University of Alberta Computer Process Control Group

Subspace Closed-loop Identification

Limited Trial Version

Written by: CPC Control Group, University of Alberta

Version 1.0

Table of Contents

List of Figures

Introduction

The "Subspace Closed-loop Identification" toolbox was developed by the Computer Process Control Group at the University of Alberta to allow closed-loop identification to be performed in MATLAB using a subspace approach.

A "Quick Start" approach to using this algorithm is presented, along with a detailed section containing full explanations and examples for using this algorithm.

System Requirements

In order to run this program properly, the following programmes are required:

1) MATLAB 2006a (MATLAB 7.1) or better. It should be noted that the newest version of MATLAB (MATLAB 2008a) makes the toolbox run slower.
2) The SYSTEM IDENTIFICATION TOOLBOX from MATLAB is required.

Quick Start

For quickly using the toolbox, the following steps should be followed:

1) Unzip the files to the desired location.
2) Start MATLAB, and point the current directory to the location of the unzipped files.
3) At the command prompt, type **"`>>main_CLsysID`"** to start the toolbox. The GUI shown in Figure 1 should appear.

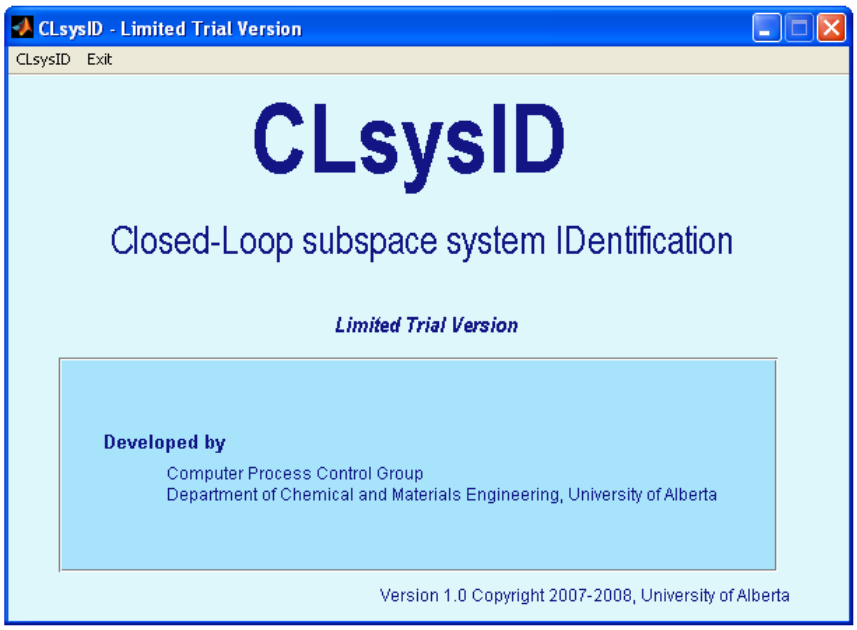

Figure 1: The First GUI that appears

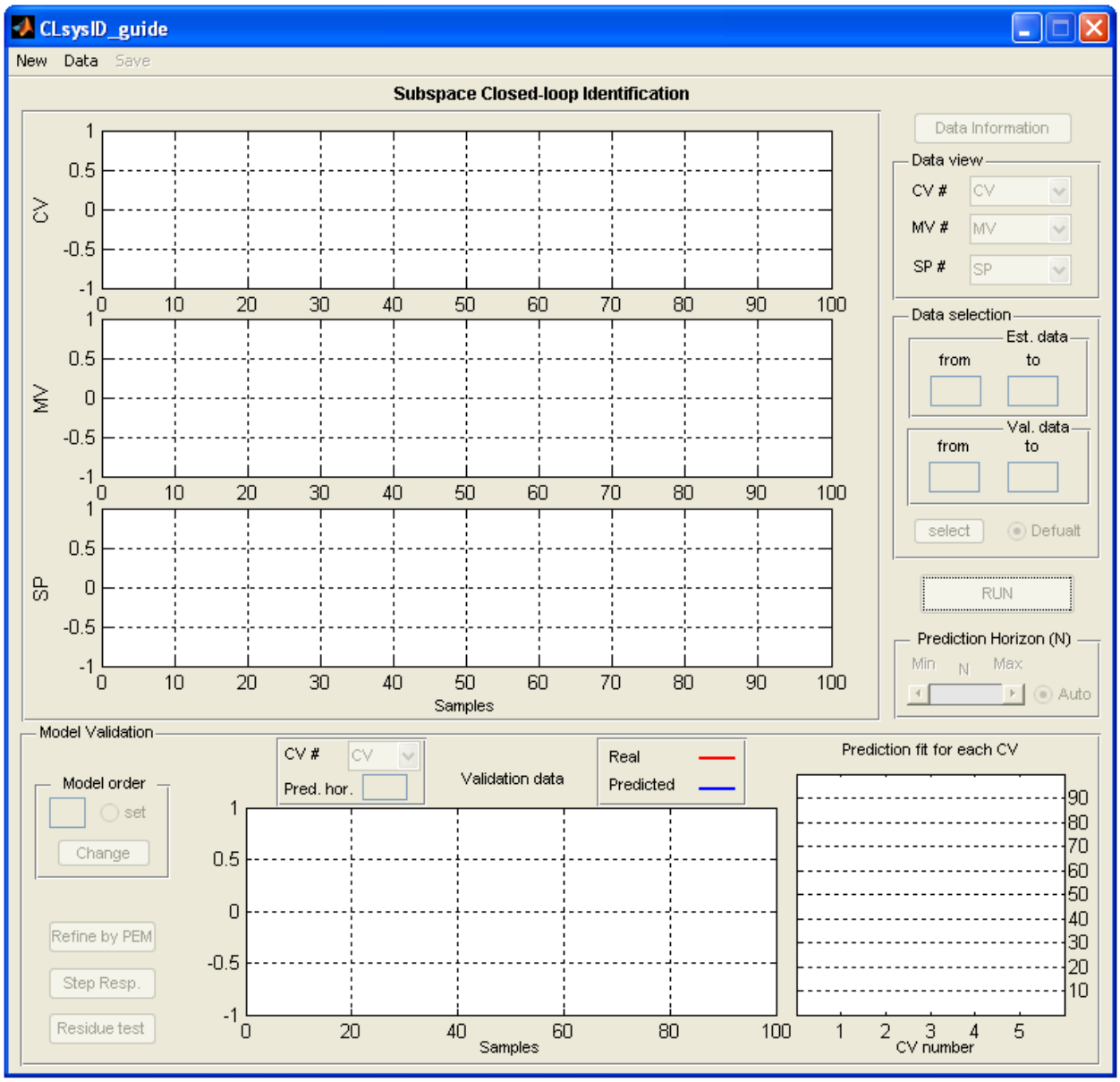

Figure 2: The main GUI for this toolbox

4) Press the **'CLsysID' menu**. A new GUI will appear that is shown in Figure 2.
5) This GUI tool provides closed-loop subspace identification using "**Joint Input-Output Identification**" method.
6) You have to upload a set of closed-loop experiment data in *Compact* format (to be explained shortly) using **'Data'** menu.
7) For the purpose of this quick start, we will use the sample data provided from a MIMO example.

8) When the data is uploaded, the corresponding information about it can be obtained by pressing the **'Data Information'** button.

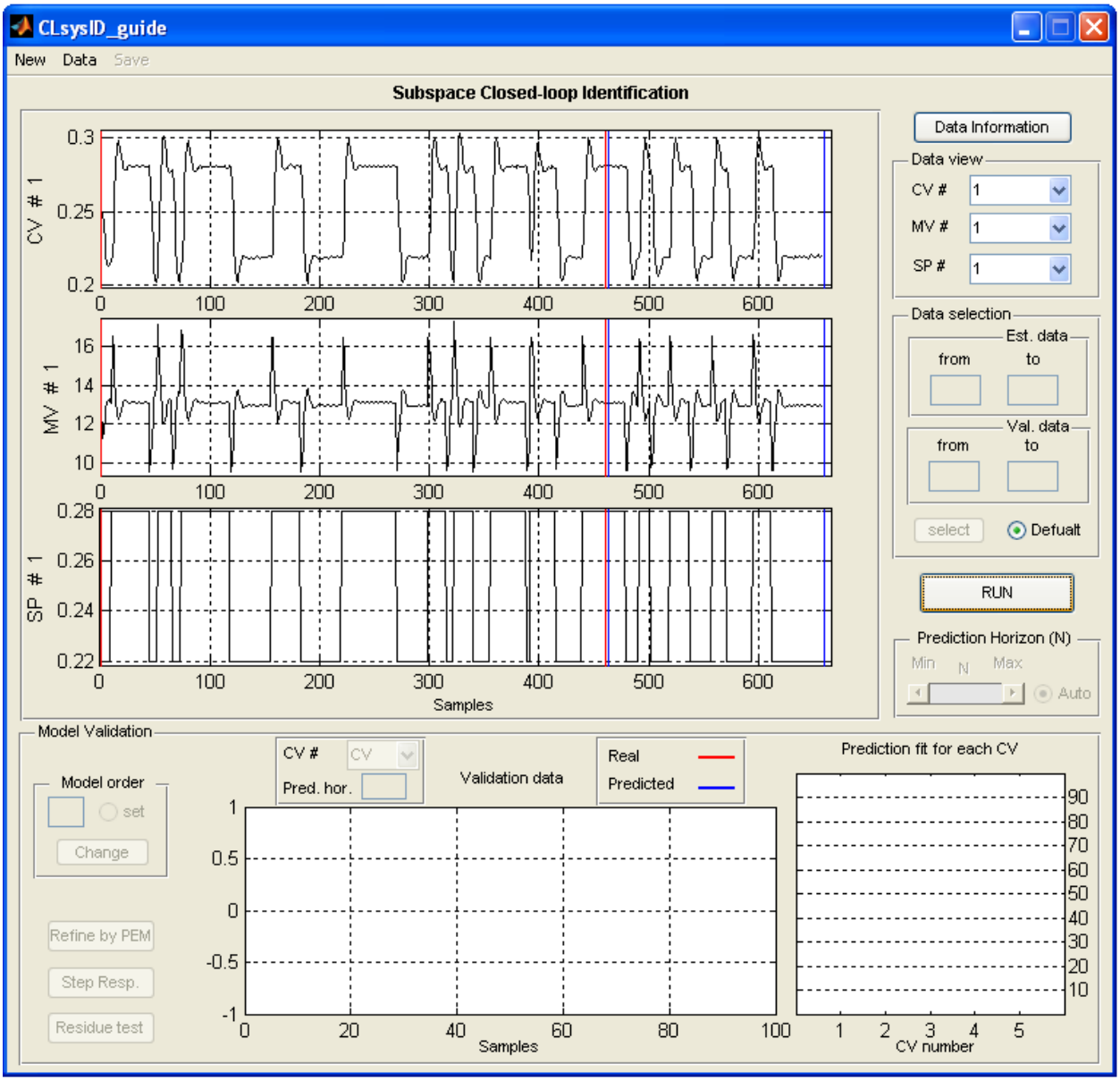

Figure 3: Closed-loop data loaded in

9) Use the **'Run'** button to estimate the process model from closed-loop data. Depending on the type of process dynamics, number of inputs and outputs and number of data points, it takes a few seconds to a few minutes to perform the calculation.

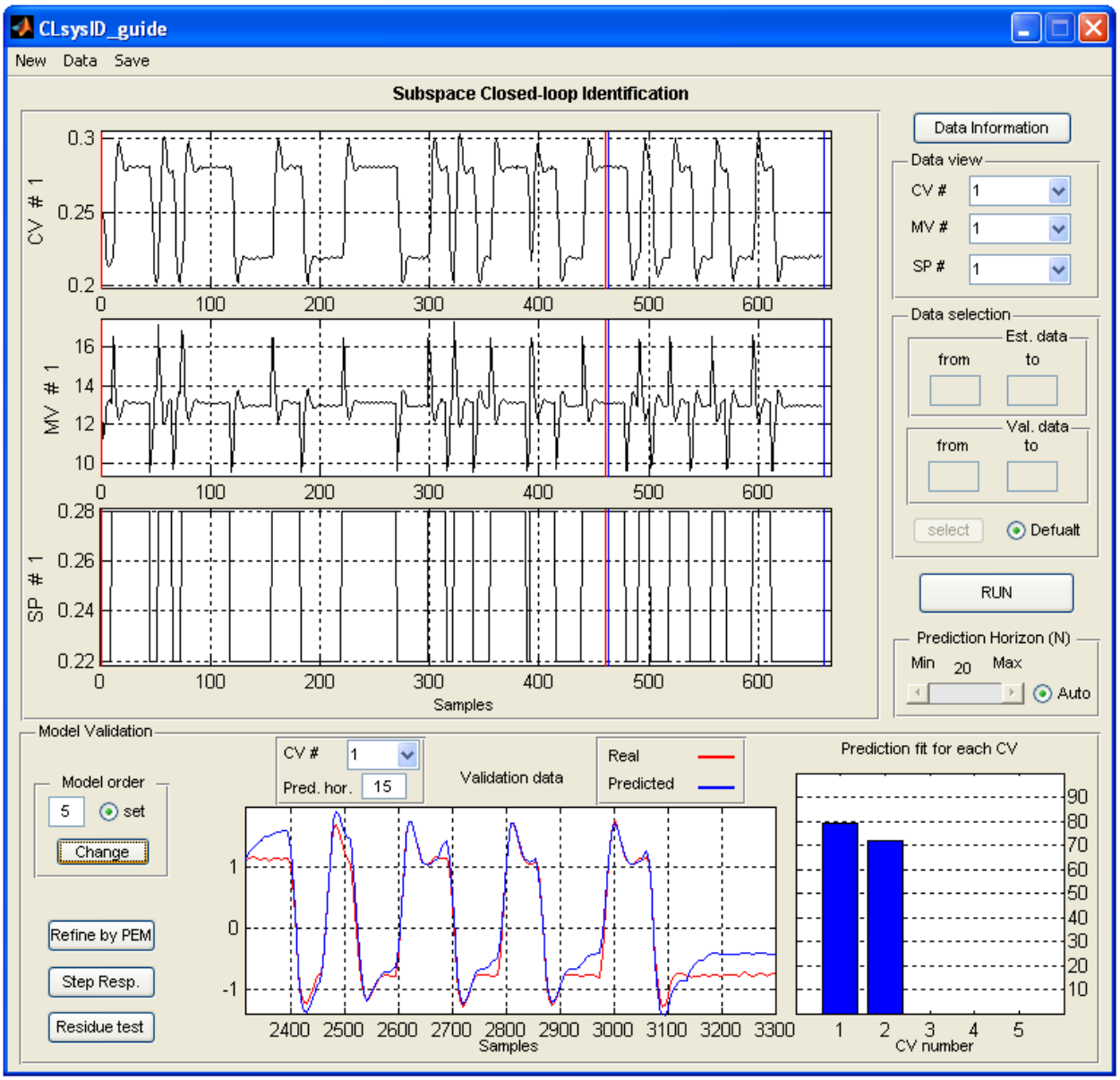

Figure 4: The state space model estimated and the prediction fit is shown

10) The results of model validation are provided in the **'Validation'** panel including the prediction fit and residual test results.
11) The estimated model can be refined using the Prediction Error Method. You can use **'Refine by PEM'** button in the **'Validation'** panel.
12) Direct step response estimation and continuous-time transfer function models between each input and output can also be obtained using this tool. **'Step Resp.'** button in the **'Validation'** panel can be used for this purpose.

13) To start a new problem, press the **'New'** Menu.

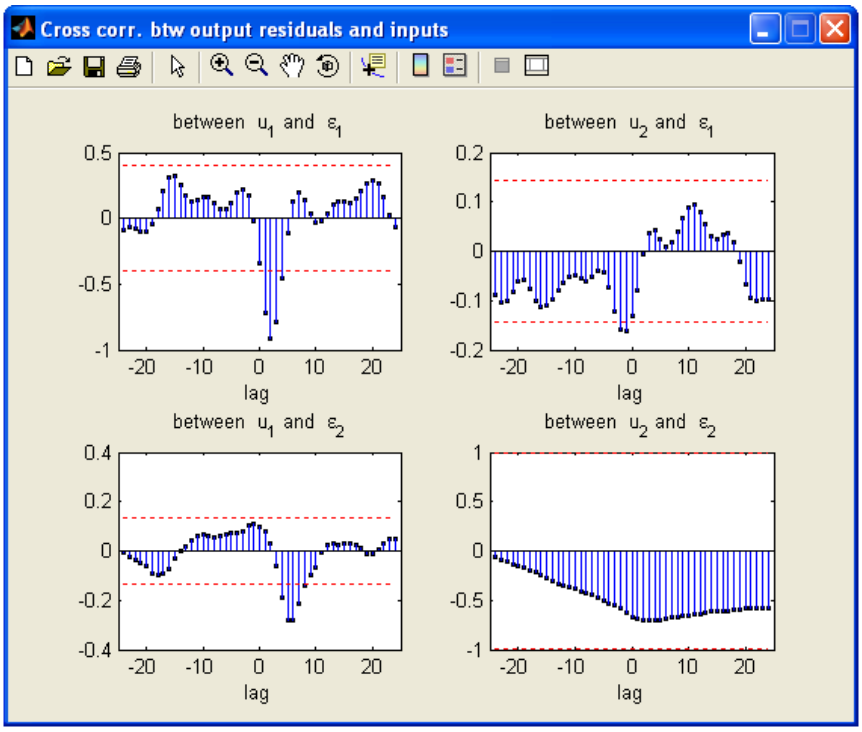

Figure 5: Residue test results

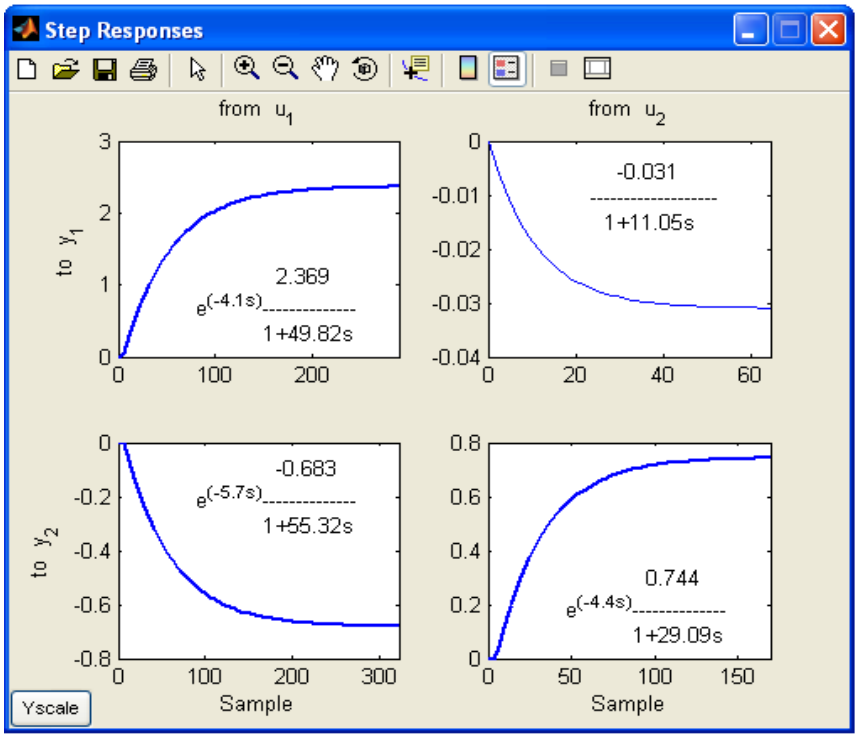

Figure 6: Step response model results

Detailed Instructions

Theory

Subspace identification methods provide an alternative approach to classic system identification methods like the prediction error method [1] and instrument variable method [2]. Subspace methods use efficient computational algorithms such as QR-factorization and singular value decomposition, which makes them intrinsically robust from a numerical point of view. Various methods of subspace identifications have been developed in the past two decades such as Regression analysis approach, N4SID (Numerical SubSpace State Space IDentification), MOESP (MIMO Output Error State sPace), CVA (Canonical Variate Analysis) [3-9]. Many studies have been devoted to the closed-loop identification as well because of its extensive use in control-relevant identification and controller performance assessment. Most of subspace methods have been also extended for closed-loop identification (see [3,10,11,12,13]). A closed-loop subspace identification method was also provided in by Kadali and Huang in [14] based on joint input-output identification approach. An alternative formulation of this method with guaranteed consistency is proposed in [15]. This GUI is based on the last algorithm.

Subspace Identification

Consider the following state space representation for a linear system with *l* inputs and *m* outputs:

$$\begin{cases} x_{t+1} &= Ax_t + Bu_t + Ke_t \\ y_t &= Cx_t + Du_t + e_t \end{cases} \tag{1}$$

Where $\boldsymbol{x_t} \in \boldsymbol{R^n}$, $\boldsymbol{u_t} \in \boldsymbol{R^l}$, $\boldsymbol{y_t} \in \boldsymbol{R^m}$ and $\boldsymbol{e_t} \in \boldsymbol{R^m}$ is white noise. System (1) can be represented by the following basic subspace equations for the output and states ([16]):

$$Y_f = \Gamma_N X_f + L_u U_f + L_e E_f$$
$$X_f = A^N X_p + \Delta_N^d U_p + \Delta_N^s E_p$$

where subscript ‘p’ stands for ‘past’ and ’f’ stands for ‘future’. $\boldsymbol{X_f} \in \boldsymbol{R^{n \times j}}$ is the subspace matrix of future states given by

\begin{equation}

$$X_f = \begin{pmatrix} x_N & x_{N+1} & \cdots & x_{N+j-1} \end{pmatrix}$$

$\Gamma_N \in \boldsymbol{R}^{mN\times n}$ is the extended observability matrix which is defined by

$$\Gamma_N = \begin{pmatrix} C \\ CA \\ \cdots \\ CA^{N-1} \end{pmatrix}$$

$\boldsymbol{L_u} \in \boldsymbol{R}^{mN\times lN}$ and $\boldsymbol{L_e} \in \boldsymbol{R}^{mN\times mN}$ which provide information of the process and disturbance dynamics are defined as:

$$L_u = \begin{pmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ CA^{N-2}B & CA^{N-3}B & \cdots & D \end{pmatrix} \qquad (2)$$

$$L_e = \begin{pmatrix} I & 0 & \cdots & 0 \\ CK & I & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ CA^{N-2}K & CA^{N-3}K & \cdots & I \end{pmatrix}$$

$\Delta_N^{\boldsymbol{d}} \in \boldsymbol{R}^{n\times lN}$ and $\Delta_N^{\boldsymbol{s}} \in \boldsymbol{R}^{n\times mN}$ are the reversed extended controllability matrices of {A,B} and {A,K}, respectively, which are given by

$$\Delta_N^d = \mathcal{C}(A,B) \qquad , \qquad \Delta_N^d = \mathcal{C}(A,K)$$

where Ω operator is defined as:

$$\mathcal{C}(A,B) = (A^{N-1}B \;\; A^{N-2}B \;\; \cdots \;\; AB \;\; B)$$

$\boldsymbol{U}_{\boldsymbol{p}}$ and $\boldsymbol{U}_{\boldsymbol{f}} \in \boldsymbol{R}^{lN\times j}$ and $\boldsymbol{Y}_{\boldsymbol{p}}, \boldsymbol{Y}_{\boldsymbol{f}}, \boldsymbol{E}_{\boldsymbol{p}}$ and $\boldsymbol{E}_{\boldsymbol{f}} \in \boldsymbol{R}^{mN\times j}$ are data Hankel matrices for past and future inputs, outputs and noise. As an example, the input data Hankel matrices are given by

$$U_p = U_{0|N-1} = \begin{pmatrix} u_0 & u_1 & \cdots & u_{j-1} \\ u_1 & u_2 & \cdots & u_j \\ \cdots & \cdots & \cdots & \cdots \\ u_{N-1} & u_N & \cdots & u_{N+j-2} \end{pmatrix}$$

$$U_f = U_{N|2N-1} = \begin{pmatrix} u_N & u_{N+1} & \cdots & u_{N+j-1} \\ u_{N+1} & u_{N+2} & \cdots & u_{N+j} \\ \cdots & \cdots & \cdots & \cdots \\ u_{2N-1} & u_{2N} & \cdots & u_{2N+j-2} \end{pmatrix}$$

Other data Hankel matrices are defined similarly.

Typically, in open-loop subspace identification, *j* should be much larger than *max(mN,lN)* to reduce sensitivity to the noise. In fact, *j* has the same role as the number of observations in classic identification. Number of rows in the data Hankel matrices, *N*, is related to the order of system.

Using regression analysis approach [3], it can be shown that

$$Y_f = L_w W_p + L_u U_f + L_e E_f \tag{3}$$

where $\boldsymbol{L_w} \in \boldsymbol{R}^{\boldsymbol{mN \times (l+m)N}}$ and $\boldsymbol{W_p}$ is defined by

$$W_p = \begin{pmatrix} Y_p \\ U_p \end{pmatrix}$$

This provides a form of input-output equation in subspace framework.

Subspace definitions in a closed-loop system

In a closed-loop system such as Fig. 7, two separate open-loop models can be defined: a model from setpoint $\boldsymbol{r_t}$ to the output $\boldsymbol{y_t}$ and the other one from $\boldsymbol{r_t}$ to the controller output $\boldsymbol{u_t}$. Similar to Eq. (3), these two systems can be presented by the following input-output relations [15]:

$$Y_f = L_Y \begin{pmatrix} Y_p \\ R_p \end{pmatrix} + L_{YR} R_f + L_{YE} E_f$$

$$U_f = L_U \begin{pmatrix} U_p \\ R_p \end{pmatrix} + L_{UR} R_f + L_{UE} E_f$$

And we define:

$$W_p^Y = \begin{pmatrix} Y_p \\ R_p \end{pmatrix} \quad and \quad W_p^U = \begin{pmatrix} U_p \\ R_p \end{pmatrix}$$

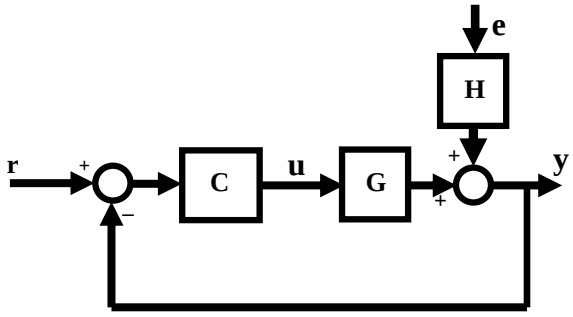

Figure 7: Residue test results

Choosing $\boldsymbol{R_f}$ to be uncorrelated with $\boldsymbol{E_f}$, $\boldsymbol{W_p^U}$ and $\boldsymbol{W_p^Y}$, each of these open-loop problems can be solved by least square estimation.

In [15], it has been shown that the open-loop subspace matrices $\boldsymbol{L_u}$, $\boldsymbol{L_e}$, Γ_N and the input noise covariance matrix can be estimated from the above closed-loop subspace matrices, $\boldsymbol{L_{YR}}$, $\boldsymbol{L_{YR}}$ and $\boldsymbol{L_{UE}}$, using a joint input-output framework. Then state space matrices can be obtained as follows:

$$\begin{aligned}
\Gamma_N^u &= \Gamma_N(1:(N-1)m,:) \\
\Gamma_N^y &= \Gamma_N(m+1:Nm+,:) \\
\hat{C} &= \Gamma_N(1:m,1:n) \\
\hat{A} &= (\Gamma_N^u)^\dagger \Gamma_N^y \\
\hat{B} &= (\Gamma_N^u)^\dagger L_u(m+1:Nm,1:l) \\
\hat{D} &= L_u(1:m,1:l) \\
\hat{K} &= (\Gamma_N^u)^\dagger \hat{P}_v(m+1:Nm,1:m)\hat{R}^{-1}
\end{aligned}$$

Please see Ref. [3, 14, 15] for more details on the algorithm.

The $\boldsymbol{L_u}$ and $\boldsymbol{L_e}$ matrices given in (2) contain the impulse response coefficients (Markov parameters) of the process. Therefore, step response coefficients between each input and output can be directly extracted out of these subspace matrices without having the explicit model of the process. In fact, this approach provides more reliable estimation of the step responses, specially in the case where the final model realization step does not result in an accurate model. In this toolbox, nonlinear regression can be used to fit a continuous-time transfer function model to each set of step response coefficients. Both 'First-Order Plus Time-Delay' (FOPTD) and 'second-Order Plus Time-Delay' (SOPTD) models are fitted to each set and better the fit is chosen.

Data Storage

There is a code provided in file "**gen_cmpct_data.p**" which can be used to generate compact data objects and save them on the disk for using in the GUI. The data format generated by this code is as follows:

Data Storage Format

The data format is called *Compact* format. Assume that there are ***j*** samples of output, input and setpoint data with a total of ***m*** controlled variables and ***l*** manipulated variables with a total of ***q*** tags. In the compact data storage method, the data is stored as an object containing the following entries:

1) **controller_Status**: This is a *j*-by-1 double matrix that contains the status of each of the controllers, where 1 represents a controller that is "on" and 0 represents a controller that is "off."
2) **cell_char_TagList**: This is a *q*-by-1 cell matrix that contains the name of each of the tags that are presented in the process.
3) **cell_char_TimeStamp**: This is a *j*-by-1 cell matrix that contains the time stamp for each of the samples.
4) **dbl_Compact_Data**: This is a *j*-by-(2*m*+*l*) double matrix that contains the values for each of the outputs, inputs and setpoints at sample periods.
5) **dbl_SamplingTime**: This is a scalar double that contains the sampling time for the data.

6) **int_CVNumber**: This is a scalar integer that contains the number of controlled variables in the process, that is, *m*.
7) **int_MVNumber**: This is a scalar integer that contains the number of manipulated variables in the process, that is, *l*.
8) **status**: This is a *j*-by-(*l* + *m*) double matrix that stores the data in the following manner: The first *t* columns contain the status of the controller variables, while the remaining *s* columns contain the status of the manipulated variables. A value of 1 signifies that the data is good.

Data Generation

The data storage file can be generated using "**gen_cmpct_data.p**". Based on the step-by-step comments after running the code, you would be able to create your data file in a *Compact* format. A set of **closed-loop** experiment data (identification data) must be available in the Workspace.

Run the code:

If your data is ready in the Workspace, run the code. The followings will appear on the command window:

```
************************************************************
Consider a process with m inputs and p outputs:
The following information should be available in WORKSPACE:

 1. CL Excitation data; Output, Input and Setpoint data (y,u,r)
 2. Sampling time
************************************************************
Press ENTER to continue or type X to exit :
```

At this stage, you may want to stop going further (by typing X) to prepare your data in the Workspace.

You may press ENTER to go further and provide **closed-loop experiment data** as the following message appears:

Provide Closed-loop excitation data matrix [y u r] :

Provide the data arranged in the given order: output, input and setpoint. Keep the '[' and ']' symbols. If presses enter without providing closed-loop experiment data, the code will stop.

Then, you will be asked about the number of process outputs:
number of outputs:

Sampling time will be asked at this stage:

Sampling time:

Name of the files for saving the *Compact* data is requested at the last step:

The file name to save compact data (e.g. test_cmpct_data):

The "gen_cmpct_data" code, will save a '.mat' file on the current active path of MATLAB which you can upload and use in the GUI for analysis.

Using the Toolbox

Installation

The toolbox can be installed by simply unzipping the files to any desired location. In order for the toolbox to function properly, the System Identification Toolbox should be installed.

Starting the Toolbox

The toolbox can be accessed from MATLAB using the following sequence of commands. First MATLAB itself should be started from the directory pointing to the folder containing the files for this toolbox. Next, at the command prompt, type **"`>> main_CLsysID`"**.

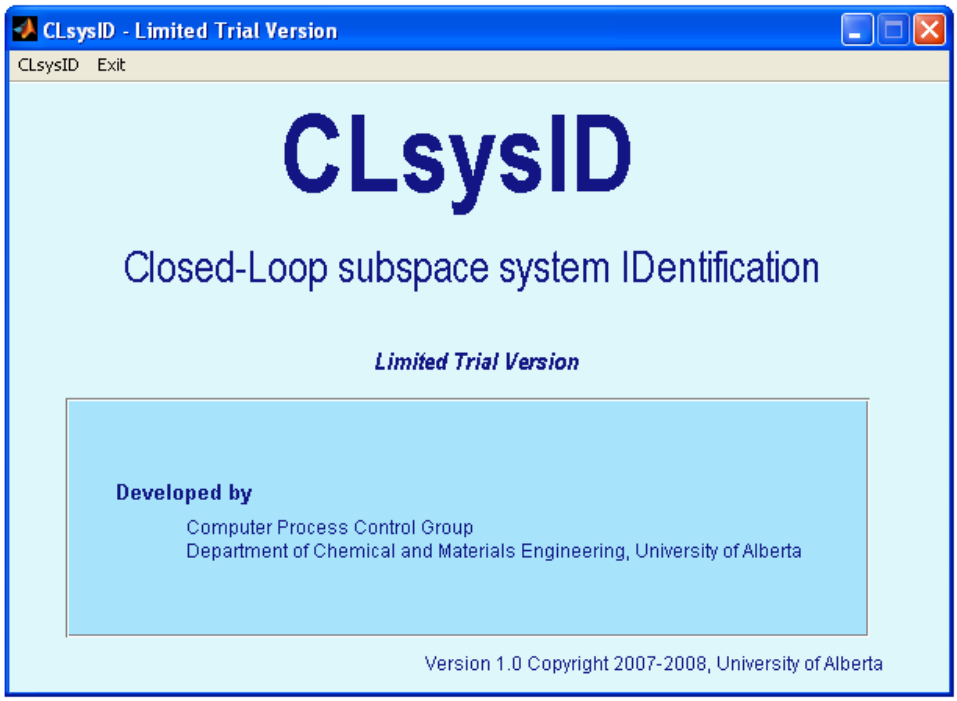

Figure 8: The first GUI that appears

The GUI shown in Figure 8 should appear. This GUI is the main access to the toolbox. To start a session of the toolbox, click on the **'CLsysID' menu**. This will bring up a new GUI, which is shown in Figure 9. Each of the main parts of the GUI in Figure 9 will be discussed separately later.

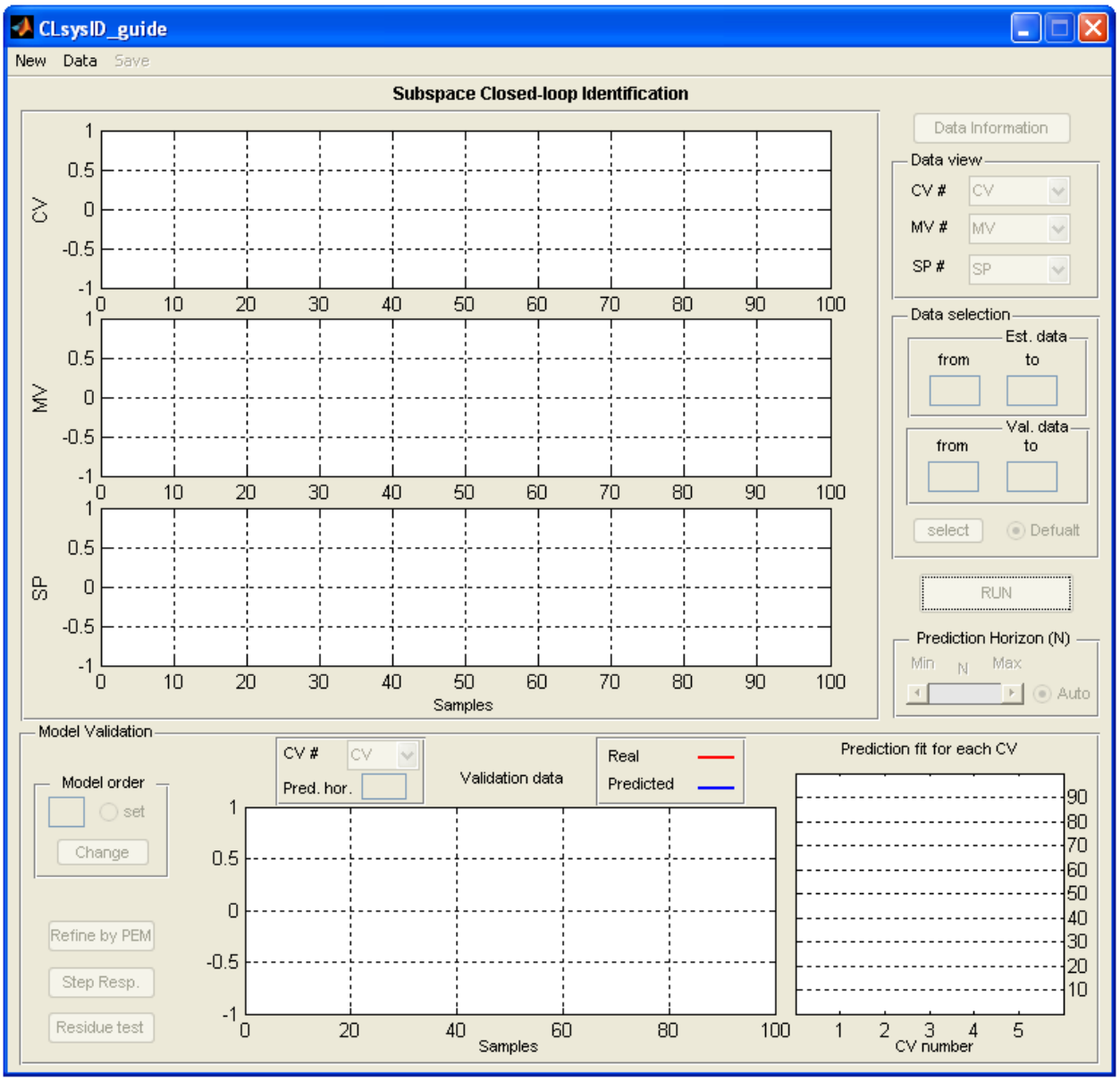

Figure 9: The main GUI for this toolbox

In-depth Discussion of the Toolbox

Section 1: Main Menu

The Main Menu consists of the following 3 areas:

1) **New:** Clicking this menu will clear all the data from the current GUI and allow the user to restart the analysis from a clean layout.
2) **Data:** Clicking this menu will allow the user to upload the closed-loop experiment data for analysis.

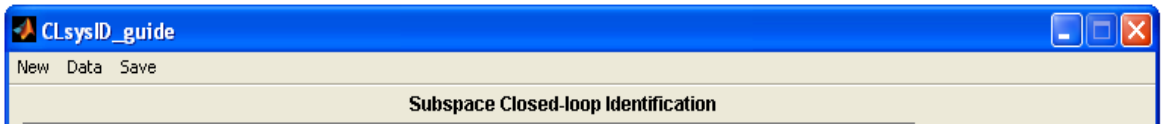

Figure 10: The 'Data' menu

3) **Save:** Clicking this menu will allow the user to save the resulted state space model or step response coefficients on the disk.

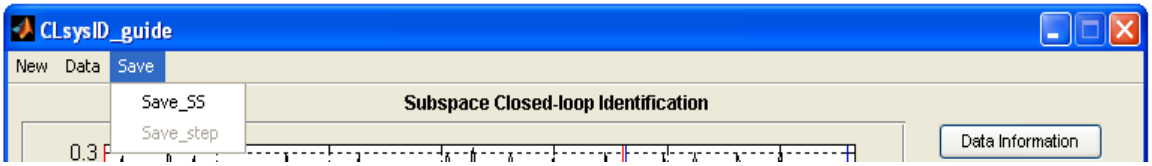

Figure 11: The 'Save' menu

Section 2: 'Data Information' button

This button opens a new window which shows the information about the uploaded experiment data. Data file name, number of CVs and MVs, sampling time, Starting and Ending time of the data are provided in this window.

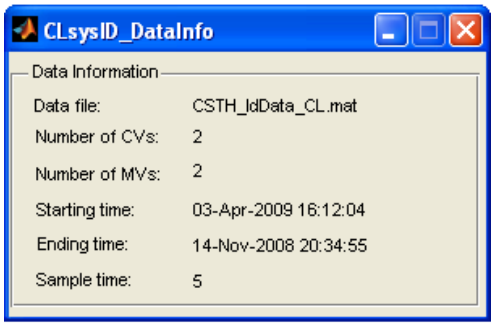

Figure 12: The 'Data Information' window

Section 3: 'Data view & selection' panels

The tools in these panels can be used for data selection and vision as follows:

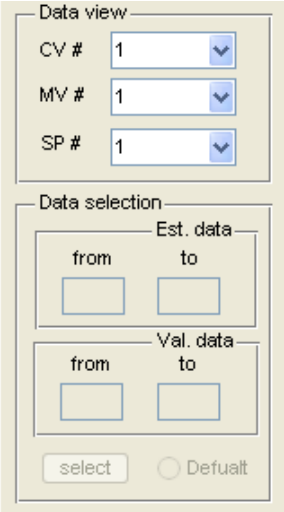

Figure 13: The 'Data ' panel

1) **Data view:** This pane enables you to choose the controlled variable (CV), manipulated variable (MV) or setpoint variable (SP) which you want to be shown on the corresponding axes.
2) **Data selection:** In this filed, the user can specify what parts of the data should be used as identification and validation sections. Two entries for starting samples and two entries for ending values are considered. The ending value for each section must be greater than its starting value. Clicking **'select'** will display the selected identification section with red lines and validation section with blue lines.

Section 3: 'Run' panel

1) **Run:** This button is used to run the main algorithm of closed-loop identification. The button would be disabled until the data is uploaded.

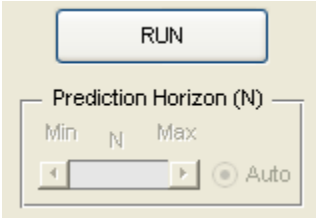

Figure 14: The 'Run ' panel

2) **‘Prediction Horizon (N)’ panel:** A slider has been provided in this panel which can be used to change the value of prediction horizon used for the subspace identification. For the definition of this parameter please refer to the **‘Theory’** section. At the first run, this parameter is being estimated by the algorithm and a proper range for possible changes is provided. The user is then able to change the value of **N** which results in the algorithm to be run again with new **N**.

   For the slider to be enabled, the radio button **‘Auto’** must be off. When it is on, the algorithm estimates a reasonable value for **N.**

Section 4: Plot data

Three axes have been provided in this section which allows the user to view CV, MV and SP data. Each one can only show one set of data at a time, so you should use **‘Data selection’ panel** to choose the number of CV, MV or SP you want to view.

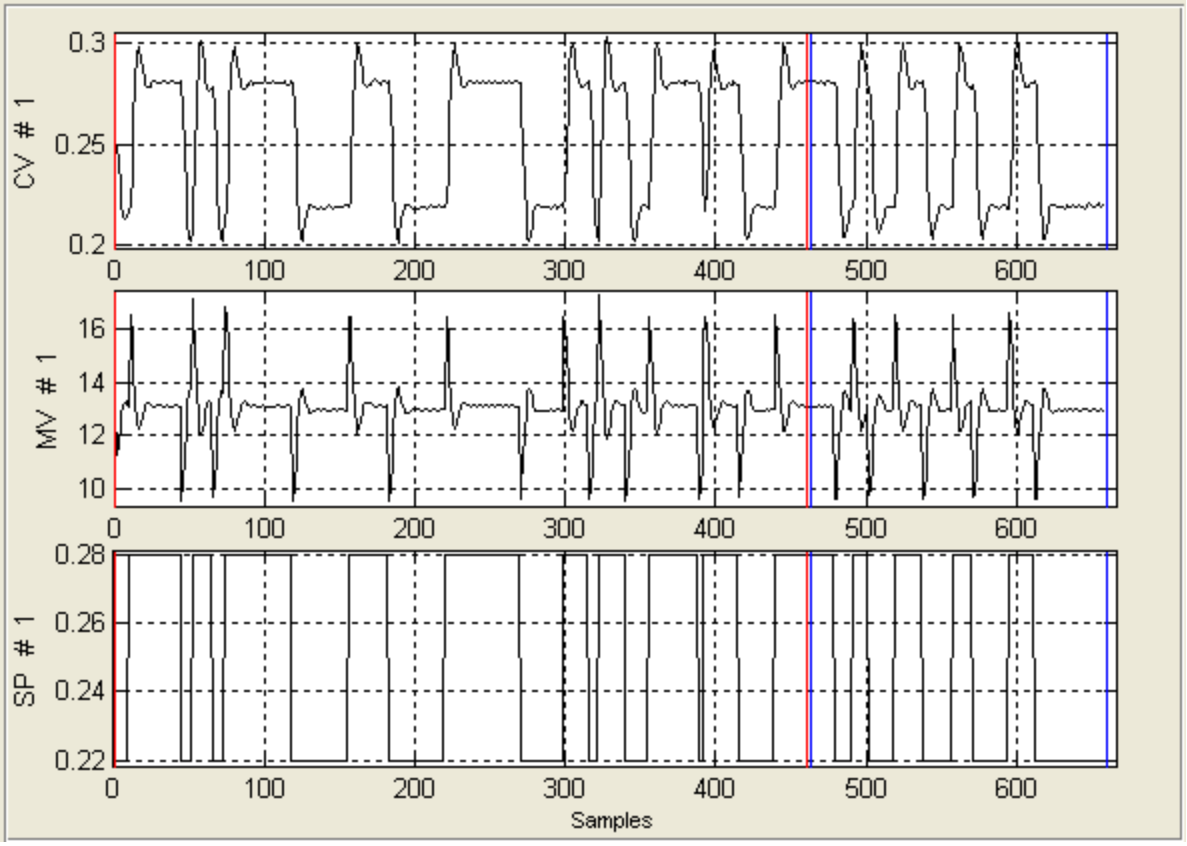

Figure 15: The panel being used for plotting data

Section 5: 'Validation' panel

In this panel, the results of two model validation tests are provided including prediction fit and residue test. The estimated model order is shown here and can be changed. The step response estimation can also be performed in this panel. Accessories in the panel are as follows:

1) **'Model order' sub-panel:** The estimated order for the state space model is shown in this sub-panel. By enabling the **'Set'** option, you will be able to change the order. When the new order is entered, press the **'Change'** button to run the order change algorithm. The calculations will be fast, because subspace matrices are not required to be estimated again. After pressing the **'Change'** button, the new prediction fit results will be automatically uploaded to the GUI.

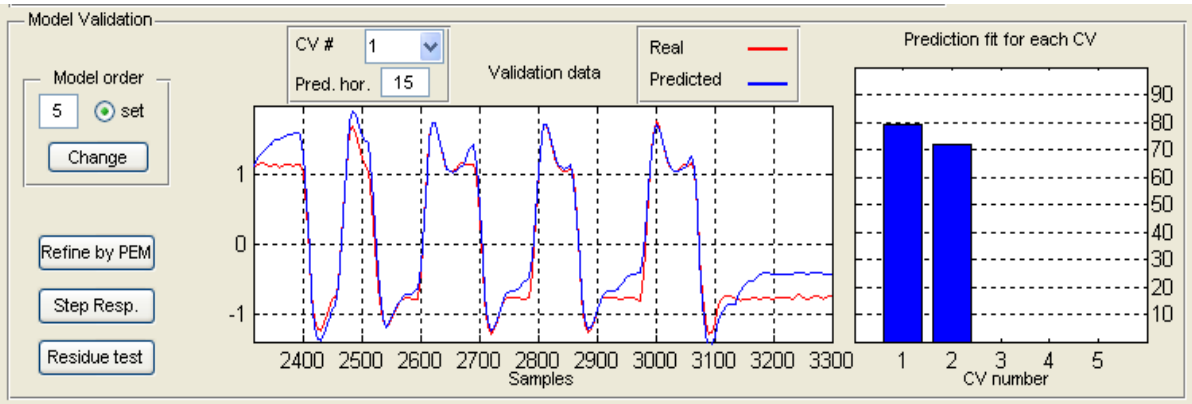

Figure 16: 'Validation' panel

2) **'Validation Data' axis:** This axis shows the validation data as well as the model prediction. Only one pair of data can be shown at one time, so one has to choose which output to be viewed using the **'CV'** popup menu placed at the top of the axes. The prediction horizon can also be changed by the **'Pred. hor.'** edit box. The default value for the prediction horizon is 15.
3) **'Prediction fit for each CV' axis:** This axis provides the results of comparing the validation output to the model prediction for each of the process outputs. Each bar presents the fit percentage for one controlled variable.

In this panel, three push buttons are provided as follows:

1) **Refine by PEM:** Pressing this button will run the '**pem'** function of the system identification toolbox to refine the identified model resulted from subspace identification. Note that it may not result in a better model necessarily. If the new model is not desired and you want to get back to the previously estimated model (from subspace identification), press the **'Change'** button.
2) **Residue test:** This button can be used to view the results of residue test on the last estimated model. Only the cross-correlation test is performed, so the disturbance model is not tested.

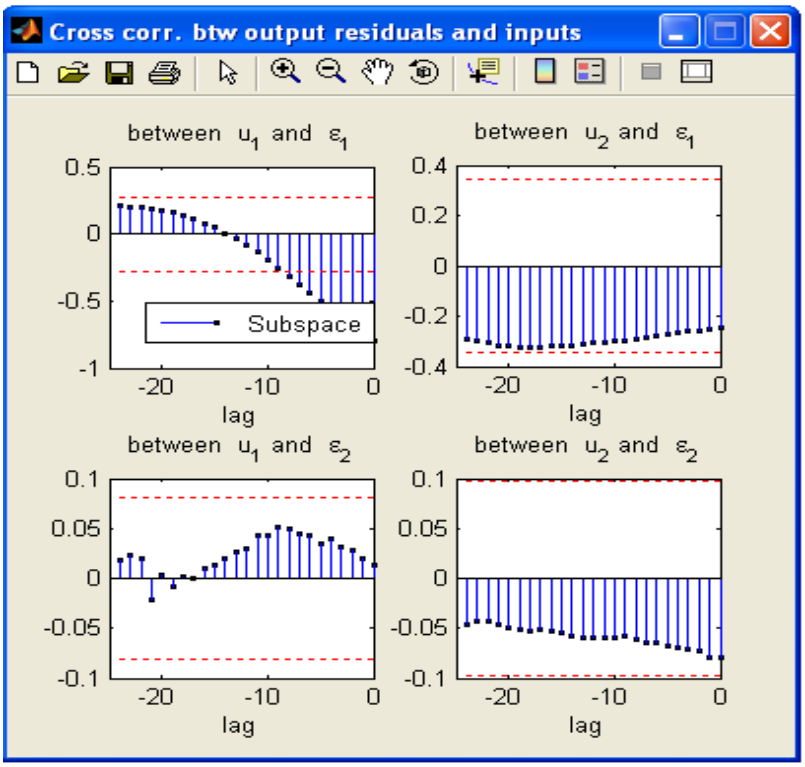

Figure 17: Residue test

3) **Step Resp.:** By pressing this button, a new window is opened which shows the step responses coefficients directly estimated from the subspace matrices.

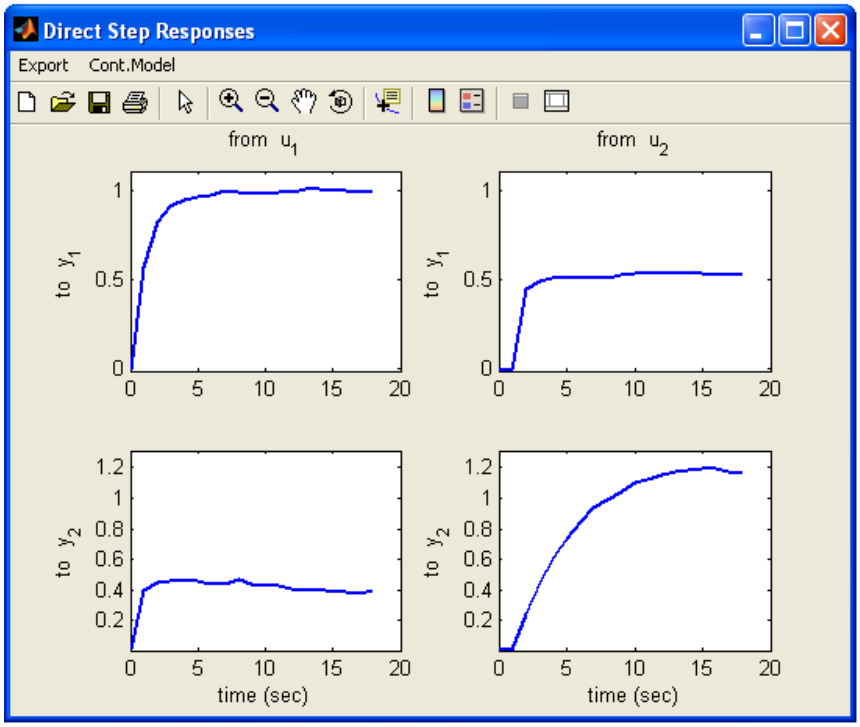

Figure 18: Step response coefficients

The **"Export"** menu in this figure can be used to save the step response coefficients on the disk.

The **"Cont. Model"** menu in this figure can be used to obtain the continuous-time model (FOPTD or SOPTD) for the estimated step response coefficients. A window as in Figure 19 will be opened to enter the model order for each step response.

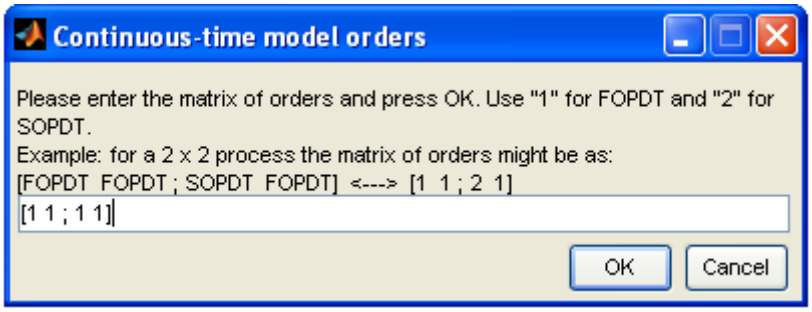

Figure 19: Window for Continuous-time model orders

A nonlinear regression algorithm from statistics toolbox of MATLAB is used to fit a continuous-time model to each step response. A new window will show the results (see Figure 20). **Note that the results of this calculation may not be reliable.**

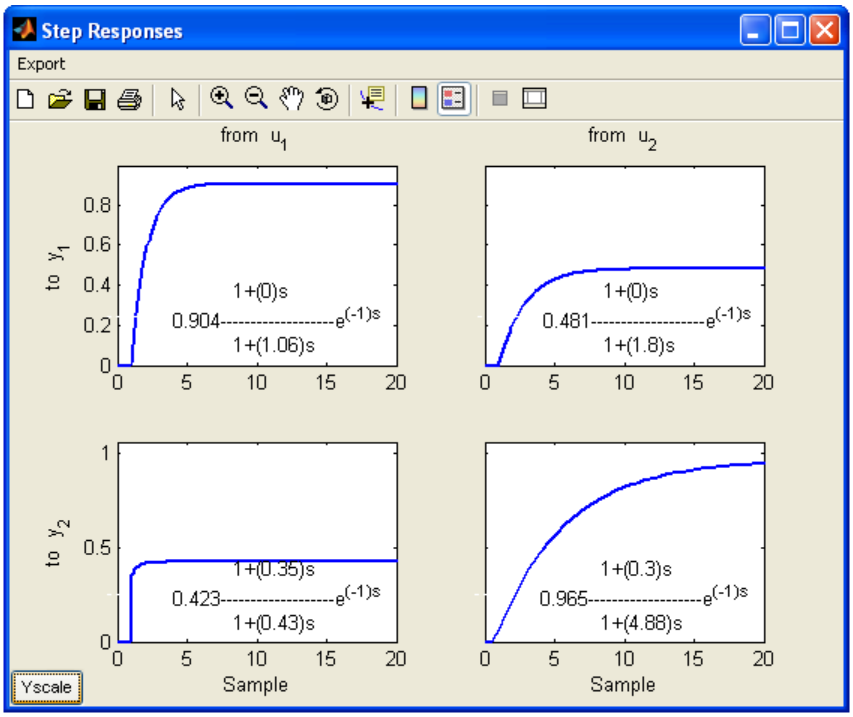

Figure 20: The continuous-time models fitted to the step responses coefficients

Example 1

A set of closed-loop identification data generated from the following system is provided with the toolbox in "**SISO_IdData_CL.mat**" :

$$\boldsymbol{x}_{t+1} = \begin{pmatrix} 0.6 & 0.6 & 0 \\ -0.6 & 0.6 & 0 \\ 0 & 0 & 0.7 \end{pmatrix} \boldsymbol{x}_t + \begin{pmatrix} 1.6161 \\ -0.3481 \\ 2.6319 \end{pmatrix} \boldsymbol{u}_t + \begin{pmatrix} -1.1472 \\ -1.5204 \\ -3.1993 \end{pmatrix} \boldsymbol{e}_t$$

$$\boldsymbol{y}_t = \begin{pmatrix} -0.4373 & -0.5046 & 0.0936 \end{pmatrix} \boldsymbol{x}_t - 0.7759 \boldsymbol{u}_t + \boldsymbol{e}_t$$

A PI controller, $[\,0.1 + \frac{0.05}{\boldsymbol{s}}\,]$ is used for this process. A simple Simulink model has been prepared to collect closed-loop experimental data (see Figure 21). The identification test signal, $\boldsymbol{r}(\boldsymbol{t})$, is designed by '*idinput*' command:

```
r=idinput(1000,'rbs',[0,0.06], [-1,1]);
```

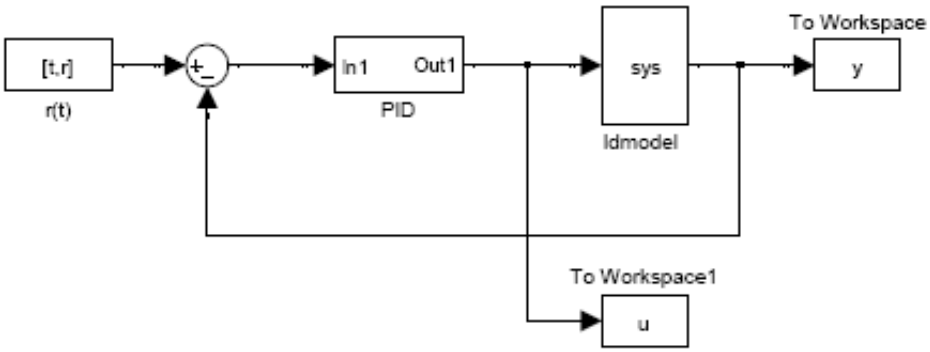

Figure 21: Simulink model for collecting closed-loop data

The '*idmodel*' block needs a variable named 'sys' to be defined and be ready in the workspace. This can be done as follows:

```
A=[0.6 0.6 0;-0.6 0.6 0;0 0 0.7];
B=[1.6161; -0.3481; 2.6319];
C=[-0.4373 -0.5046 0.0936];
Du=-0.7759;
K=[-1.1472; -1.5204; -3.1993];
sys=idss(A,B,C,D,K,[0;0;0],1);
sys.NoiseVariance=0.01;
```

In the block parameters of '*idmodel*', there is an option "Add noise" which must be checked. When data variables are present in Workspace, we run "gen_cmpct_data.p" to create and save the data files in *Compact* format named as "SISO_IdData_CL".

Loading this data file in GUI, the state space model and continuous-time transfer function model of this process can be estimated.

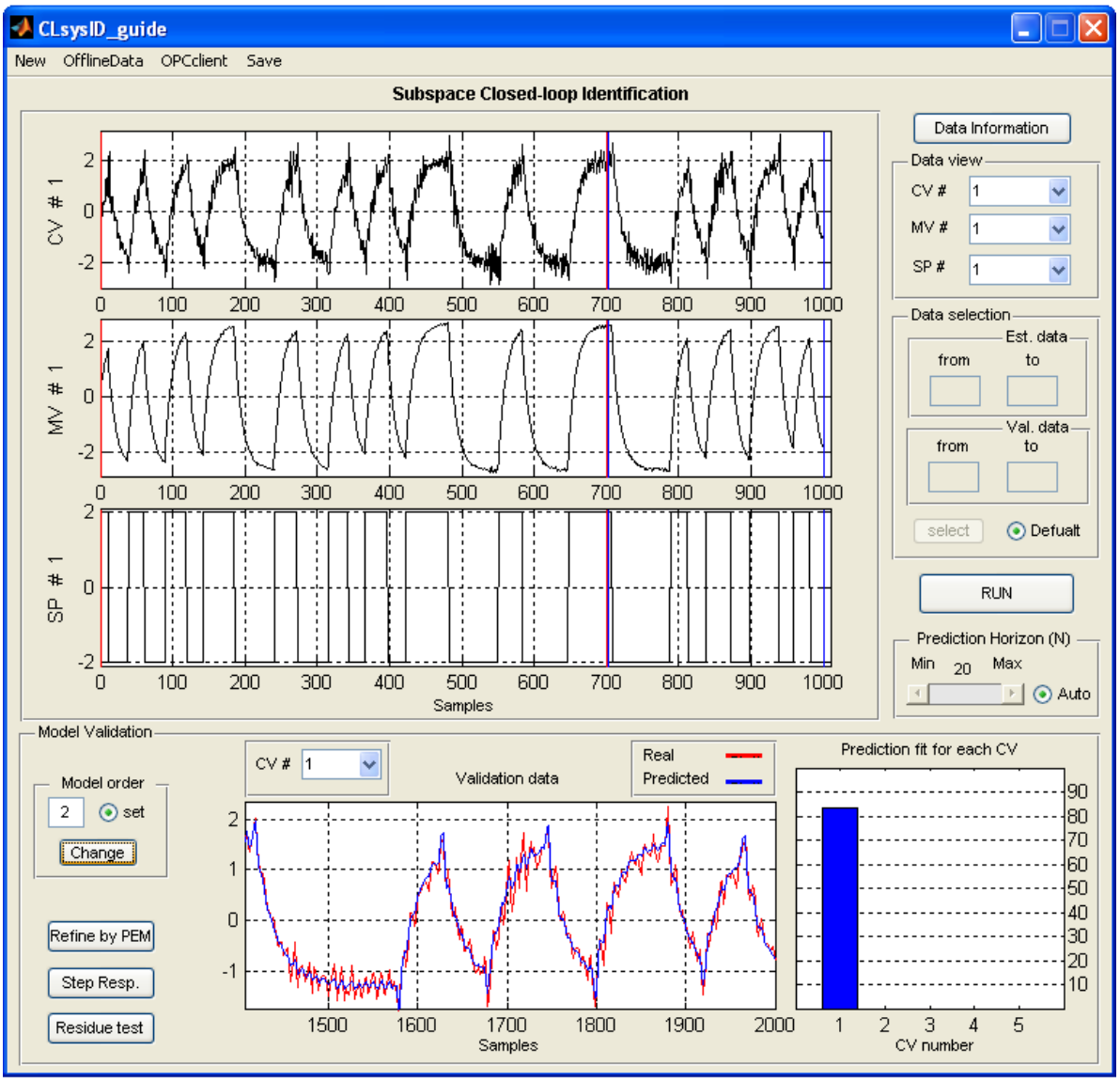

Figure 22: Results of closed-loop identification for example 1

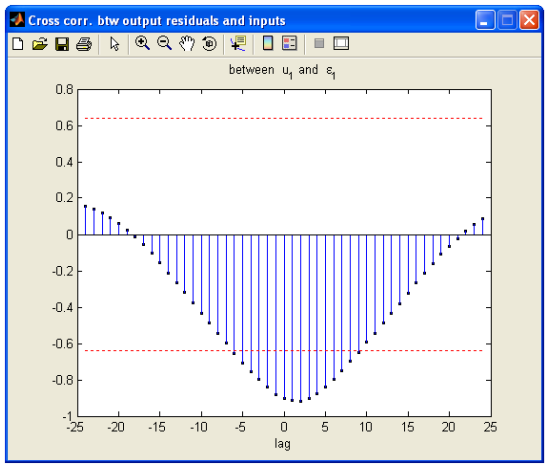

Figure 23: Results of residue test for example 1

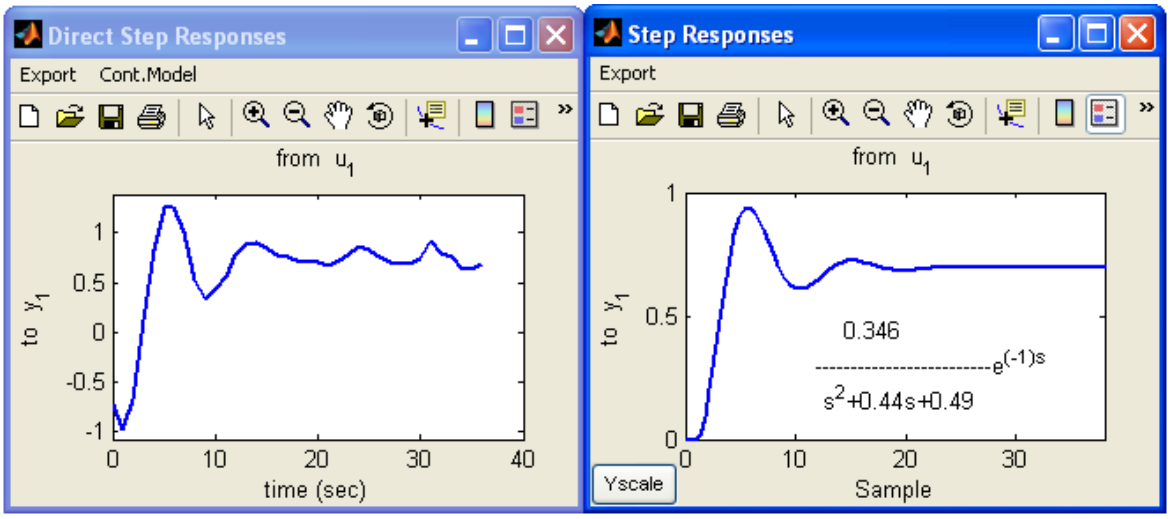

Figure 24: Results of step response estimation for example 1

Example 2

A MIMO example is also provided with the toolbox. The process to be controlled is a 2 input, 2 output process described by the following transfer function matrices:

$$G_p = \begin{pmatrix} \dfrac{z^{-1}}{1-0.4z^{-1}} & \dfrac{0.5z^{-2}}{1-0.1z^{-1}} \\ \dfrac{0.3z^{-1}}{1-0.4z^{-1}} & \dfrac{z^{-2}}{1-0.8z^{-1}} \end{pmatrix} \quad \textit{and} \quad G_l = \begin{pmatrix} \dfrac{1}{1-0.5z^{-1}} & \dfrac{-0.6z^{-1}}{1-0.5z^{-1}} \\ \dfrac{0.5z^{-1}}{1-0.5z^{-1}} & \dfrac{1}{1-0.5z^{-1}} \end{pmatrix}$$

The following controller is implemented on the process:

$$\boldsymbol{G}_c = \begin{pmatrix} \dfrac{0.5 - 0.2\boldsymbol{z}^{-1}}{1 - 0.5\boldsymbol{z}^{-1}} & 0 \\ 0 & \dfrac{0.25 - 0.2\boldsymbol{z}^{-1}}{(1 - 0.5\boldsymbol{z}^{-1})(1 + 0.5\boldsymbol{z}^{-1})} \end{pmatrix}$$

The following procedure can be followed to generate a set of closed-loop experiment data from this process:

```
Gp=tf({[1],[4];[0.3],[1]},{[0 1 -.4],[1 -0.1 0];[ 0 1 -0.1],[1 -0.8 0]},1);
Gl=tf({[1 0],[-.6];[0.5],[1 0]},{[1 -.5],[1 -.5];[1 -.5],[1 -.5]},1);
Gc=tf({[.5 -.2],[0];[0],[.25 -.2 0]},{[1 -.5],[1];[1],[1 0 -.25]},1);
r1=idinput(1000,'rbs',[0,0.03], [-5,5]);
r2=idinput(1000,'rbs',[0,0.03], [-5,5]);
r = [r1 r2];
t=[1:1000]';
[A,B,C,D,K] = tf2ssGpGl(Gp,Gl);
% controller
cont=idss(Gc);
Ac = cont.a; Bc = cont.b; Cc = cont.c; Dc = cont.d;
seeds = [1 2];


function [A,B,C,D,K] = tf2ssGpGl(Gp,Gl)
    ny = size(Gp.OutputDelay,1);
    nu = size(Gp.InputDelay,1);
    NUMGt={Gp.num Gl.num};
    DENGt={Gp.den Gl.den};
    Gt = tf([NUMGt{1,1} NUMGt{1,2}] ,[DENGt{1,1} DENGt{1,2}] ,1);
    set(Gt,'InputGroup',struct('Noise',[nu+1:nu+ny]))
    model = idss(Gt);
    A = model.a; B = model.b; C = model.c;
    D = model.d; K = model.k;
end
```

Then use a Simulink model similar to Example 1 to generate data.

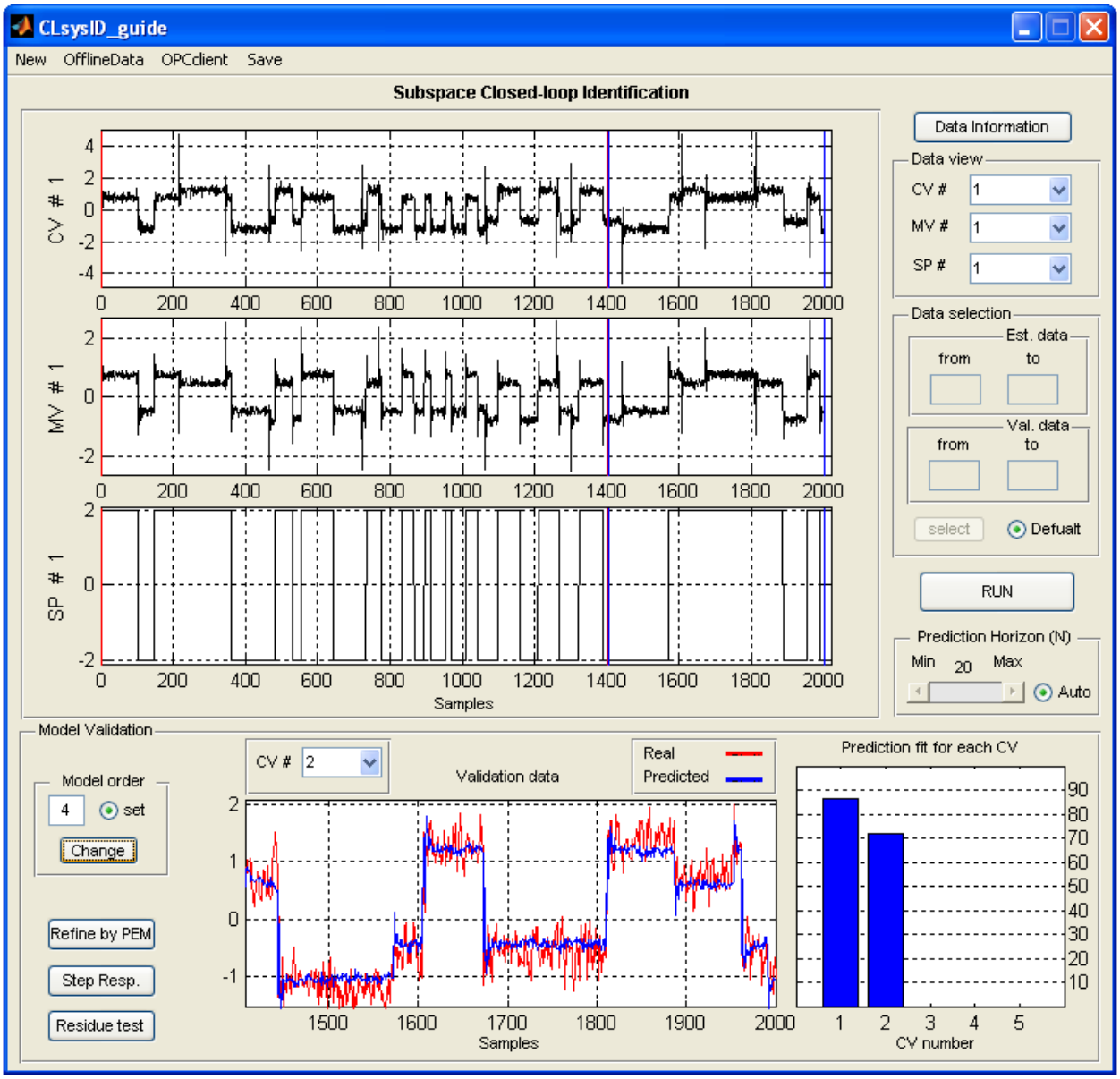

Figure 25: Results of closed-loop identification for example 2

The same procedure as the previous example is followed to create and save *Compact* data file named "**MIMO_IdData_CL**", load it in the GUI and obtain the model. Results are shown in Figures 24 to 26.

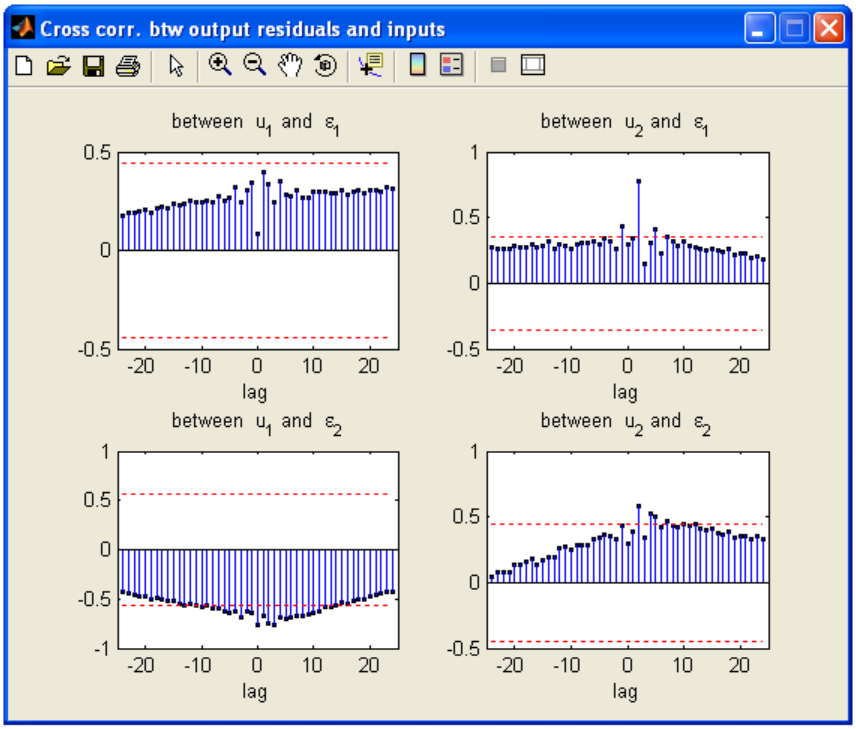

Figure 26: Results of residue test for example 2

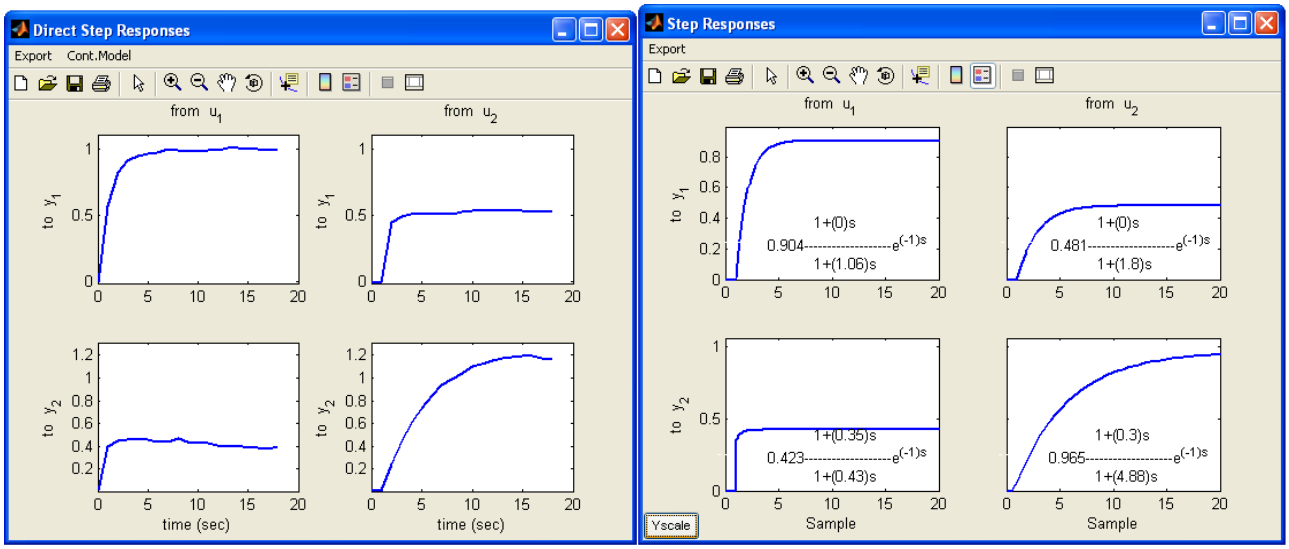

Figure 27: Results of step response estimation for example 2

References

1- L. Ljung. *System identification: Theory for the user*. Prentice-Hall, 1999.

2- T. Soderstrom and P. Stoica. *System Identification*. Prentice Hall International, 1989.

3- T. Knudsen. Consistency analysis of subspace identification methods based on linear regression approach. *Automatica*, 37:81-89, 2001.

4- W.E. Larimore. Statistical optimality and canonical variate analysis system identification. *Signal Processing, 52:131-144, 1996.*

5- M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle. On- and off-line identification of linear state-space models. *International J. Control*, 49(1):219-232, 1989.

6- P. Van Overschee and B. De Moor. N4sid: Subspace algorithm for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75-93, 1994.

7- P. Van Overschee and B. De Moor. A unifying theorem for three subspace system identification algorithms. *Automatica*, 31(12):1877-1883, 1995.

8- M. Verhaegen and P. Dewilde. Subspace model identification part 1. the output-error state-space model identification class of algorithms. *International J. of Control, 56(5):1187-1210, 1992.*

9- P. Van Overschee and B. de Moor. *Subspace identification for Linear Systems; Theory, Implementation, Application. Katholieke Universiteit, 1996.*

10- L. Ljung and T. McKelvey. Subspace identification from closed loop data. *Signal Processing*, 52:209-215, 1996.

11- A.K. Tangirala, S. Lakshminarayanan, and S.L. Shah. Closed-loop identification using canonical variate analysis. Edmonton, Canada, 1997. 47th CSChE Confer ence.

12- P. Van Overschee and B. De Moor. Closed-loop sub space system identification. In *36th conference on Decision and Control, pages 1834{1853, 1997.*

13- R. Kadali and B. Huang. Estimation of dynamic matrix and noise model for predictive control using closed-loop data. *Ind. Eng. Chem. Res.*, 41:842{852, 2002..

14- Danesh Pour, N. Huang, B. and Shah S.L. Closed-loop subspace identification for performance assessment. Submitted to *Journal of Process Control*, December 2008.

15- B. Huang and R. Kadali. *Dynamic Modeling, Predictive Control and Performance Monitoring: A Data- driven Subspace Approach. Springer Verlag, London, 2008.*