by the toolbox now defined, this section details the range of toolbox-implemented estimation methods employing these structures. The algorithms used in the toolbox to implement these methods are also described.

5.1. Minimisation of a least squares criterion

The default estimation method, for all the model structures just presented, is the "least squares" one that is defined as the solution of the following optimisation problem:

$$\widehat{\theta}_N\triangleq\arg\min_{\theta\in\mathbf{R}^n}V_N(\theta), \tag{38}$$

where the criterion $V_N(\theta)$ defined as

$$V_N(\theta)\triangleq\mathrm{Tr}\{E(\theta)^\star E(\theta)\} \tag{39}$$

(with $\cdot^\star$ denoting conjugate transpose) is a quadratic function of the error vector

$$E^T(\theta)\triangleq[\varepsilon_1(\theta),\ldots,\varepsilon_N(\theta)] \tag{40}$$

of differences between the observed data and the response of a model parametrized by θ.

In the case of time domain data, the elements of (40) are defined by

$$\varepsilon_t(\theta) \triangleq y_t - \widehat{y}_{t|t-1}(\theta) \tag{41}$$

where $\widehat{y}_{t|t-1}(\theta)$ 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) $\widehat{y}_{t|t-1}(\theta)$ is computed by a square root implementation of a Kalman filter, while in the transfer function case $\widehat{y}_{t|t-1}(\theta)$ is computed via the steady state Kalman filter in transfer function form; viz.

$$\widehat{y}_{t|t-1}(\theta) = H^{-1}(\rho,\theta)Z(z_t,\theta) + [1 - H^{-1}(\rho,\theta)]y_t,$$
$$z_t = \sum_{i=1}^{\ell} G^i(\rho,\theta)X^i(u_t^i,\theta). \tag{42}$$

In the situation of estimation from frequency domain data, the elements $E(\theta)$ are defined via (16) as

$$\varepsilon_k(\theta) \triangleq Y(\omega_k) - G(\gamma_k,\theta) \tag{43}$$

with $G(\gamma_k,\theta)$ given by (6)–(8), (17) in the case of transfer function modelling, and by (17), (27) in the case of state space descriptions.

In all these situations, the estimate $\widehat{\theta}_N$ satisfying the least squares criterion (38) is computed via one of a selection of gradient search based techniques that all depend on computing the Jacobian matrix

$$J(\theta) = \frac{\partial}{\partial\theta}\mathrm{vec}\{E(\theta)\} \tag{44}$$

of the error vector (40). A standard Gauss–Newton based search for $\widehat{\theta}_N$ then involves iterations $\{\theta_k\}$ starting from an initial guess θ_0 which are refined according to

$$\theta_{k+1} = \theta_k + \mu p, \tag{45}$$

where μ is a scalar "step length" and the search direction p is a solution of

$$[J(\theta_k)^T J(\theta_k) + \delta I]p = -J(\theta_k)^T E(\theta_k). \tag{46}$$

Here, $\delta = 0$ (the default in the toolbox) implies a Gauss–Newton method, while $\delta > 0$ implies a "Levenberg–Marquardt" update strategy (Dennis & Schnabel, 1983).

These (and other) choices of optimisation algorithm are specified by elements of the `opt` structure in the function call (1). The default search direction is a certain "robust" Gauss–Newton based technique developed in Wills & Ninness (2008), which may also be explicitly selected via a specification `opt.dir = 'rgn'`.

This may be altered to a Levenberg–Marquardt one by setting the algorithm search direction as `opt.dir = 'lm'`, with δ being the value specified in `opt.delta` (default $=10^5$). Other options, whose theoretical formulation will not be discussed here, are also available. These include trust region methods `opt.dir = 'trust'`, quasi-Newton BFGS methods `opt.dir = 'bfgs'`, and pure gradient search `opt.dir = 'grad'` (Dennis & Schnabel, 1983).

All these gradient-based methods may terminate in two ways. The first, most usual case is if the relative decrease of the cost at an iteration falls below the value specified in `opt.mdec` (default $=10^{-9}$). The second is if the number of iterations grows greater than the upper limit specified in `opt.miter` (default $=100$).

Setting the initial value θ_0 for the gradient search is also done in two ways. For transfer function models, the Steiglitz–McBride method over five iterations is employed to initialise the input–output dynamics, and the Hannan–Rissanen method is used to initialise the noise model. For state-space models, an N4SID subspace-based estimate as described in the following Section 5.3 is used.

5.2. Maximum likelihood

When employing any of the state space model structures (18), (19), (27) or (35), in the Gaussian case the toolbox also supports estimation according to the Maximum-Likelihood (ML) criterion

$$\widehat{\theta}_N \triangleq \arg\min_{\theta\in\mathbf{R}^n} L(\theta), \tag{47}$$

where $L(\theta)$ is the negative log-probability of the observed data.

In the case of the innovations form structures (19) and (35), the optimisation problem (47) reduces to (Söderström & Stoica, 1989)

$$\widehat{\theta}_N \triangleq \arg\min_{\theta\in\mathbf{R}^n} \det\left(\sum_{t=1}^{N} \varepsilon_t(\theta)\varepsilon_t^T(\theta)\right). \tag{48}$$

Being a smooth optimisation problem, the toolbox handles this via the gradient based search strategies explained in the previous section. The determinant based cost in (48) must be specified via the setting `opt.cost = 'det'`. It can be reverted to the default least squares cost (39) via the setting `opt.cost = 'trace'`.

Alternate to this approach, the toolbox also implements the Expectation-Maximisation (EM) algorithm (Dempster, Laird, & Rubin, 1977) for the computation of ML estimates of the state space model structures (19) and (35). This is invoked by setting `opt.alg = 'em'`, and can be reverted to the gradient based search algorithm by setting `opt.alg = 'gn'`.

This algorithm has not been commonly employed in system identification applications. However, the authors have found it to be very effective due to its robustness against trapping in local minima, and its modest computational requirements, especially for models of high input–output and/or state dimension (Gibson & Ninness, 2005; Gibson et al., 2005). One contribution of the toolbox is it allows other researchers to very simply evaluate this perhaps unappreciated approach.

The essence of the method is to use an initial estimate of the state space system matrices A,B,C,D to compute an estimate of the state sequence $\{x_t\}$ via its Kalman-smoothed values.

These are then used to re-estimate the system matrices A,B,C,D in via linear regression. This Kalman-smoothing/linear regression process is then repeated. Full details are available in Gibson & Ninness (2005) and Gibson et al. (2005). The algorithm terminates when either the relative likelihood increase is below `opt.mdec` or the iteration count exceeds the value specified by `opt.miter`.

Finally, in the case of frequency domain measurements $\{Y(\omega_k)\}$ and Gaussian noise, the joint log-likelihood of $Y(\omega_1),\ldots,Y(\omega_N)$ is (Wills et al., 2009)

$$L(\theta) = \sum_{k=1}^{N} \log|P_k| + \|P_k^{-1/2}(Y_k - G(\gamma_k))\|^2, \tag{49}$$

where

$$P_k \triangleq CA_k^{-1}QA_k^{-\star}C^T + R, \quad A_k \triangleq \gamma_k I - A. \tag{50}$$

As per the time domain case, the toolbox supports both gradient-based and EM algorithm-based approaches for computing the ML estimate (47) according to how `opt.alg` is specified. The EM-based method is a new technique developed by the authors in Wills et al. (2009).

5.3. Subspace-based projection/realisation

Again in the situation of linear state space modelling of the form (19) or its frequency domain equivalent (27), the toolbox

offers a further estimation option via "subspace-based" approaches. The advantage of these techniques is that they are not iterative, and hence do not require initialisation.

In the case of time domain data, these methods depend on the state space structure (19) with operator $\rho = q$ implying that

$$Y_f = \beta Z_p + \mathcal{U}_f U_f + \varepsilon, \tag{51}$$

where for some user-chosen integer forward horizon length f,

$$\mathcal{U}_f \triangleq \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ CA^{f-2}B & \cdots & \cdots & D \end{bmatrix} \tag{52}$$

and

$$Y_f = \begin{bmatrix} y_t & y_{t+1} & \cdots & y_{t+N-1} \\ \vdots & \vdots & & \vdots \\ y_{t+f-1} & y_{t+f} & \cdots & y_{t+N+f-2} \end{bmatrix} \tag{53}$$

with U_f being defined in an identical manner to Y_f but using the input observations u_t. Additionally, with p being a user-chosen past horizon length

$$Z_p = \begin{bmatrix} u_{t-1} & u_t & \cdots & u_{t+N-2} \\ \vdots & \vdots & & \vdots \\ u_{t-p} & u_{t-p+1} & \cdots & u_{t+N-1-p} \\ y_{t-1} & y_t & \cdots & y_{t+N-2} \\ \vdots & \vdots & & \vdots \\ y_{t-p} & y_{t-p+1} & \cdots & y_{t+N-1-p} \end{bmatrix} \tag{54}$$

while ε is a "residual" dependent on the noise process $\{\epsilon_t\}$ in (19). The defaults in the toolbox for the horizons f and p are 2.5 times the state space dimension, but this may be altered by specifying `opt.horizon`.

Furthermore

$$\beta = \mathcal{O}_f \mathcal{K}_p. \tag{55}$$

where

$$\mathcal{O}_f \triangleq \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{f-1} \end{bmatrix}, \quad \mathcal{K}_p \triangleq \left[\tilde{B}, \ldots, \tilde{A}^{p-1}\tilde{B}, K, \ldots, \tilde{A}^{p-1}K\right], \tag{56}$$

$$\tilde{A} \triangleq A - KC, \quad \tilde{B} \triangleq B - KD. \tag{57}$$

This allows the state-space structure (19) to be considered as a linear regression (51) in parameter vectors $\beta, \mathcal{U}_f$ which via (52), (56), (57) are completely dependent on the state-space system matrices A,B,C,D,K.

All subspace-based system identification methods supported in the toolbox (and indeed the very great majority of methods existing in the literature) exploit this fact by first obtaining a joint estimate of $[\beta, \mathcal{U}_f]$ via minimisation of the least squares criterion

$$[\widehat{\beta}, \widehat{\mathcal{U}_f}] = \arg\min_{\beta, \mathcal{U}_f} \|Y_f - (\beta Z_p + \mathcal{U}_f U_f)\|_F. \tag{58}$$

Here $\|A\|_F$ denotes Frobenius norm, in which case the optimisation problem (58) is a linear regression with straightforward closed form solution (Golub & Loan, 1989). This estimate for $\widehat{\beta}$ is then factored to deliver estimates of $\mathcal{O}_f$ and $\mathcal{K}_p$ by first forming the singular value decomposition

$$W_f \widehat{\beta} W_p = [U_1, U_2] \left[\begin{array}{cc|c} S_1 & \phi & \\ \phi & S_2 & \phi \end{array}\right] \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \approx U_1 S_1 V_1^T, \tag{59}$$

where in forming the approximation it has been assumed that the singular values in S_1 dominate those in S_2. Via (55) this implies estimates

$$\widehat{\mathcal{O}}_f \triangleq W_f^{-1} U_1 (S_1)^{1/2}, \quad \widehat{\mathcal{K}}_p \triangleq (S_1)^{1/2} V_1^T W_p^{-1}. \tag{60}$$

The matrices W_f and W_p are user chosen weightings. The next step involves noting that the definitions (51)–(52) imply that $\beta Z_p = \mathcal{O}_f X_t$ where $X_t \triangleq [x_t, \ldots, x_{t+N-1}]$, and hence the state sequence may be estimated as

$$\widehat{X}_t = \arg\min_{X_t} \|\widehat{\beta} Z_p - \widehat{\mathcal{O}}_f X_t\|_F \tag{61}$$

which again is a linear regression with simple closed form solution (Golub & Loan, 1989). Finally, this state estimate allows the system matrices to be estimated via solving the linear regression

$$\widehat{A}, \widehat{B}, \widehat{C}, \widehat{D} = \arg\min_{A,B,C,D} \left\| \begin{bmatrix} \widehat{X}_{t+1} \\ Y_t \end{bmatrix} - \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \widehat{X}_t \\ U_t \end{bmatrix} \right\|_F. \tag{62}$$

This approach, in combination with the weighting matrix choices

$$W_f = I, \quad W_p = Z_p$$

is known as the N4SID method (van Overschee & de Moor, 1994; van Overschee & Moor, 1996). It may be selected for use in the toolbox by specifying `opt.alg='n4sid'`, and is the default as well.

Another option, known as "Canonical Correlation Analysis" (CCA) (Larimore, 1990; Ljung, 1999) may be selected by specifying `opt.alg='cca'`. In this case the weighting matrices are taken as

$$W_f = (Y_f \Pi_{U_f} Y_f^T)^{-1/2}, \quad W_p \triangleq (Z_p \Pi_{U_f} Z_p^T)^{1/2}. \tag{63}$$

$$\Pi_{U_f} \triangleq I - U_f^T (U_f U_f^T)^{-1} U_f \tag{64}$$

and the extraction of system estimates is done in a way that is different to (62), but of a similar theme (Ljung, 1999).

In case of frequency domain data, the toolbox implements the subspace based algorithm presented and analysed in McKelvey, Akçay, & Ljung (1996) wherein taking the DFT at frequency ω_k of both sides of the state space model (19) implies

$$\Omega_{1,f} \mathcal{Y} = \mathcal{O}_f \mathcal{X} + \mathcal{U}_f \Omega_{\ell,f} + V, \tag{65}$$

with ℓ being the number of inputs (columns of B) in the model (19), and with the definitions (50) and $X(\omega_k) \triangleq A_k^{-1} B$:

$$\mathcal{Y} \triangleq \operatorname*{diag}_{1 \le k \le N} Y(\omega_k), \quad \mathcal{X} \triangleq [X(\omega_1), \ldots, X(\omega_N)], \tag{66}$$

$$\Omega_{\ell,f} \triangleq \begin{bmatrix} I_\ell & I_\ell & I_\ell & I_\ell \\ \gamma_1 I_\ell & \gamma_2 I_\ell & \cdots & \gamma_M I_\ell \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_1^{f-1} I_\ell & \gamma_2^{f-1} I_\ell & \cdots & \gamma_M^{f-1} I_\ell \end{bmatrix}. \tag{67}$$

Finally, V in (65) depends on the DFT of the innovations ε_t. As per the time domain case, (65) is a linear regression, with all information parametrizing dynamics contained in $\mathcal{O}_f \mathcal{X}$ and $\mathcal{U}_f$.

Using the over-bar notation $\overline{\mathcal{X}} = [\mathrm{Re}\{\mathcal{X}\}, \mathrm{Im}\{\mathcal{X}\}]$ to denote splitting into real and imaginary components, real valued estimates of $\beta \triangleq \mathcal{O}_f \overline{\mathcal{X}}$ and $\mathcal{U}_f$ are then obtained from the pseudo-inverse based closed form solution to

$$[\widehat{\beta}, \widehat{\mathcal{U}_f}] = \arg\min_{\beta, \mathcal{U}_f} \|\overline{\Omega_{1,f} \mathcal{Y}} - (\beta + \mathcal{U}_f \overline{\Omega_{\ell,f}})\|_F. \tag{68}$$

As in the time domain case $\widehat{\beta}$ is extracted and $\mathcal{O}_f$ is estimated by the SVD-based factorisation (59), but in this case with $W_p = W_f = I$. Estimates $\widehat{A}$, $\widehat{C}$ are then extracted from $\widehat{\beta}$ in the same way as for the time domain CCA case. The estimates of B and

D are obtained by solving the linear regression problem

$$\widehat{B},\widehat{D} = \arg\min_{B,D} \sum_{k=1}^{N} \|Y(\omega_k) - [D + \widehat{C}(\gamma_k I - \widehat{A})^{-1}B]\|_F^2. \quad (69)$$

This algorithm is selected when the data structure element `z.y` contains complex valued measurements and `opt.alg = 'sid'` is specified.

5.4. Bayesian methods

Bayesian estimation of SISO linear transfer function models is supported by the provision of tools for computing the posterior density

$$p(\theta|y_1, \ldots y_N) \quad (70)$$

and the conditional mean point estimate

$$\widehat{\theta} = \mathbf{E}\{\theta|y_1, \ldots, y_N\}. \quad (71)$$

This is achieved by calling the `postdist` command

```
g = postdist(z, g)
```

which takes a model structure `g` and uses the Markov-Chain Monte-Carlo (MCMC) approach described in Ninness & Henriksen (2010) to return the conditional mean estimate as `g.A,g.B,g.C,g.D` and also return the associated posterior density as `g.pa, g.pb`. The latter may be conveniently plotted using the call `showdist(g)`.

5.5. Non-parametric methods

This class of methods, as their epithet implies, do not involve estimation of a parameter vector θ or indeed an assumption of a particular type of model structure. Rather, a frequency response estimate $\widehat{G}(\omega)$ at specified frequencies $\{\omega_k\}$ is the deliverable. The toolbox implements two well known methods for achieving this via the specification `m.type = 'nonpar'`. The frequencies $\{\omega_k\}$ may be specified by the elements of a vector `m.w`, but if not, default to a logarithmically spaced set finishing at the folding frequency.

5.5.1. Blackman–Tukey estimate

Define (respectively) the sample auto-covariance of the observed input u_t and the sample cross-covariance between u_t and observed output y_t as

$$\widehat{R}_u(\tau) \triangleq \sum_{t=1}^{N} u_t u_{t-\tau}, \quad \widehat{R}_{yu}(\tau) \triangleq \sum_{t=1}^{N} y_t u_{t-\tau}, \quad (72)$$

where quantities are periodically extended (e.g. $u_\tau = u_{\tau-N}$ for $\tau < 1$) to accommodate negative indices. These allow the formation of estimated auto- and cross-spectral densities as

$$\widehat{\Phi}_u(\omega) = \sum_{\tau=-N}^{N} w_\tau \widehat{R}_u(\tau) e^{-j\omega\tau T}, \quad (73)$$

$$\widehat{\Phi}_{yu}(\omega) = \sum_{\tau=-N}^{N} w_\tau \widehat{R}_{yu}(\tau) e^{-j\omega\tau T}, \quad (74)$$

where $\{w_t\}$ is an arbitrary window sequence and extension of quantities, but this time symmetric (e.g. $\widehat{R}_u(-\tau) = \widehat{R}_u(\tau)$) is used in the case of negative indices. The so-called "Blackman–Tukey" estimate implemented in the toolbox is provided as

$$\widehat{G}(\omega) = \widehat{\Phi}_{yu}(\omega)/\widehat{\Phi}_u(\omega). \quad (75)$$

It is the default method, and may also be chosen by setting `opt.alg = 'bltuk'`. The window $\{w_t\}$ defaults to a Hamming one, but may be set as Bartlett, Hanning and Boxcar, via `opt.window`. For example, `opt.window = 'hanning'`. The window length is by default 20% of the data length, but may be set arbitrarily via specification of the integer `opt.N`.

Table 1
Summary of supported model structures and employed algorithms. NB. TF, Transfer Function; SS, State Space; H/W, Hammerstein–Wiener; GN, Gradient Based Search; SID, Subspace Identification.

Time domain data							
Model Type		**Algorithm**			**Operator**		
		GN	EM	SID	q	δ	s
Linear	TF	✓	×	×	✓	✓	×
	SS	✓	✓	✓	✓	×	✓
Non-linear	H/W TF	✓	×	×	✓	×	×
	Bilin. SS	✓	✓	×	✓	×	×
	Gen'l. SS	×	✓	×	✓	✓	✓
Frequency domain data							
Linear	TF	✓	×	×	✓	×	✓
	SS	✓	✓	✓	✓	×	✓

5.5.2. Empirical transfer function estimate

Via the specification `opt.alg = 'etfe'`, the toolbox implements the non-parametric empirical transfer function estimate (ETFE) given by (Ljung, 1999)

$$\widehat{G}(\omega) = Y_N(\omega)/U_N(\omega), \quad (76)$$

where Y_N, U_N are, respectively, the windowed discrete Fourier transforms (DFTs) of y_t and u_t. As per the previous case, the default window is a Hamming one of 20% data length, but may be altered as just explained.

6. Capabilities summary

As is evident from the preceding discussion, the toolbox supports a range of data formats, model structures and algorithms. However, the availability of these options is coupled. For example, delta operator models are only supported in transfer function formats, and continuous time model estimation with respect to time domain data are only supported for state-space model structures.

For ease of understanding, these capabilities and couplings are summarised in Table 1. Where a capability is not available, there is an underlying technical reason. For example, subspace identification (SID) methods are not available for transfer function model structures since they are specifically developed to address state space structures.

7. Application example

In order to illustrate the application and performance of the toolbox, this section discusses the problem of identifying a linear system model of a lightly damped system from frequency domain measurements.

7.1. Apparatus description

The physical apparatus studied here consists of an aluminium beam, clamped at one end but free at the other, as shown in Fig. 1 and illustrated diagrammatically in Fig. 2. It is 970 mm in length, 5 mm in thickness, and 25 mm in width.

Two *Physik Instrumente* PIC151 ceramic piezoelectric transducers are bonded to the beam with centres 105 mm from the

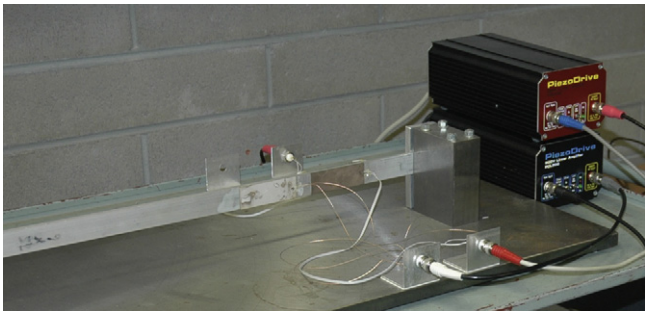

Fig. 1. Experimental apparatus.

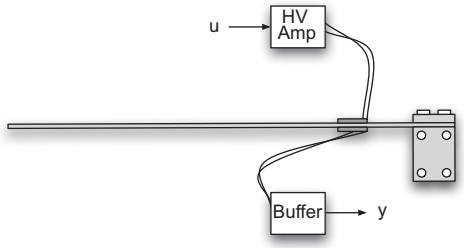

Fig. 2. Plan view schematic of the experimental apparatus.

clamped base, with configuration as shown in Fig. 2. One transducer is driven by a 200 V PDL200 high voltage amplifier in order to induce lateral beam displacements that are proportional to the applied voltage. The input to this amplifier is taken as the input excitation signal u_t.

The resulting beam displacement at any point is proportional to the mechanical strain at that point, and this is measured by buffering and acquiring the induced open-circuit voltage of the piezoelectric laminate also mounted 105 mm from the base, but on the opposite side to the actuating transducer laminate. This displacement measurement is taken as the system response signal y_t.

The physical modelling of the beam itself is very well studied, and well accepted to be the solution of a Bernoulli–Euler partial differential equation (Meirovitch, 1996). As established in Alberts, DuBois, & Pota (1995) and Pota & Alberts (1995), this can be simply modified to include the piezolaminates, and the solution of the resulting equation can be accurately represented via a linear time invariant system that is the sum of second order resonant transfer functions. The number of functions, and hence the order of the overall model is determined by the bandwidth of interest, and can therefore be high.

This apparatus has been chosen for study since it is challenging from a system identification viewpoint, while also being relevant to a range of interesting problems, particularly in the area of active vibration control (Meirovitch, 1996).

7.2. Frequency domain system identification

For the purposes of presenting some of the available frequency domain system identification methods, the input–output response of the beam in Figs. 1 and 2 was measured using a Hewlett–Packard HP89410A vector signal analyzer. This equipment is designed for frequency response measurement. While it offers several approaches to achieve this, for the measurements used here it injected a repeated chirp signal excitation. It then measures the output, computes its windowed discrete Fourier transform (DFT) over each completed chirp cycle, and then averages the results.

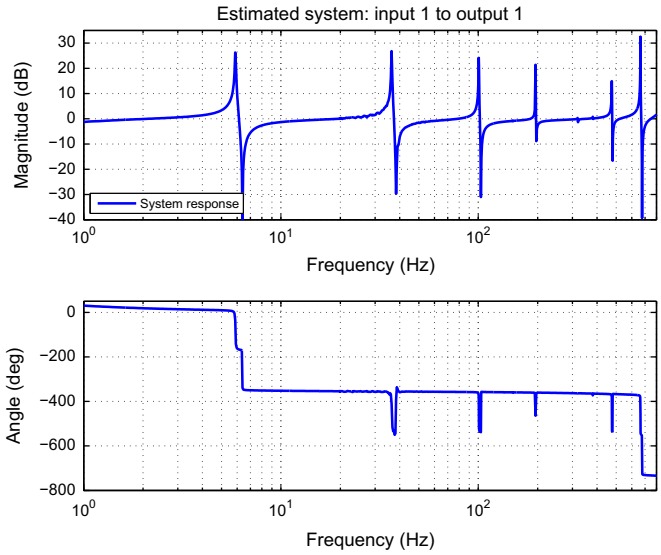

Fig. 3. System frequency response measured by HP89410A analyzer.

The resulting measured magnitude frequency response over a 1 kHz frequency range is shown as the solid line in Fig. 3.

The first problem addressed in this section is that of estimating a linear system model whose frequency response matches the measured response shown in Fig. 3.

Considering Fig. 3, it is clear that (as mentioned in the previous section) the system involves six lightly damped resonant pole/zero combinations, together with two further minor ones between the 4th and 5th significant ones.

This suggests the need for linear system description of at least order 16 is required to model the system. This was tested, and in fact a model of order 18 was found necessary. To explain the underlying process, it is assumed the vector `F` of the complex valued frequency measurements displayed in Fig. 3 together with associated vector `W` of radian per second frequencies at which they were obtained are specified in a data structure `z` according to (4).

An 18th order linear transfer function model structure of the form (16) using the shift operator q may then be specified by executing the commands

```
>> m.A = 18; m.type = 'oe'; m.op = 'q';
```
(77)

A model of this order is then computed using the default least squares criterion (38)–(40), (43) and the default robust Gauss–Newton algorithm (Wills & Ninness, 2008) by executing the command

```
>> gtf = est(z,m);
```
(78)

This returns a structure `gtf` containing over 30 elements that define the estimated model, the details of the estimation algorithm settings used, and any error bounding analysis that was also computed. A summarised overview of this information can be presented to the screen by executing the command `details(gtf)`.

Of primary interest is to compare the frequency response of the estimated model `gtf` with the measured frequency response `z`. This can be simply achieved with the `showbode` command as follows:

```
>> showbode(z,gtf);
```
(79)

with results shown in Fig. 4. Clearly, while this estimation approach (77), (78) produces a model capable of accurately representing the measured response above 10 Hz, it completely

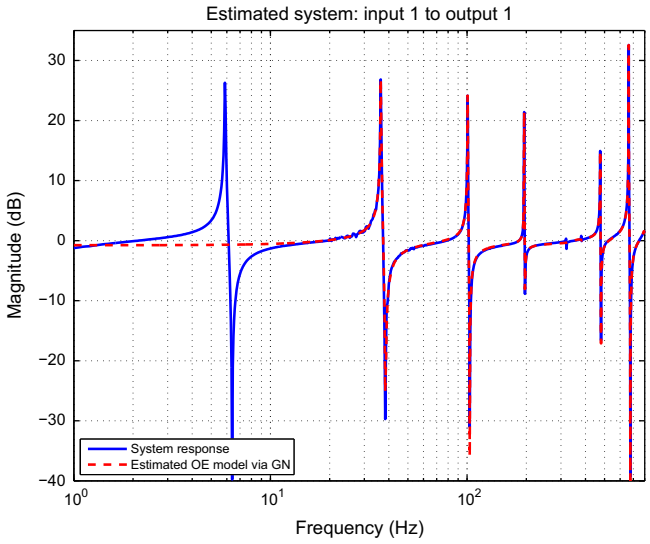

Fig. 4. Measured magnitude frequency response together with response of estimated models 18th order transfer function model.

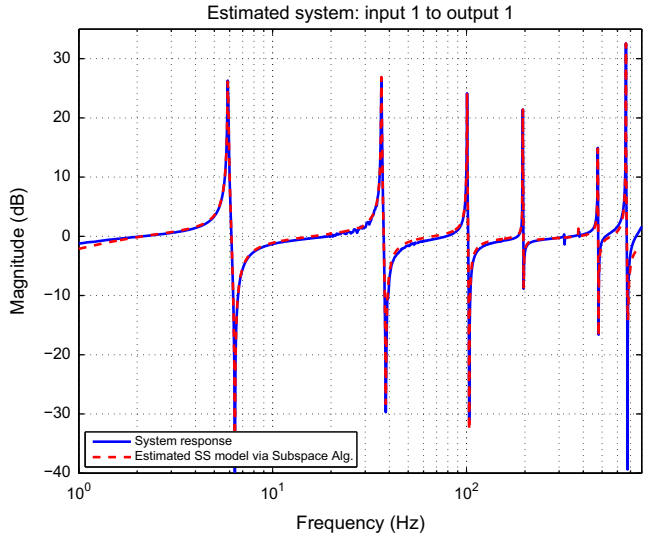

Fig. 5. Measured magnitude frequency response together with response of estimated 18th order state space model computed using a subspace-based method.

misses the resonant mode at 6 Hz. This suggests exploring an alternative approach, the simplest of which is the subspace-based technique explained in (65)–(69). This can be achieved by selecting a state space model structure and specifying that a subspace algorithm is to be used:

```
≫ m.type = 'ss'; m.op = 's'; opt.alg = 'sid';
```
(80)

Note, that for the purposes of illustration, a continuous time model structure is now also being trialled. Computing the estimate then proceeds via the function call

```
≫ gsid = est(z,m,opt);
```
(81)

The results of executing `showbode(z,gsid)` to display model response versus measured response are illustrated in Fig. 5. This illustrates an overall more accurate modelling than in the transfer function case. More precisely, the mode at 6 Hz is now captured, at the expense of the high frequency mode at around 700 Hz being captured less accurately.

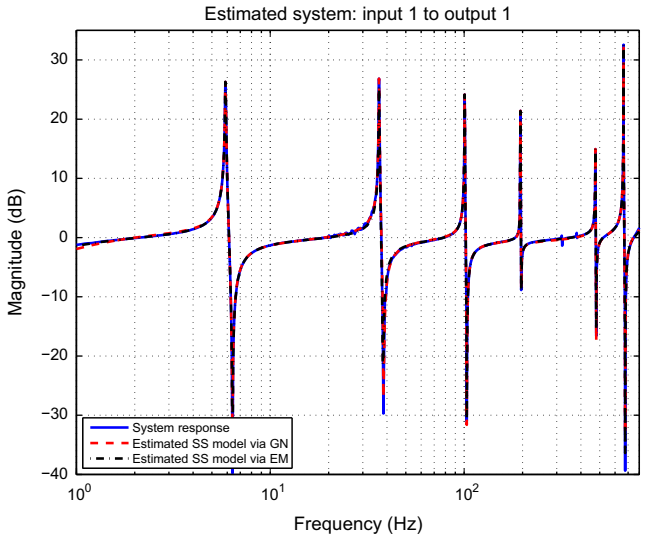

Fig. 6. Measured magnitude frequency response together with response of estimated 18th order state space models computed using robust Gauss–Newton gradient based search, and the EM algorithm, both initialised at the subspace-based estimate illustrated in Fig. 5.

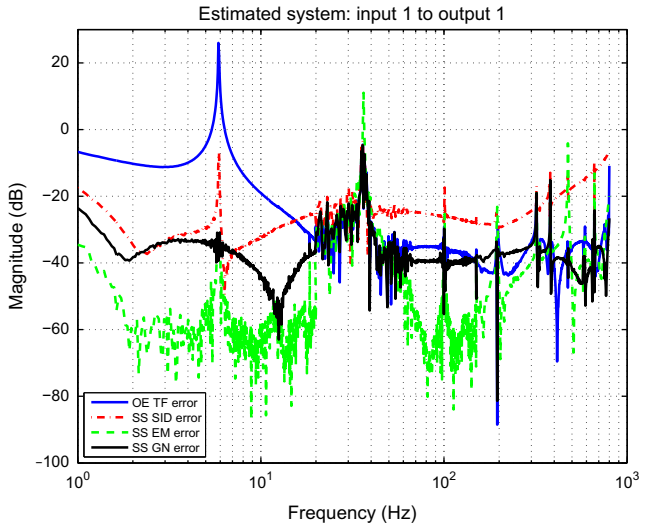

Fig. 7. Magnitude of modelling error for each of the estimated models shown in the previous figures.

This raises the question of whether either gradient based or expectation–maximisation based search can be used to improve on this estimate. This is achieved by simply specifying the required algorithm in `opt.alg`, and using the returned estimate `gsid` as the model structure specification so that the iterative search is initialised at that model; viz.

```
≫ opt.alg = 'gn'; ggn = est(z,gsid,opt);
```
(82)

```
≫ opt.alg = 'em'; gem = est(z,gsid,opt);
```
(83)

The command `showbode(Z,ggn,gem)` then shows the results illustrated in Fig. 6. Both of these estimates capture the measured frequency response more accurately than either of the transfer function or subspace-based ones preceding them. To examine this, together with any differences between the accuracy of the estimates shown in Fig. 6, the magnitudes of the frequency response modelling error for all four estimated models are shown

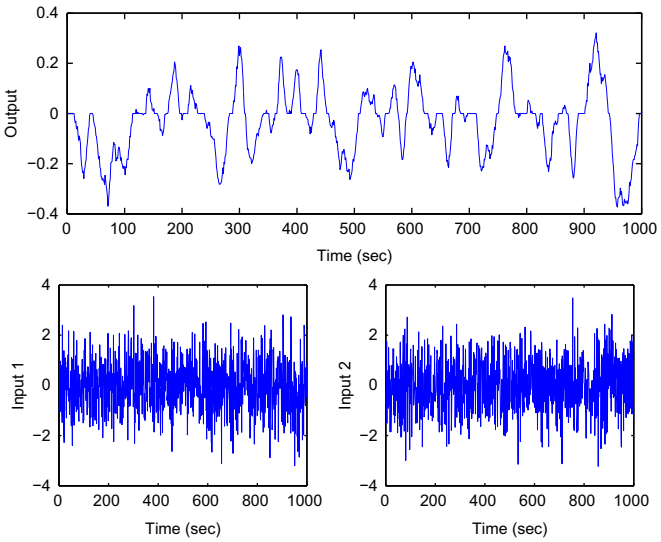

Fig. 8. Observed MISO Input/Output Data from Hammerstein–Wiener System.

in Fig. 7. This illustrates that the state-space model computed using the EM algorithm generally provides the most accurate modelling, with estimation errors below -60 dB over about one half of the frequency range of interest.

In relation to this bode plot presentation, it is worth mentioning that the toolbox also delivers estimated model structures as Matlab dynamic system objects compatible with the Mathworks control systems toolbox and other commercial (Ljung, 2012) and free (Garnier et al., 2012; Young & Taylor, 2012) software packages. For example, the command `bode(gem.sysG)` will deliver a version of the Bode plots via the Matlab control systems toolbox functionality.

8. Simulation example

In order to illustrate the non-linear system estimation capabilities of the toolbox, we now turn to a simulation example involving the observed MISO data shown in Fig. 8. This comes from a system with saturation and dead-zone non-linearities on inputs 1 and 2 (respectively), followed by linear systems of order 4 and 3. The summed outputs are then passed through a dead-zone non-linearity.

In order to estimate a MISO Hammerstein–Wiener 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='saturation';
≫ m.in(2).type='deadzone';
≫ m.out.type='deadzone';
≫ g=est(z,m);
```

Note that now two model orders $m_a^1=4$, $m_a^2=3$ 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 a SISO model. Note also that in this MISO situation of $\ell=2$ (in (28), (29)) the model structure `m` contains an array of two input substructures, each of which is set independently.

The results of this estimation experiment can be summarised via use of the `details` command.

```
≫ details(g)
_____________________________________________________

Details for Estimated Model Structure
_____________________________________________________

Operator used in model=q
Sampling Period=1.000000 seconds
Estimated Innovations Variance=8.458428e-05
Model Structure Used=Output Error
Estimation algorithm=Gauss-Newton search
Input1blocktype = saturation
Input2blocktype = deadzone
Output block type=deadzone

Model Equations are:

  x_1(t)=saturation(u_1(t))
  x_2(t)=deadzone(u_2(t))

       2   B_i(q)                      Order of B: 3 2
  z(t)=sum  ------x_i(t)+e(t)
      i=1  A_i(q)                      Order of A: 4 3

  y(t)=deadzone(z(t))
_____________________________________________________

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

   1        q^1      q^2      q^3
B=0.0009   0.0023   0.0022   0.0018
SD=0.0005  0.0015   0.0016   0.0025

   1        q^1      q^2      q^3       q^4
A=1.0000   -2.3641  1.8706   -0.5117   0.0125
SD=0   0.5325  1.2737   1.0140   0.2692

delay=1 samples

Poles at 0.0270, 0.6799, 0.7526, 0.9046.

_____________________________________________________

Input #2 to Output #1 Estimated T/F model
+ standard devs:
_____________________________________________________

    1        q^1      q^2
B=0.0015   0.0055   -0.0005
SD=0.0005  0.0011   0.0010

   1        q^1      q^2      q^3
A=1.0000   -2.3965  1.9043   -0.5019
SD=0   0.0350  0.0625  0.0279

delay=1 samples

Poles at 0.7129, 0.7746, 0.9090.
_____________________________________________________

Input Non-linearity Parameters and standard
deviations:
_____________________________________________________

Input block #1 of type saturation has estimates
and standard dev:

upper limit=0.9126,sd=0.0067
lower limit=-0.8115,sd=0.0060

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

```
upper limit=0.6017,sd=0.0037
lower limit=-0.4806,sd=0.0041
```

```
Output Non-linearity Parameters and standard
deviations:
```

```
Output block of type deadzone has estimates
and standard dev:

upper limit=0.0413, sd=0.0011
lower limit=-0.0554, sd=0.0011
```

By way of information, the true noise variance was $\sigma^2 = 10^{-4}$, the true saturation on input 1 was (lower, then upper limit) $[-0.8,0.9]$, the true dead-zone on input 2 was $[-0.5,0.6]$ and the true dead-zone on the output was $[-0.05,0.05]$.

Standard validation tests for this model may be applied via the command

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

which generates three the comparison of data versus model predictions shown in Fig. 9, a whiteness test on the data minus prediction residuals shown in Fig. 10, and a cross-correlation test between residuals and inputs shown in Fig. 11.

Finally, in order to illustrate the employment of the very general non-linear model structure (36), (37) consider the time varying dynamics

$$x_{t+1} = \theta_1 x_t + \theta_2 \frac{x_t}{1+x_t^2} + \theta_3 \cos(1.2t) + w_t, \tag{84}$$

$$y_t = \theta_4 x_t^2 + e_t, \tag{85}$$

where w_t and e_t are uncorrelated zero mean i.i.d. Gaussian processes and the true parameters are

$$\theta^\star = [\theta_1^\star, \theta_2^\star, \theta_3^\star, \theta_4^\star] = [0.5,25,8,0.05]. \tag{86}$$

Using the toolbox to estimate these parameters from data requires first specifying that a general non-linear state space model of state dimension $n_x = 1$ will be employed:

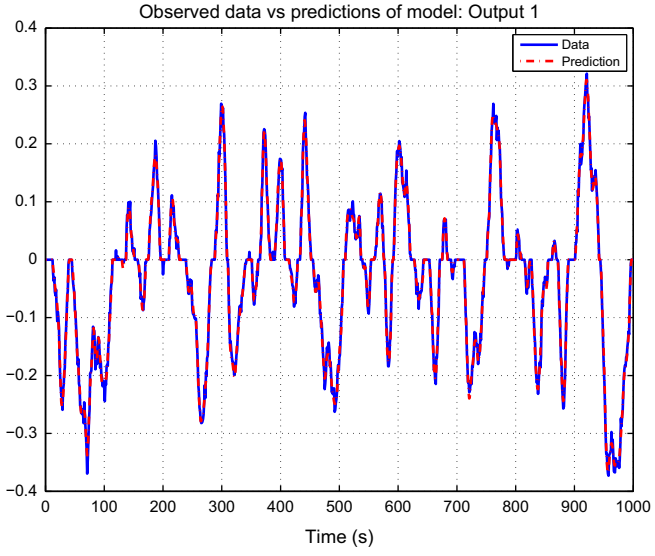

Fig. 9. Observed MISO Hammerstein–Wiener data (solid blue) versus estimated model predictions (dashed red). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

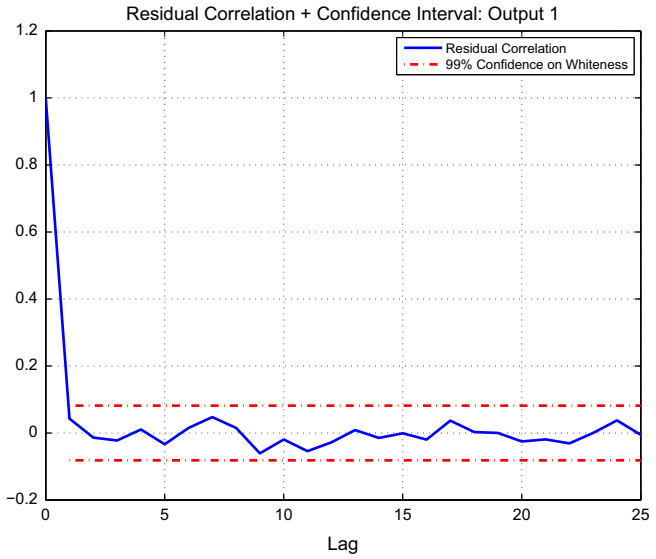

Fig. 10. Whiteness test on residuals of estimated MISO Hammerstein–Wiener model with 99% confidence interval.

```
≫ m.type = 'nlss';
≫ m.nx = 1;
```

and then specifying the details of the model structure (84), (85) via the specification of three functions

```
≫ m.nlss.init = 'nlsstestinit';
≫ m.nlss.estep = 'nlsstestestep';
≫ m.nlss.mstep = 'nlsstestmstep';
```

The first, specifies initial value for the parameters:

```
function g=nlsstestinit(z,m);
g=m; % Preserve what is input;
%Then add initial estimate values
g.theta=[0.4; 29; 7; 0.0]
```

The second, which is too lengthy to reproduce here (but is included with the toolbox distribution) uses the particle smoothing techniques described in Schön et al. (2011) to compute smoothed state estimates `g.xs` based on the model parametrized by `g.theta`. The third and final function

```
function g=nlsstestmstep(z,m,opt)
g=m; % Preserve what is input
% Then maximise wrt current likelihood function estimate
[theta,cost,g]=argmin(z,'LL',m.theta,opt,m);
% Return results
g.theta=theta; g.cost=cost;
```

then uses this smoothed state estimate to approximate the log-likelihood (LL) function via the function `LL` and compute an updated parameter estimate via the `argmin` function included in the toolbox.

The resulting evolution of estimate versus EM-algorithm iteration number is illustrated in Fig. 12.

9. Graphical user interface

The paper has concentrated on the command line functionality of the toolbox. This has highlighted that there are very many options available regarding data formats, model structure

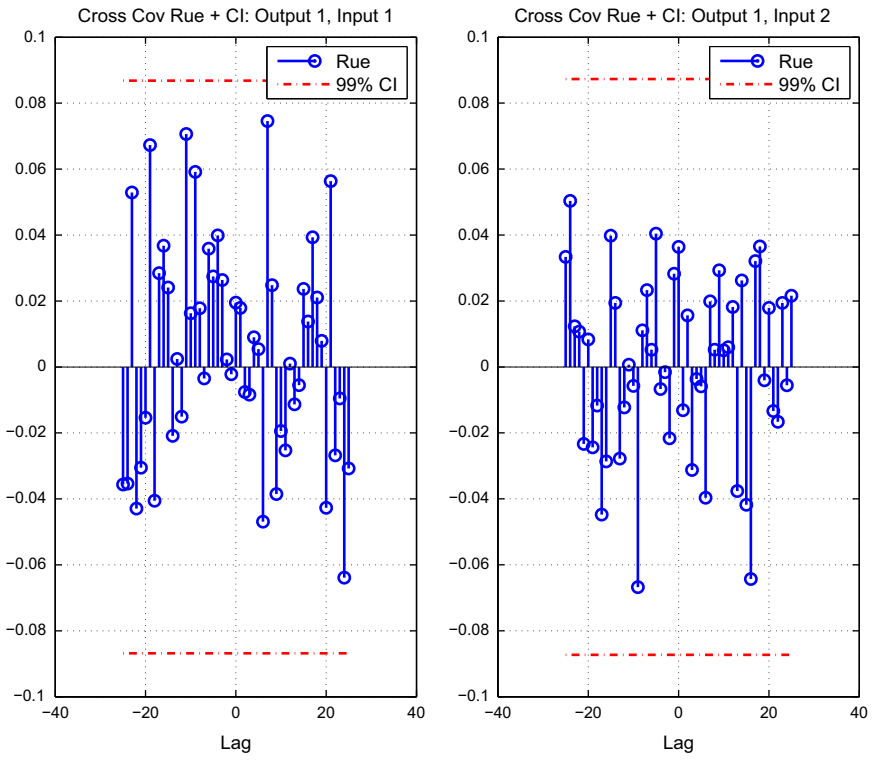

Fig. 11. Cross-correlation estimated MISO Hammerstein–Wiener model residuals and observed inputs together with 99% confidence bounds.

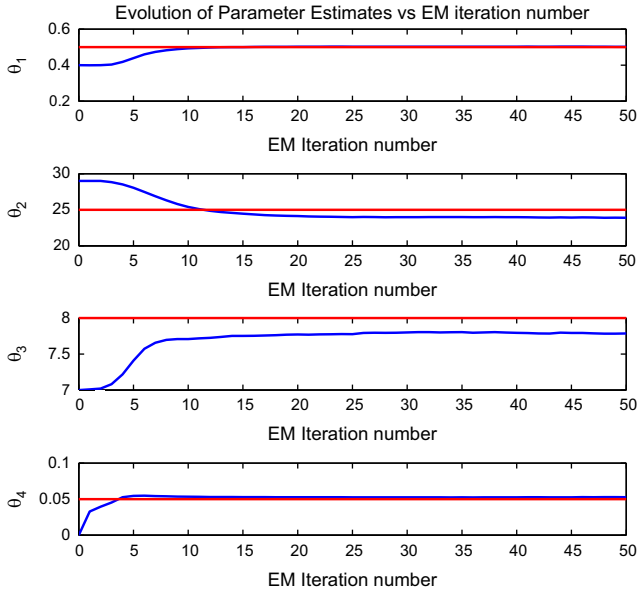

Fig. 12. EM algorithm parameter estimate evolution for general non-linear time varying model estimation problem.

specification, and optionally specified arguments determining the algorithm to be used, and its tuning.

In fact, only a selection of the most important possibilities has been described, since there is insufficient space to catalog them all.

In order to address this wide range of capability, a comprehensive graphical user interface (GUI) has also been developed. Associated with this is the further purpose of simplifying the use of the toolbox for non-expert users, and hence broadening its utility. It is also freely available (www.sigpromu.org/staff/idtoolbox).

The features of the command line version of the toolbox are available via this GUI. The language chosen to implement it is Java (Arnold & Gosling, 1998). This is primarily in the interests of it running consistently and reliably on all platforms for which Matlab is available.

The GUI is designed to be used as a series of four panes. The first, labelled "Data Collection" is used for importing data, and implementing any necessary pre-processing, such as trend removal or filtering. The second, labelled "Model Structure" is used for specifying whatever model structures the user wishes to employ. The third, labelled "Estimate Parameters" is used to specify the estimation method to be used, the algorithm to be deployed to implement the desired method, and any parameters that tune the algorithm.

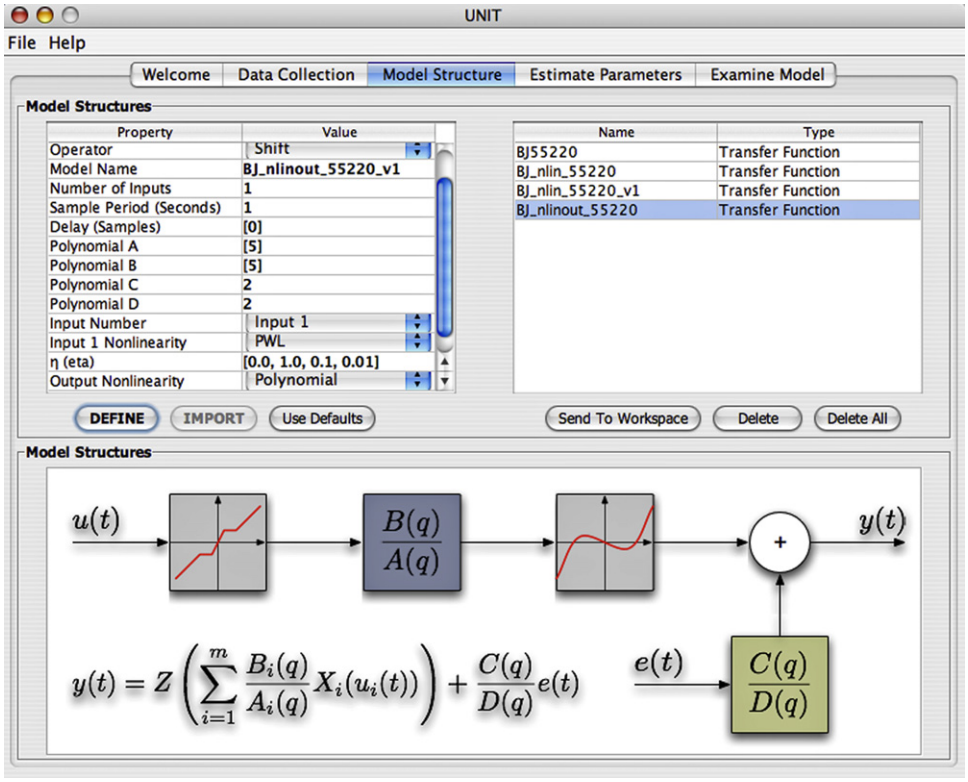

Fig. 13. Illustration of the GUI. This shows the pane used to define model structures. The specification of a Hammerstein–Wiener structures is shown.

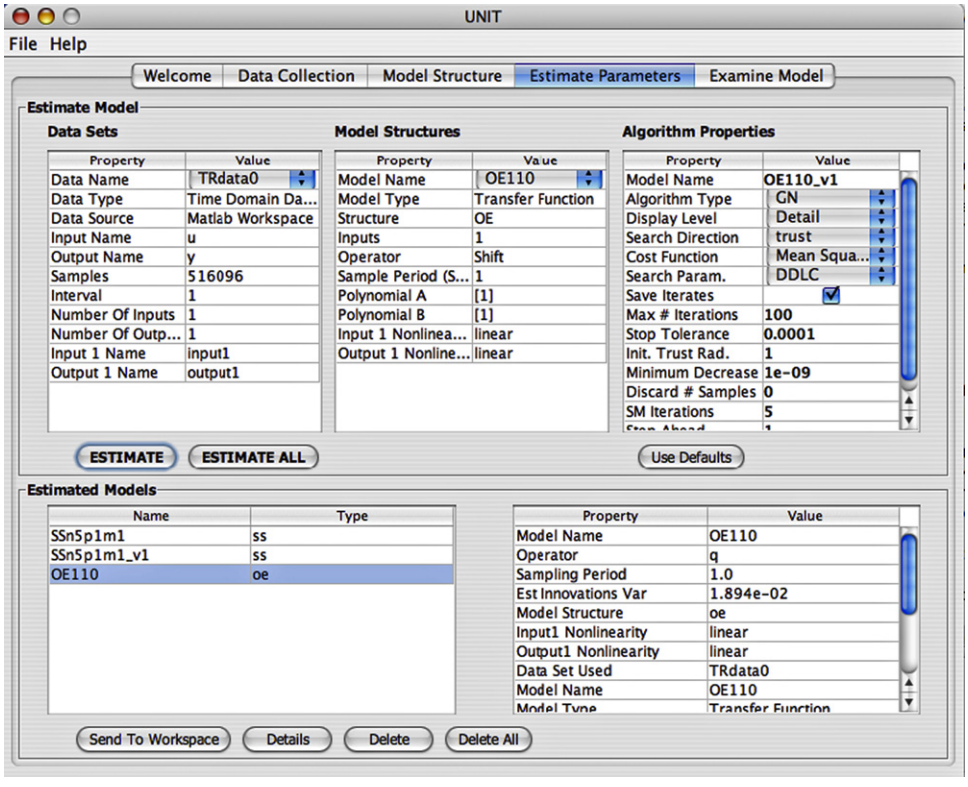

Fig. 14. Illustration of the GUI. This shows the pane used to perform the estimation by defining the estimation method used, and the algorithm to implement it, together with selection of a model structure and data set.

The final pane, labelled "Examine Model" is used for evaluating estimated models by plotting their frequency responses, examining model validation results, plotting error residuals, etc.

Two example panes are illustrated in Figs. 13 and 14. In Fig. 13, the model structure specification pane is shown, which illustrates an example of defining a Hammerstein–Wiener non-linear

structure. All the available options for specifying the model are available in the top left part of the pane using drop-down lists and dialogs. A graphic clarifying the specified structure occupies the bottom half of the pane. The top right contains a list of defined model structures which are then available for use in the subsequent estimation stage.

Fig. 14 illustrates the pane where defined model structures may be estimated. The top left window is used for selection of the data set to be used. The top middle window allows selection of the model structure to be employed. The top right window allows specification of the estimation method and the algorithms used to implement it together with all possible settings that tune the algorithm. The lower left and right windows display, respectively, a list of estimated models, together with overview information about any that are selected.

Other features of the GUI include pop-up "tool-tips" in which hovering the mouse over a model structure or data set displays an overview summary of its properties, and a tutorial mode (selected from the help tab) which guides the user step by step through the estimation process.

10. Conclusion

This paper has detailed the model structures, identification methods, and numerical algorithms employed in a freely available system identification toolbox developed to aid objective evaluation of methods and algorithms.

In order to convey the structure and design philosophy of the toolbox, the essentials of how it is used have been presented. Full usage details are provided via a suite of examples in the `testvec` directory of the toolbox, which can also be accessed by typing `demo_unit` at the command line.

Acknowledgement

The authors would like to sincerely thank Dr. Andrew Fleming for his generous assistance in the building and commissioning of the experimental apparatus used in this paper.

References

⟨www.sigpromu.org/staff/idtoolbox⟩.

Alberts, T., DuBois, T., & Pota, H. (1995). Experimental verification of transfer functions for a slewing piezoelectric laminate beam. *Control Engineering Practice*, *3*, 163–170.

Arnold, K., & Gosling, J. (1998). *The Java programming language* (2nd ed.). Addison–Wesley ISBN 0-201-31006-6.

Bai, E.-W. (1998). An optimal two stage identification algorithm for Hammerstein–Wiener nonlinear systems. *Automatica*, *34*, 333–338.

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

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Dennis, J. E., & Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice Hall.

Favoreel, W., De Moor, B., & Van Overschee, P. (1999). Subspace identification of bilinear systems subject to white inputs. *IEEE Transactions on Automatic Control*, *44*, 1157–1165.

Garnier, H., Gilson, M., & Laurain, V. (2012). Developments for the CONTSID toolbox. In *Proceedings of the 16th IFAC symposium on system identification*, Brussels, Belgium, ISBN: 978-3-902823-06-9.

Gibson, S., & Ninness, B. (2005). Robust maximum-likelihood estimation of multivariable dynamic systems. *Automatica*, *41*, 1667–1682.

Gibson, S., Wills, A., & Ninness, B. (2005). Maximum-likelihood parameter estimation of bilinear systems. *IEEE Transactions on Automatic Control*, *50*, 1581–1596.

Golub, G., & Loan, C. V. (1989). *Matrix computations*. Johns Hopkins University Press.

Guzmán, J., Rivera, D., Dormido, S., & Berenguel, M. (2012). An interactive software tool for system identification. *Advances in Engineering Software*, *45*, 115–123.

Kollár, I., Pintelon, R., & Schoukens, J. (2006). Frequency domain system identification toolbox for Matlab: Characterizing nonlinear errors of linear models. In *Proceedings of the 14th IFAC symposium on system identification* (pp. 726–731), Newcastle, Australia.

Larimore, W. (1990). Canonical variate analysis in identification, filtering and adaptive control. In *Proceedings of the 29th IEEE conference on decision and control* (pp. 596–604), Hawaii.

Larimore, W. E. (2000). The adaptx software for automated multivariable system identification. In *Proceedings of the 12th IFAC symposium on system identification*, Santa Barbara, USA.

Ljung, L. (1999). *System identification: Theory for the user* (2nd ed.). New Jersey: Prentice-Hall, Inc.

Ljung, L. (2012). *MATLAB system identification toolbox users guide*, version 8. The Mathworks.

Ljung, L., & Wills, A. (2010). Issues in sampling and estimating continuous-time models with stochastic disturbances. *Automatica*, *46*, 925–931.

Mathworks (2012). *MATLAB users guide*. The Mathworks.

McKelvey, T., Akçay, H., & Ljung, L. (1996). Subspace-based multivariable system identification from frequency response data. *IEEE Transactions on Automatic Control*, *41*, 960–979.

Meirovitch, L. (1996). *Elements of vibration analysis* (2nd edn.). McGraw Hill.

Middleton, R., & Goodwin, G. (1990). *Digital estimation and control: A unified approach*. New Jersey: Prentice-Hall, Inc.

Ninness, B., & Henriksen, S. (2010). Bayesian system identification via Markov chain Monte Carlo techniques. *Automatica*, *46*, 40–51.

Pota, H., & Alberts, T. (1995). Multivariable transfer functions for a slewing piezoelectric laminate beam. *ASMW Journal of Dynamic Systems, Measurement, and Control*, *117*, 352–359.

Schön, T., Wills, A., & Ninness, B. (2011). System identification of nonlinear state-space models. *Automatica*, *37*, 39–49.

Söderström, T., & Stoica, P. (1989). *System identification, systems and control engineering*. Prentice Hall.

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

van Overschee, P., & Moor, B. D. (1996). *Subspace identification for linear systems*. Kluwer Academic Publishers.

Wills, A., & Ninness, B. (2008). On gradient-based search for multivariable system estimates. *IEEE Transactions on Automatic Control*, *53*, 298–306.

Wills, A., Ninness, B., & Gibson, S. (2009). Maximum likelihood estimation of state space models from frequency domain data. *IEEE Transactions on Automatic Control*, *54*, 19–33.

Young, P. (2011). *Recursive estimation and time series analysis: An introduction for the student and practitioner*. Berlin: Springer-Verlag.

Young, P. C., Taylor, & C. J. (2012). Recent developments in the CAPTAIN toolbox for Matlab. In *Proceedings of the 16th IFAC symposium on system identification*, Brussels, Belgium, ISBN: 978–3–902823–06–9.