Jurnal Teknologi, 36(D) Jun 2002: 1–22
© Universiti Teknologi Malaysia

A GRAPHICAL USER INTERFACE APPLICATION FOR CONTINUOUS-TIME IDENTIFICATION OF DYNAMICAL SYSTEM

MOHD. FUA'AD RAHMAT[1], ROSLI OMAR[2], HISHAMUDDIN JAMALUDDIN[3]

Abstract. This paper introduces a Graphical User Interface (GUI) application in system identification and parameter estimation of dynamic systems using Generalized Poisson Moment Functionals (GPMF) method based on Instrumental Variable (IV) algorithm. The GUI based on MATLAB consists of data preprocessing, parameter estimation, model validation and model simulation. A step-by-step instruction on how to use the GUI employed in the study is also presented. A validation process had been implemented using cross validation process. Simulation process based on the estimated mathematical model was performed to analyze the dynamic behaviors of the model. From the simulation results and analysis, it could be concluded that the model obtained using this GUI is reliable.

Keywords: GUI, system identification, parameter estimation, GPMF, IV

Abstrak. Kertas kerja ini memperkenalkan teknik grafik pengantaramukaan pengguna di dalam pengenalpastian sistem dan penganggaran parameter sistem menggunakan kaedah GPMF yang berdasarkan algorithma IV. GUI yang berdasarkan kepada perisian MATLAB ini mengandungi pra pemproses data, anggaran parameter, penentusah model dan simulasi model. Prosedur penggunaan GUI juga ditunjukkan dalam kajian ini. Proses penentusah model adalah menggunakan proses penentusahan silang. Proses simulasi yang berdasarkan kepada anggaran model matematik telah dilaksanakan untuk menganalisis ciri-ciri dinamik model. Dari keputusan simulasi dan analisis didapati model matematik yang diperolehi adalah boleh dipercayai.

Kata kunci: GUI, pengenalpastian sistem, anggaran parameter, GPMF, IV

1.0 INTRODUCTION

The classic definition of a user interface is the hardware and software through which human and computer interact. Over the years, this concept has evolved, incorporating more aspects of the human-computer experience and applying them to varied contexts. The definition has also changed to reflect the technological advances, moving from a command-line oriented interface to one that includes graphic features.

[1] Department of Control & Instrumentation, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310, Skudai, Johor, Malaysia. e-mail: fuaad@suria.fke.utm.my
[2] Post Graduate Student, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310, Skudai, Johor, Malaysia. e-mail: to_lideq@hotmail.com
[3] Faculty of Mechanical Engineering, Universiti Teknologi Malaysia, 81310, Skudai, Johor, Malaysia. e-mail: hishamj@fkm.utm.my.

Graphical interfaces have grown in popularity, resulting in widely accepted conventions for the use of the components of these interfaces[1]. A graphical user interface (GUI) is a user interface made up of graphical objects, such as menus, buttons, lists, and fields.

The goal of this paper is to introduce a GUI that can be used to estimate the parameters of mathematical models of dynamic system using the Generalized Poisson Moment Functionals (GPMF) method based on the IV algorithm from experimental input output data.

MATLAB was used as the programming language to develop a GUI that allows the user to perform model identification, validate the models and finally simulate the models. The GUI that contains specific commands will mani pulate the GPMF based on IV algorithm in order to estimate the parameters when the appropriate input and output relationshi p of a dynamic system is applied. Consequently, the computer calculates the estimated parameters.

This paper is divided into four sections and organized as follows. Section 1 discusses a brief introduction to GUI and its objective. Section 2 presents a literature on continuous-time system identification. A theoretical framework of the Poisson Moment Functional(PMF) algorithm and instrumental variable estimator necessary for the study are also discussed in this section. Section 3 explains the software development that includes a step-by-step explanation on how to use the GUI. Finally, section 4 concludes with a summary on the contributions of the paper.

2.0 DYNAMIC SYSTEM MODEL IDENTIFICATION

2.1 Continuous-time verses Discrete

All dynamic system models may be divided into two main types based on whether they characterize a continuous-time or discrete-time process [2]. In the field of system identification and parameter estimation, the attention received by the discrete-time models is so enormous that the continuous-time counterpart was completely overshadowed until a few years ago [3]. The rapid development of the parameter estimation procedures for discrete-time models has tended obscure parallel developments in continuous-time model estimation, despite the fact that much conceptual control systems analysis is still carried out in terms of continuous-time differential equation [4]. However, the situation is gradually changing and the relevance of continuous-time models has been established in a number of situations [5]. Much of the current literature on identification is now concerned with the identification of continuous-time models [3]. Furthermore, for many applications, continuous time models are more appealing to engineers than discrete-time models because they are closely related to the underlying physical systems, whereas discrete-time model is considered to be defined at a sequence of time-instants related to measurement[6]. Moreover, most models encountered in the physical world are continuous and the development of automatic control owes a great deal to the concepts evolved originally in continuous-time domain. Another motivation for continuous time modelling is that as the cost of computation becomes cheaper, data acquisition equi pment in today's industry provides near continuous time measurement. Fast sampled data can be more naturally dealt with using a continuous time model than a discrete model.

In addition, the process of continuous-time system modelling is always developed based on physical laws such as Newton's Second Law or Ohm's law, which provides *a priori* knowledge. A typical example of *a priori* knowledge is the choice of the order of the transfer function that is obtained from differential equation after Laplace Transform was applied. This *a priori* knowledge was necessary to be defined before the parameters can be estimated to ensure the accurate model that represents the real model to be identified. Besides, *a priori* information can typically best be incorporated into a continuous time model. In the resulting differential equation obtained from physical law, the coefficients are closely related to the corresponding physical parameters in the system.

2.2 Identification Procedure

Continuous-time model identification may be seen to consist of two stages [2]

(i) *The primary stage* in which the system equation of the parameter estimation is derived from the dynamical model of the system to be identified is applied at this stage.
(ii) *The secondary stage* in which the continuous-time parameters are estimated using IV method within the framework of a parameter estimation method.

The primary stage arises out of the derivative measurement problem [2]. The GPMF transformation that is derived in this first stage converts the process differential equation into an algebraic form. This method helps in reducing the calculus of continuous-time dynamical systems into appropriate algebra required for parameter estimation.

The secondary stage is independent of the original model form and depends only on the system of IV algorithm arising out of it. These two stages are shown in Figure 1.

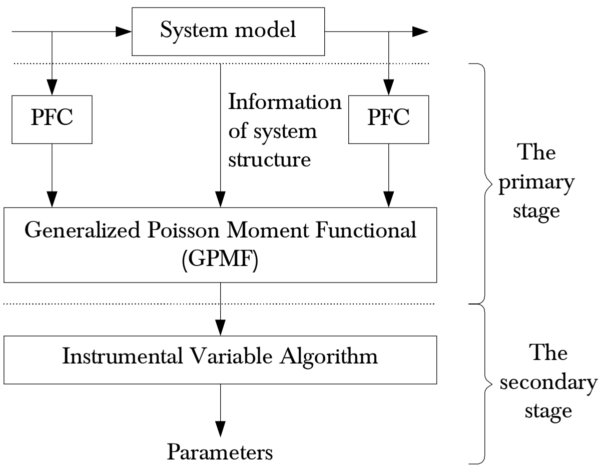

System model
PFC
Information of system structure
PFC
Generalized Poisson Moment Functional (GPMF)
Instrumental Variable Algorithm
Parameters
The primary stage
The secondary stage

Figure 1 Parameter estimation in continuous-time models

PFC denotes the Poisson Filter Chain operation. PFC is shown in Figure 2.

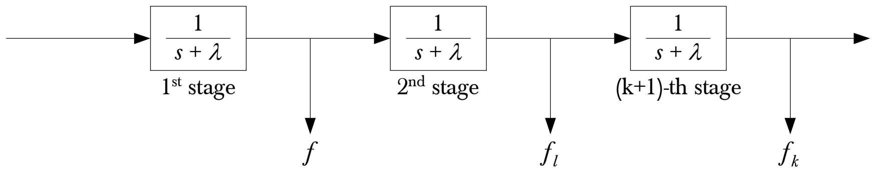


Figure 2 Piosson filter chain (PFC)

Each element of the PFC has a transfer function of the form $\frac{1}{s+\lambda}$. The PFC operation is such that while retaining the parameters of the continuous model in their actual original form, it facilitates the generation of the appropriate measurements for the parameter estimation equation. The PMF of the signals are generated by passing them through PFC. In GPMF, the generalized PFC (GPFC) is used. Hence each element of the GPFC has a transfer function $\frac{\beta}{s+\lambda}$ where β is a positive real number.

2.3 Poisson Moment Functionals (PMF)

A signal $f(t)$, $t \in (0, t_0)$, is treated as a distribution or generalized function, and expanded about a time instant t_0 as shown in the following expression.

$$f(t) = \sum_{k=0}^{\infty} M_k \{f(t)\} \quad \exp[-\lambda(t-t_0)]\delta^{(k)}(t-t_0) \tag{1}$$

where $\delta^{(k)}(t - t_0)$ is the k-th generalized time derivative of an impulse distribution occurring at $t = t_0$ and

$$M_k\{f(t)\} \equiv f_k^0 = \int_0^{t_0} f(t)\, p_k(t_0 - t)\, dt \tag{2}$$

with

$$p_k(t_0) \equiv p_k^0 = \frac{t_0^k}{k!} \exp(-\lambda t_0) \tag{3}$$

and λ is a positive real number. p_k^0 is called the k-th order Poisson pulse function (PPF) at t_0 and f_k^0 is termed as the k-th PMF of $f(t)$ about $t = t_0$. f_k^0 may be viewed as the output due to an input $f(t)$, at $t = t_0$, of the $(k+1)$-th stage of a cascaded filter with identical stages, also called the PFC, each element of which has a transfer function $1/(s + \lambda)$ as indicated in Figure 2.

The PMF transformation about $t = t_0$ of $y(t)$ viz. $M_k\{y(t)\}$ is defined as

$$M_k\left\{\frac{dy(t)}{dt}\right\} \equiv \int_0^{t_0} \frac{(t_0 - t)}{k!} e^{-\lambda(t_0 - t)} \frac{dy(t)}{dt} dt \tag{4}$$

Integrating by parts, the right hand side of Equation (4) yields

$$\frac{(t_0 - t)^k}{k!} e^{-\lambda(t_0 - t)}\, y(t)\Big|_0^{t_0} - \int_0^{t_0}\left[-\frac{(t_0 - t)^{k-1}}{(k-1)!} e^{-\lambda(t_0 - t)} + \lambda \frac{(t_0 - t)^k}{k!} e^{-\lambda(t_0 - t)}\right] y(t)\, dt, \tag{5}$$

implying that

$$M_k\left\{\frac{dy(t)}{dt}\right\}_{t_0} \equiv M_k\left\{y^{(j=1)}(t)\right\}_{t_0} = y_{k-1}^0 - \lambda y_k^0 - p_k^0 y^{(0)}(0), \tag{6}$$

wherein the subscript '0' signifies the transformation about $t = t_0$, the subscript k denotes the order of the PMF and the subscript (j) denotes the order of the derivative term; $y^{(0)}(0) = y(t = 0)$. p_k^0 is the value of the PPF of order k at $t = 0$. By analysis similar to the above, it can be shown that

$$M_k\left\{y^{(2)}(t)\right\}_{t_0} = y_{k-2}^0 - 2\lambda_{k-1}^0 + \lambda^2 y_k^0 - \left(p_{k-1}^0 - \lambda p_k^0\right) y^{(0)}(0) - p_k^0 y^{(1)}(0) \tag{7}$$

The derivatives of PMF's are expressed as linear combinations of those of the original function. These derivatives can be measured as the outputs of a PFC, excited by the original function.

Consider a differential equation

$$a\frac{dy(t)}{dt} + y(t) = bu(t) \tag{8}$$

The k-th PMF transformation of Equation (8) about t_0

$$aM_k\left\{\frac{dy(t)}{dt}\right\}_{t_0} + M_k\{y(t)\}_{t_0} = bM_k\{u(t)\}_{t_0} \tag{9}$$

Inserting Equation (6) into Equation (9) yields

$$a\left[y^0_{k-1} - \lambda y^0_k - p^0_k y^{(0)}(0)\right] + \left[y^0_k\right] = b\left[u^0_k\right] \tag{10a}$$

or

$$\left[y^0_{k-1} - \lambda y^0_k - p^0_k y^{(0)}(0) - u^0_k\right]\begin{bmatrix} a \\ b \end{bmatrix} = y^0_k \tag{10b}$$

In Equation (10a) the initial condition $y^{(0)}(0)$ is known from $y(t)\big|_{t=0}$.
Let $k = 1$, $t_0 = t_1$ and t_2 $(j = 1,2)$, then

$$\begin{bmatrix} y^1_0 - \lambda y^1_1 - p^1_1 y^{(0)}(0) & -u^1_1 \\ y^2_0 - \lambda y^2_1 - p^2_1 y^{(0)}(0) & -u^2_1 \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -y^1_1 \\ -y^2_1 \end{bmatrix} \tag{11}$$

2.4 Identification of Lumped Linear Time-Invariant SISO System

Consider the n-th order linear differential equation of lumped linear time-invariant SISO system with constant coefficients

$$\sum_{i=0}^{n} a_i \frac{d^i y(t)}{dt^i} = \sum_{i=0}^{m} b_i \frac{d^i u(t)}{dt^i}, \qquad m < n \tag{12a}$$

which can be used to model an n-th order lumped linear time-variant SISO system with the input $u(t)$ and output $y(t)$. In most physical systems, the value of m does not exceed $n - 1$. Without loss of generality, in Equation (12a), fix $a_0 = 1$ or $a_n = 1$. However, for the purpose of a general treatment these conditions are not imposed at the beginning. Let $m = n$ for symmetry in the mathematical expressions and Equation (12a) was rewritten in the form

$$\alpha^T\ y(t) = \beta^{\mathrm{T}}\ u(t), \tag{12b}$$

where

$$\alpha = \left[a_0 \quad a_1 \quad \ldots \quad a_n\right]^T \tag{12c}$$

$$\beta = [b_0 \quad b_1 \quad \dots \quad b_n]^T \tag{12d}$$

$$\mathbf{y}(t) = \left[y(t) \frac{dy}{dt} \cdots \frac{d^n y}{dt^n} \right]^T , \tag{12e}$$

and

$$\mathbf{u}(t) = \left[u(t) \frac{du}{dt} \cdots \frac{d^n u}{dt^n} \right]^T , \tag{12f}$$

The PFC operation is now applied on $y(t)$ and $u(t)$ giving certain measures of these signals and their otherwise unmeasurable time derivatives. Accordingly, if $\bar{m}_{y,k(t)}$ and $\bar{m}_{u,k(t)}$ are the k-th measures of the vectors $\mathbf{y}(t)$ and $\mathbf{u}(t)$ respectively, Equation (12b) may be written as

$$\bar{m}^T_{y,k(t)} \ \alpha = \bar{m}^T_{u,k(t)} \ \beta \tag{13}$$

2.5 The PMF Algorithm

In this section, the general PMF algorithm is developed. Taking the n-th order PMF transformation about $t = t_k$ for the PFC operation, it can be shown that

$$\bar{m}_{y,k(t)} = y_k \wedge^T - \Phi_k^T \wedge^T S_y^T \tag{14a}$$

and

$$\bar{m}^T_{u,k(t)} = y_k^T \wedge^T - \Phi_k^T \wedge^T S_u^T \tag{14b}$$

where

$$y_k = [y_0(t_k) \quad \dots \quad y_n(t_k)]^T \tag{14c}$$

$$u_k = [u_0(t_k) \quad \dots \quad u_n(t_k)]^T \tag{14d}$$

$y_i(t_k)$ is the i-th order PMF of $y(t)$ about $t = t_k$, and $u_i(t_k) = i$-th order PMF of $u(t)$ about $t = t_k$.

In other words, $y_i(t_k)$ for this example is the output of a PFC at its $(i + 1)$-th stage at $t = t_k$. Next Λ (is an $(n + 1)$ ($(n + 1)$ matrix whose ij – th element is defined as in Equations (14e) and (14f)

$$\Lambda_{ij} = \begin{cases} 0, & i+j\langle n+2 \\ (-1)^{n+j-i}\begin{bmatrix} i-1 \\ i+j-(n+2) \end{bmatrix}, & \lambda^{i=j-(n+2)}, i+j \geq n+2 \end{cases} \tag{14f}$$

Δ is an $(n + 1) \times (n +1)$ shift matrix defined as

$$\Delta = \left[\begin{array}{c|c} O^T & O \\ \hline I_n & O \end{array}\right] \tag{14g}$$

$$y^{(i)}(0) = \frac{d^i y}{dt^i}\Big|_{t=0} \tag{14h}$$

Similarly

$$S_u = \sum_{i=0}^{n-1} u^{(i)}(O)\,\Delta^{i+1} \tag{14i}$$

Φ_k is a vector of Poisson pulse functions (PPF) defined as

$$\Phi_k = \left[p_0(t_k)\,p_i(t_k)\cdots p_n(t_k)\right]^T \tag{14j}$$

$$p_i(t_k) = (t_k)^i \exp(-\lambda t_k)/i! \tag{14k}$$

Inserting Equation (14a) and (14b) into Equation (13) and rearranging, yields

$$y_k^T \Lambda^T \alpha - u_k^T \Lambda^T \beta + \Phi_k^T \Lambda^T \left[S_u^T \beta - S_y^T \alpha\right] = O \tag{15}$$

At this point, any of the terminal elements in (can be fixed as unity. Suppose if $a_{0\,=\,1}$ and $b_n = 0$ as in the case in most physical systems, then

$$\alpha = \left[1 \quad | \quad a^T\right]^T \tag{16a}$$

and

$$\beta = \left[b^T \quad | \quad 0\right]^T \tag{16b}$$

Under these condition, Equation (15) can be written as

$$y_k^T\ \Lambda^T \left[1 / a^T\right]^T - u_k^T\ \Lambda^T \left[b^T\ | O\right]^T + \Phi_k^T\ \Lambda^T \left[S_u^T \beta - S_y^T \alpha\right] = O \tag{17a}$$

or

$$\left[y_k^T\ \overline{\Lambda}^T \ | -u_k^T\ \Lambda^{*T}\ | \Phi_k^T\ \Lambda^*\right] \begin{bmatrix} a \\ b \\ f \end{bmatrix} = -y_n(t_k) \tag{17b}$$

Equation (17b) may be concisely written in standard form as

$$m_k^T p + v = y_k^* \tag{18a}$$

where

$$y_k^* = y_n(t_k) \tag{18b}$$

$$p = \left[a^T \mid b^T \mid f^T\right]^T \tag{18c}$$

$$m_k^T = \left[-y_k^T \Lambda^{-T} \mid + y_k^T \Lambda^{-T} \mid -\Phi_k^T \Lambda^*\right] \tag{18d}$$

v in Equation (18a) is a disturbance vector. $\bar{\Lambda}$ is matrix Λ with its first column removed and Λ^* is Λ with its last column removed and f is an n-vector of initial condition terms which is obtained by dropping out the last element of the vector $\left[S_u^T \beta - S_y^T \alpha\right]$. Equation (18a) has $3n$ unknowns to be determined. These include the n initial condition terms in f along with the $2n$ system parameters contained in a and b. The system of parameter estimation equations can now be obtained by taking PMF's about t_k, $k = 1, 2, \ldots, N$, with Equation (18a) in the standard form

$$Mp + v = y^* \tag{19}$$

where

$$M = \begin{bmatrix} m_l^T \\ \vdots \\ m_N^T \end{bmatrix} \tag{20}$$

and

$$y^* = [y^*_1 \cdots y^*_N]^T \tag{21}$$

When $N = 3n$, the number of unknowns, is

$$p = M^1 \; y^* \tag{22}$$

and if $N > 3n$ the IV estimate of p is

$$\hat{p} = \left(Z^T M\right)^{-1} Z^T \; y^* \tag{23}$$

where Z are the instrumental variables.

There are two conditions to impose instrumental variables Z in order to make the estimator $\hat{p}$ consistent [6]. First, the instrumental variables Z should be uncorrelated with the disturbances so that . Second, matrix must be invertible.

Hence in addition, to the two previously imposed conditions, it is necessary that $Z^T M$ be large so that $\hat{p}$ provides an efficient estimate. In other word, the instrumental variables should be chosen so that they are simultaneously uncorrelated with and highly correlated with M.

3.0 SOFTWARE DEVELOPMENT

3.1 The Software Basic Features

The GUI features all the basic components in system identification such as input-output data loading, data preprocessing, parameter estimation, model validation, and model simulation. The features can be summarized as [7]

(i) Estimate the procedures of single-input single-output (SISO) model
(ii) Loading the input-output data from various sources.
(iii) Select data preprocessing method such as remove trends, remove means and filter the data.
(iv) Select the data range for estimation.
(v) Display the result in the form of continuous-time transfer function or show the values of zeros and poles.
(vi) View the model output, frequency response, root locus, prediction error, model residual and the comparison between the estimated respond with the measured data.
(vii) Entering other set of measured data for the purpose of model validation.
(viii) Save the value of estimated parameters.
(ix) Enable the user to give his comments about the estimated model for future reference.

Figure 3 shows the flow chart representing the whole procedure of estimating the parameters of any dynamic systems.

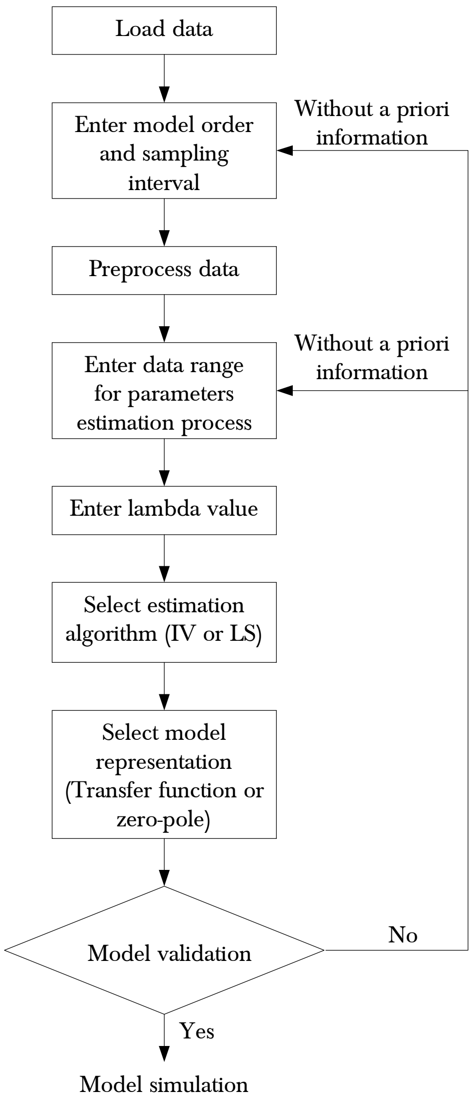
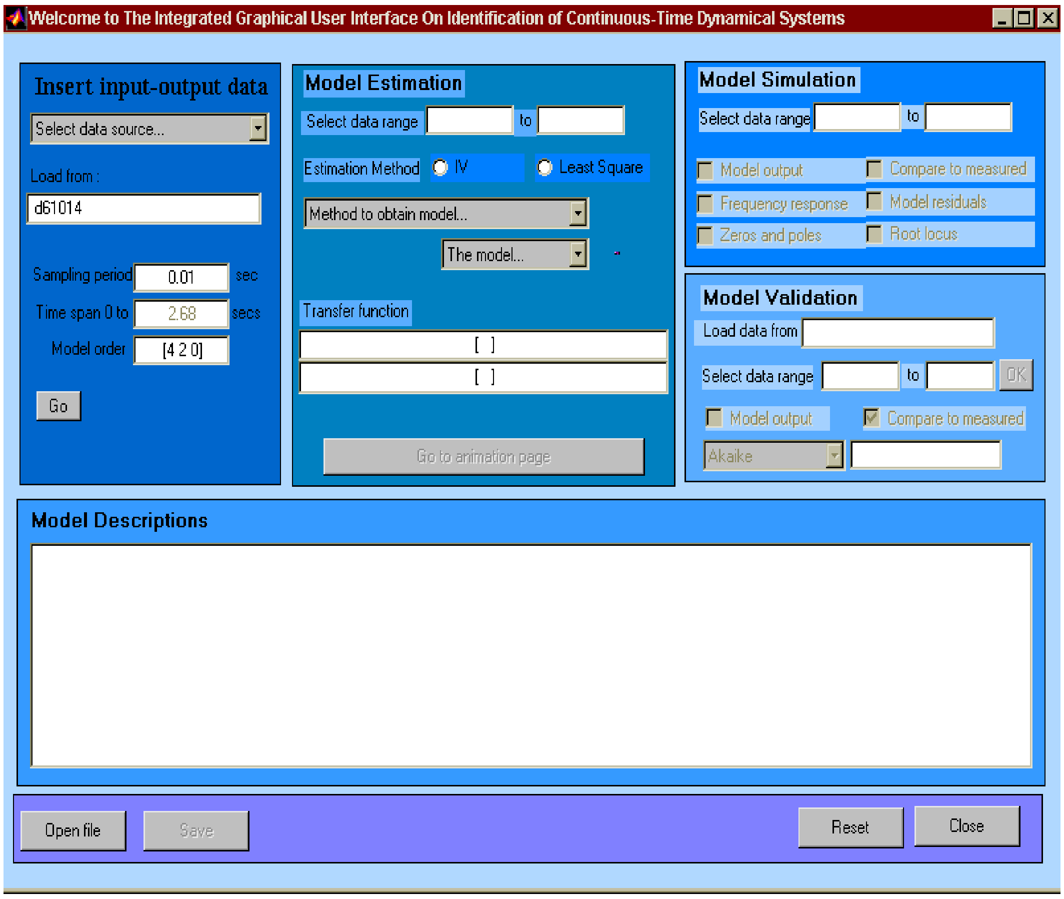

Load data
Enter model order and sampling interval
Without a priori information
Preprocess data
Enter data range for parameters estimation process
Without a priori information
Enter lambda value
Select estimation algorithm (IV or LS)
Select model representation (Transfer function or zero-pole)
Model validation
No
Yes
Model simulation

Figure 3 Flow chart for the wohoe process of estimating model parameters

3.2 The Completed GUI

The goal of the GUI is to estimate the parameters of a model to any SISO system, validate the estimated model and validate the model. The final GUI window in MATLAB environment is shown in Figure 4. Altogether, the GUI consists of 14 editable text boxes, 7 push buttons, and 2 pop-up menus. The implementation and the ideas of the GUI will be described in section 3.3.



Load from : d61014

Sampling period 0.01 sec

Time span 0 to 2.68 secs

Model order [4 2 0]

[ ]

[ ]

Figure 4 The developed GUI used to estimate parameters from input-output data.

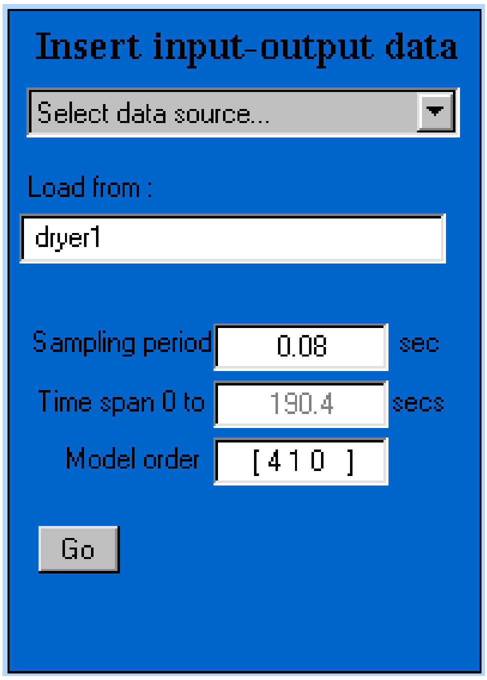
3.3 GUI Implementation

In this section, a step by step explanation on the use of GUI employed in the study is demonstrated.

(i) *Enter Input-output Data, Sampling Interval, Model Order and Pre-process the Data*

System identification is about data and models and creating models from data. Before the creation of the model can be executed, the input-output data, sampling period and model order need to be defined. At the upper left corner of the GUI window, user is able to choose the sources of the data. The data may be loaded from file saved under the MAT file extension of the MATLAB software, insert the data manually or using simulated data stored in ASCII file. Here, only the first method is discussed i.e., loading the data from a file since measured data saved in MAT file is used.

Figure 5 shows part of the GUI that is located at the upper left hand side of the GUI window. It allows the user to choose the data source, insert a sampling period and model order in the space provided.



dryer1

Sampling period 0.08 sec

Time span 0 to 190.4 secs

Model order [4 1 0]

Figure 5 Part of the GUI that allows the user to choose and enter data that are needed to estimate the parameters.

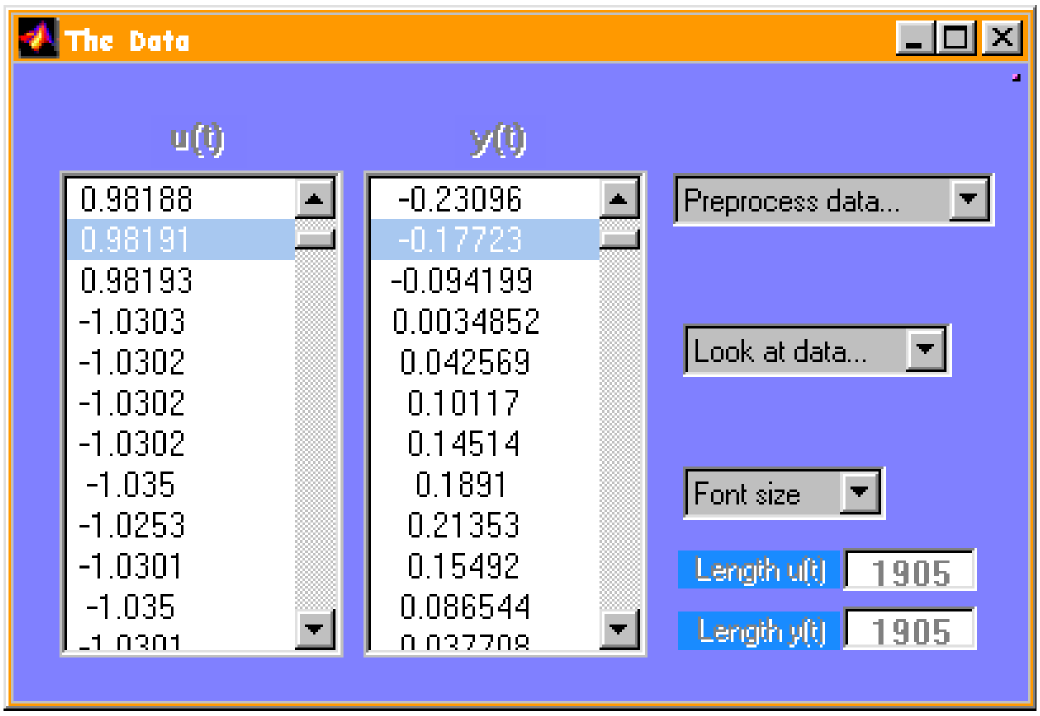
For this example, we will use a hairdryer data that can be loaded from MATLAB environment by typing "*load dryer*". Notice that there are 2 sets of input output data. The first set, set I i.e., *ze* is used as the estimation data set whereas the second data set, set II i.e., *zv* is used as the validation data set. The user is reminded to give a label to the input-output data for each set. For data set I, the first column is output data and labeled as y and second column is input data set and labeled as *u*. To label this, the user can type in the MATLAB workspace such as "$y = ze(:,1)$" and "$u = ze(:,2)$". Then type "$z = [y\ u]$". For data set II, which is used for the purpose of model validation, the output (first column) is labeled as yv and input (second column) is labeled as *uv*. To label this, go to the MATLAB workspace and type "$y = zv(:,1)$" and "$u = zv(:,2)$". Then save it as a filename. In this example, the required filename is "*dryer1*". Hence type in the MATLAB workspace "*save dryer1*". Then, put this file name in *Load from* section as shown in Figure 5.

Notice, because of the data are loaded from file, there is no need to insert the time span since the length of the data is known. It will be calculated in the background and will be displayed in the space called *Time span 0 to*. Note that the model order must have three elements. The first, second and third elements represent the order of model denominator, numerator and delay respectively. This model order is obtained from physical modeling such as Newton's second law or Ohm's Law etc.

In summary, when dealing with this *Insert Input-Output Data* part, the users are required to:

(a) Select the data source (load data from a file)
(b) Insert file name where the data is located (without the extension)
(c) Insert sampling period (in second)
(d) Insert the model order obtained from physical modeling

After all the information required were entered, the *Go* button was pressed. Then the data will be displayed as shown in Figure 6.



u(t)	y(t)
0.98188	-0.23096
0.98191	-0.17723
0.98193	-0.094199
-1.0303	0.0034852
-1.0302	0.042569
-1.0302	0.10117
-1.0302	0.14514
-1.035	0.1891
-1.0253	0.21353
-1.0301	0.15492
-1.035	0.086544

Length u(t) 1905
Length y(t) 1905

Figure 6 The input-output data to be estimated

The input and output data can also be represented graphically by clicking the *Look at data...* pop-up-menu and choose *Plot data*. Examples are shown in Figure 7.

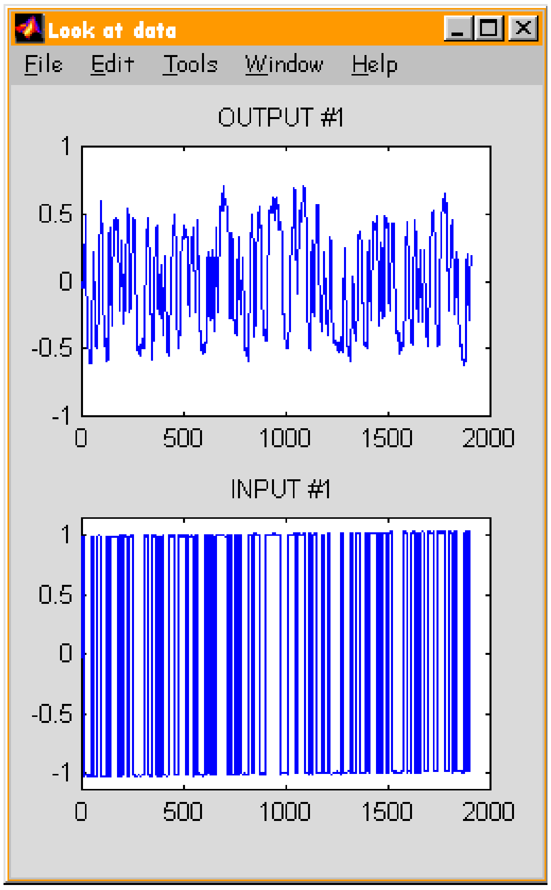
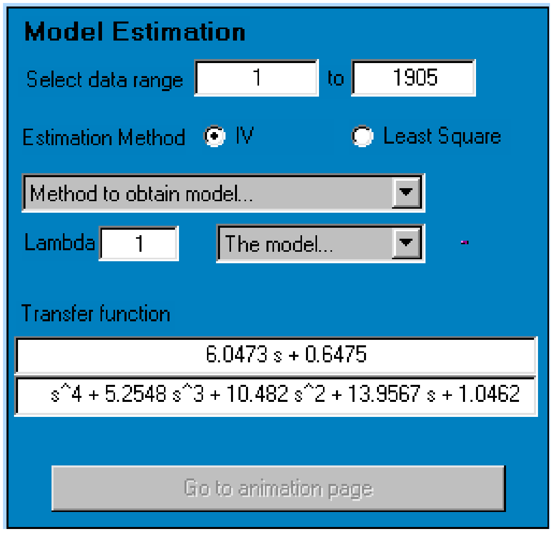


Figure 7 The graphs that represents the input-output data

The data then can be processed according to the user's request. The user may remove the mean and trend or filter the data by clicking the popup-menu denoted by *Preprocessing dat.* It may involves repairing the data in terms of replacing missing or obviously erroneous data. It typically also involves data polishing by removing undesired disturbance. This is accomplished primarily by low-pass or high-pass prefiltering and/or subtracting trends (detrend) from the data.

(ii) *Parameter Estimation*

Estimating model from the data is the central activity in this GUI. The models were estimated using the data set as displayed in Figure 6. The *Model Estimation* part as shown in Figure 8 uses the data to estimate the models. This GUI treats only a parametric method where a specific model structure is assumed and the parameters in this structure are estimated using measured data.



Model Estimation

Select data range: 1 to 1905

Estimation Method: [✓ IV] Least Square

Method to obtain model...

Lambda: 1 | The model...

Transfer function

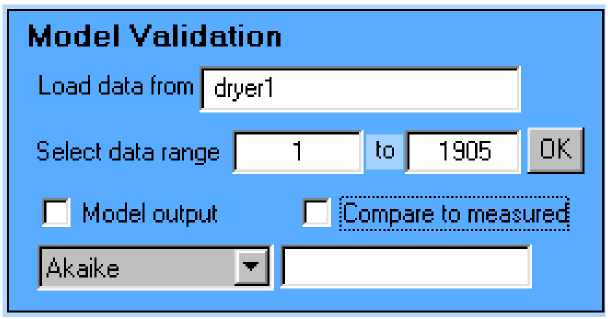
$$\frac{6.0473\,s + 0.6475}{s^4 + 5.2548\,s^3 + 10.482\,s^2 + 13.9567\,s + 1.0462}$$

Figure 8 The *Model Estimation* part to estimate the model parameters using GPMF based on IV

When dealing with the *Model Estimation* part, firstly the user must key in the data range to be executed. The user then can choose the estimator to be used, either least square (LS) or IV. IV estimator is preferred over LS because it gives more accurate estimation and reduced the bias of the parameter estimated. Value for lambda must be inserted manually. There are two ways to represent the estimated model, one by the model transfer function and by the zeros and poles representation.

(iii) *Model Validation*

Up to this stage, the mathematical model of a system has been developed. The identified model needs to be examined to verify whether it fulfills the model requirements according to subjective and objective criteria of good model approximation. Figure 9 shows an option called *Model Validation*.



Model Validation

Load data from: dryer1

Select data range: 1 to 1905

Model output | Compare to measured

Akaike

Figure 9 *Model Validation* option used to validate the model

It is good and common practice in identification to evaluate an estimated model's properties using a "fresh" data set, that is, one that was not used for the estimation. The validation data is different from the estimation data. It is called simulation. The most versatile validation tool is probably simulation [8].

In this part of the GUI, the user should enter the file name that contains the validation data set to be compared with the simulated data from the model. Then, enter the data range and push the *OK* button. It will display the input data graphically as shown in Figure 10.

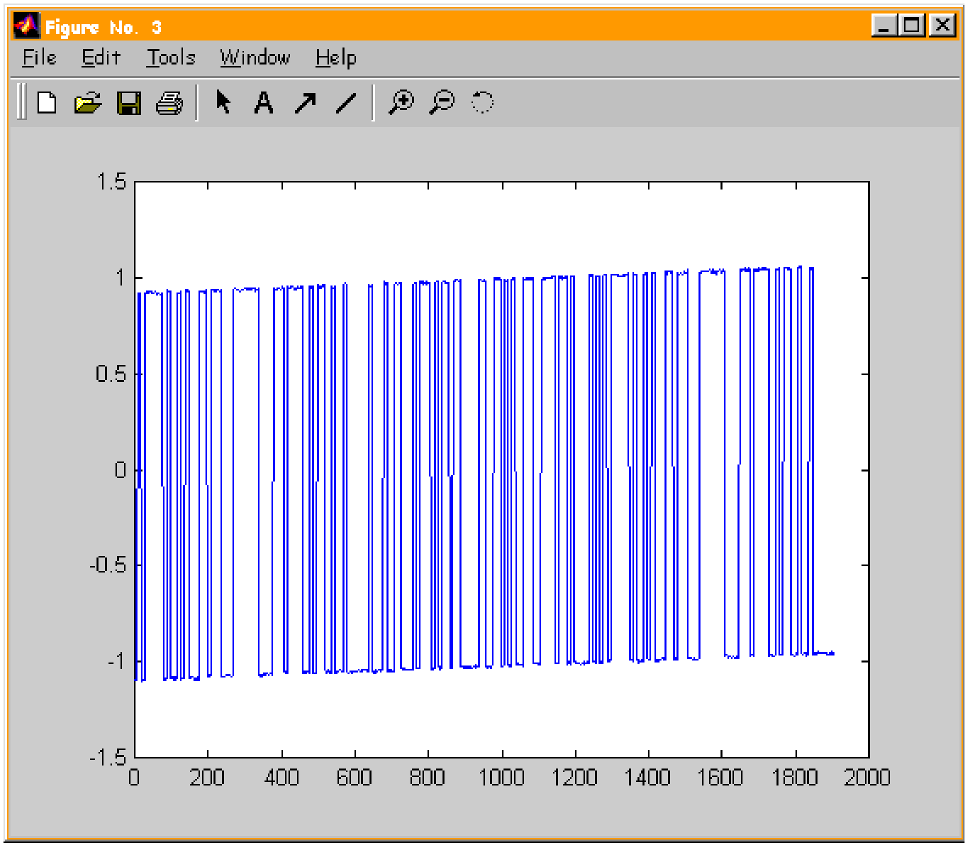


Figure 10 The graph of input data for model validation process

A very good way of obtaining insight into the quality of a model is to simulate it from a different data set, and compare the simulated output with the measured one. This test is obtained by checking *Compare to the measured*. Then the data in the validation data set will be used for comparison. The comparison between the output of validation data and simulated data is displayed graphically as shown in Figure 11.

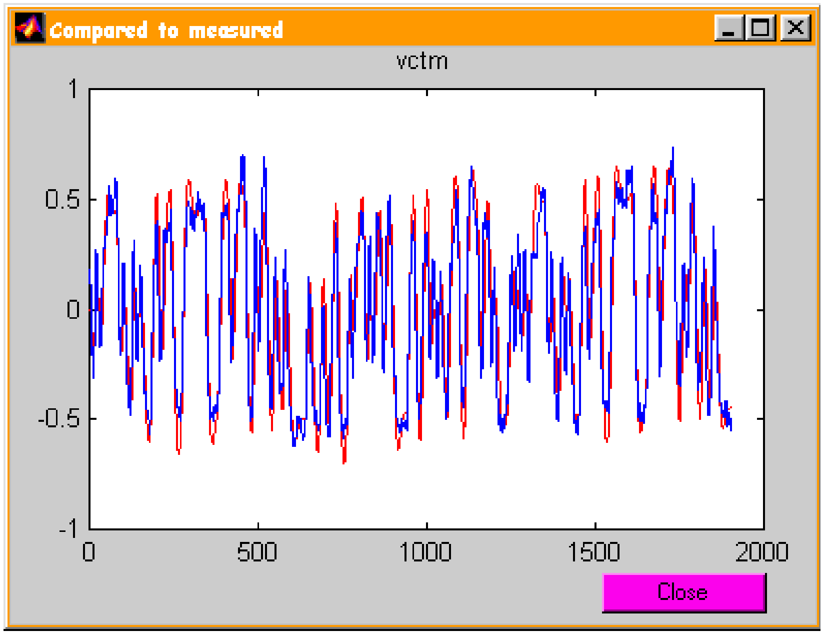
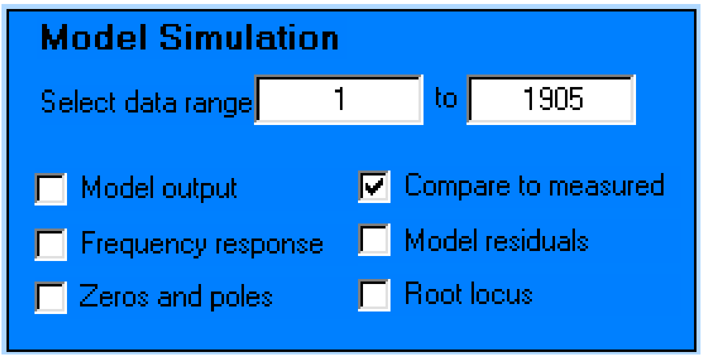


Figure 11 The window that shows the comparison between the measured data and simulated data.

(iv) *Model Simulation*

To check whether a model is capable of reproducing the observed output when driven by the actual input, a simulation can be performed. Simulation to obtain simulated output was done by checking the checkbox called *Model Output* contained in one part of the GUI named *Model Simulation* as shown in Figure 12. The output excited by the actual input is shown in Figure 13.


Model Simulation

Select data range 1 to 1905

☐ Model output
[✓ Compare to measured]
☐ Frequency response
☐ Model residuals
☐ Zeros and poles
☐ Root locus

Figure 12 *Model Simulation* part used to view the model properties

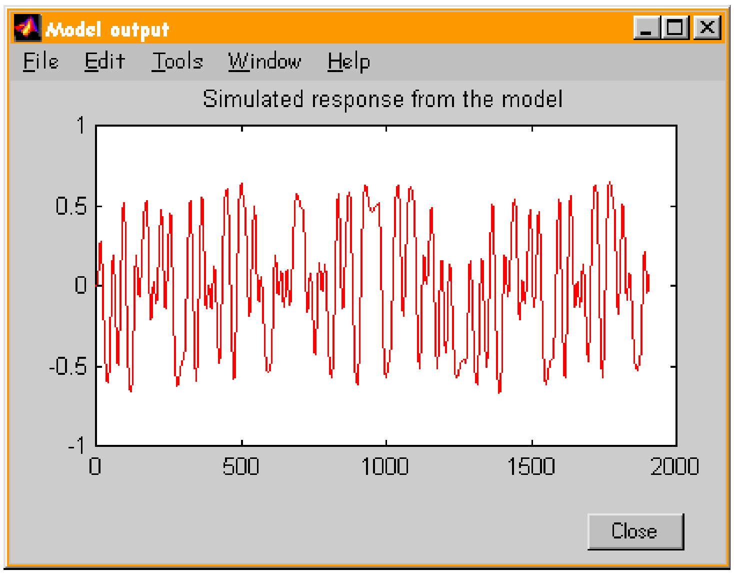


Figure 13 The output simulated from the estimated model

The true system and the derived model are then fed with the same input signals, whereupon the measured outputs are compared to the predicted output of the model. This is done when the *Compared to measured* checkbox is checked. The compared outputs is shown in Figure 14.

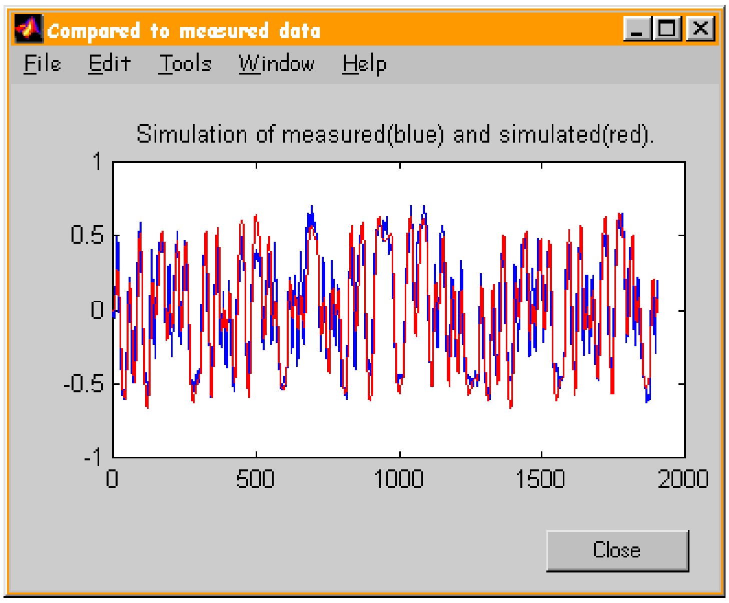


Figure 14 Comparison between simulated and measured data

Finally, the zeros and poles map of the system model can also be plotted by checking the *Zeros and poles* checkbox button as shown in Figure 15.

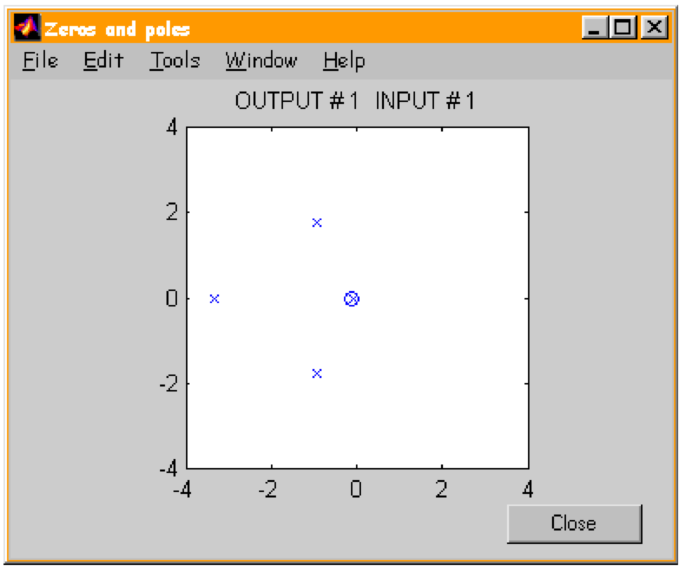
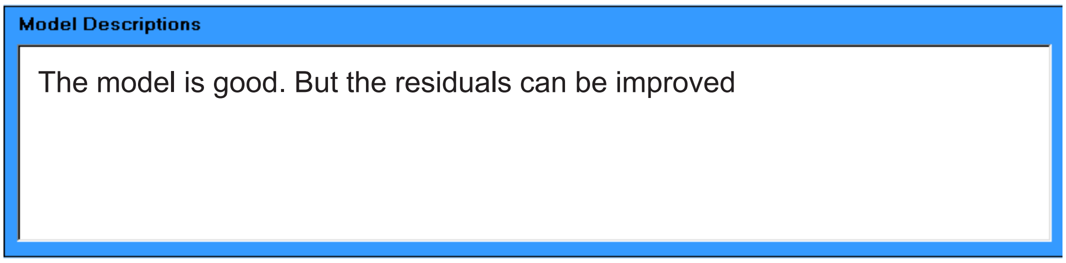


Figure 15 Poles zeros map of the system model

The poles of a system are the roots of the denominator of the transfer function, while the zeros are the roots of the numerator. In particular the poles have a direct influence on the dynamic properties of the system. The *Model Simulation* part makes the analysis of the estimated model much simpler.

(v) *Model Description*

One of the features of this GUI is that it enables the user to give some descriptions of the model obtained. It is important to the user if he wishes to keep the information about the obtained model in a safe way and easy to view format for future reference. The description of the model can be filled in the space called *Model Descriptions*. It is shown in Figure 16.


Model Descriptions

The model is good. But the residuals can be improved

Figure 16 A model properties can be described in this provided space

Then, the descriptions and the estimated models can be saved. The user can reach the file in the future and refer to the comments to get the ideas about the model. The advantage of this feature is the best model can be assessed from all models developed in the study. The user may write down the pros and cons of each model estimated. By doing this, the user can choose the best model after taking some consideration to the pros and cons of all models.

(vi) *Save, Open File and Reset Data*

It is common to most of GUIs to have features that enable the user to save and open files and reset the data. The developed GUI also has these features. After the user finished estimating the model of a system and give some descriptions to the estimated model, he can save his work by clicking the Save button located at the lower left corner of the GUI. He can also refer to the file by opening it if he wishes to know about the estimated model properties in some other time without doing the same modelling process using the same data and model order. This will considerably save his time.

4.0 CONCLUSION

It has been presented that the GUI that used the GPMF method based IV algorithm have successfully identify the appropriate model from input and output data of a system as shown in section 3. Of the existing continuous-time parameter estimation approaches based on an equation error approach which use transfer function models, the Poisson Moment Functionals approach is claimed to be one of the best under noisy conditions. IV estimator was considered rather than the least square because it gives more accurate estimation and reduced the bias of the parameter estimated. Consequently, this developed GUI has achieved the objectives and it can be considered successful. It was shown that this GUI has made the modeling process of a dynamic system easy and quick. The GUI contains six parts; *Enter input-Output data, estimate the parameters, Model Validation, Model Simulation and Model Descriptions*. It has important features that every GUI must have such as user-friendly and easy to use.

REFERENCES

[1] The Mathworks Inc. 1994. *MATLAB: High Performance Numeric Computation And Visualization Software*. Prime Park Way Natick, Massachusetts: The Mathworks, Inc.

[2] Unbehauen, H and G. P. Rao, 1987. *Identification of Continuous Systems*. New York: Elsevier Science Publishers B.V.

[3] Subrahmanyam, A. V. B., 1993. Identification of Continuous-time SISO Systems via Markov Parameter Estimation. *IEE Proc*. 140(1): 1-10.

[4] Young, P. *Parameter Estimation For Continuous-Time Models-A Survey*. Automatica: The Journal of IFAC The International Federation of Automatic Control. Great Britain : Pergamon Press Ltd.

[5] Sinha N. K and G.P. Rao. 1991. *Identification of Continuous-Time Systems: Methodology And Computer Implementation*. Netherlands: Kluwer Academic Publishers.

[6] Johansson, R., 1993. *System Modeling And Identification*. New Jersey: Prentice-Hall, Inc.

[7] Rosli Omar and Mohd Fua’ad Rahmat. 2001. Identification of Car Passive Suspension System Transfer Function Parameters. *Student Conference On Research And Development*. Kuala Lumpur, Malaysia.

[8] Lindskog, P.,1996 . *Methods, Algorithms and Tools for System Identification Based on Prior Knowledge*. Department of Electrical Engineering Linkoping University: Ph.D Thesis.