University of Alberta Computer Process Control Group

LQG Benchmark for Performance Assessment: Subspace Approach

Limited Trial Version

Written by CPC Control Group, University of Alberta

Version 2.0

Table of Contents

List of Figures

Introduction

The "LQG benchmark for Performance Assessment: Subspace Approach" toolbox was developed by the Computer Process Control Group at the University of Alberta to allow performance assessment 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_LQGPA`"** to start the toolbox. The GUI shown in Figure 1 should appear.

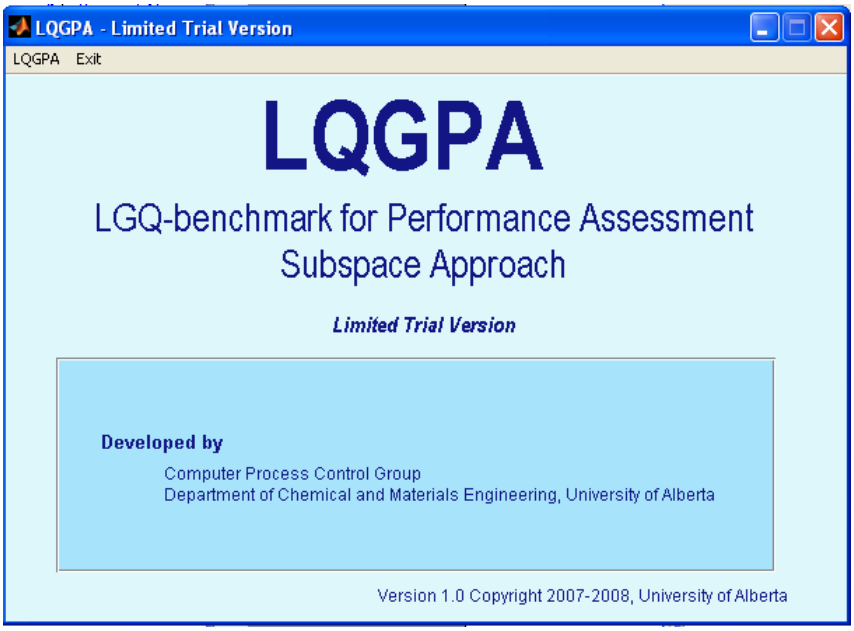

Figure 1: The first GUI that appears

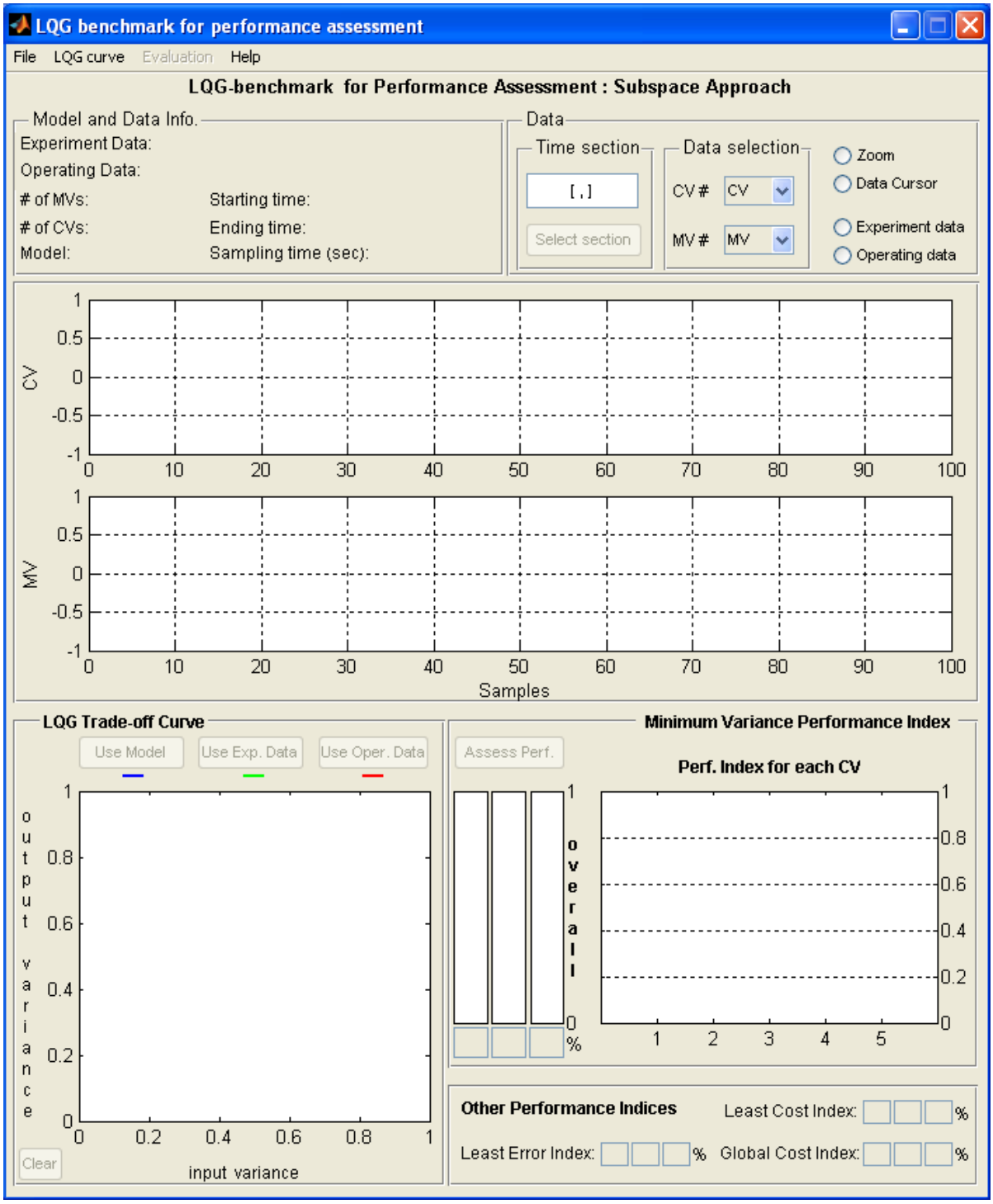

Figure 2: The main GUI for this toolbox

4) Press the **LQGPA menu**. The new GUI that is shown in Figure 2 will appear.
5) This GUI tool provides different modes of operations to load a model. The GUI accepts any of the following options:

a. Loading an open-loop model of the process (both process and disturbance models), in transfer function or state space form, the LQG trade-off curve can be calculated.
b. Loading a set of open-loop experiment (identification) data, the LQG trade-off curve can be calculated.
c. Loading a set of closed-loop experiment (identification) data, the LQG trade-off curve can be calculated.
d. Loading only process model (with no disturbance model) and a set of closed-loop routine operating data, the LQG trade-off curve can be calculated.

6) After loading the model or data, as a measure of current performance, a set of closed-loop routine operating data should be loaded.
7) For the purpose of this quick start, we will use the sample data provided in the SISO example in the zip file which contains process models in both transfer function (tf) and state space (ss) formats, a set of open-loop experiment data, a set of closed-loop experiment data, and a set of closed-loop operating data.
8) Use the **LQG curve** menu to load the model or data. When the model or data is loaded, the corresponding information will be shown in the **Model and Data Info.** panel (see Figure 3 and Figure 4).

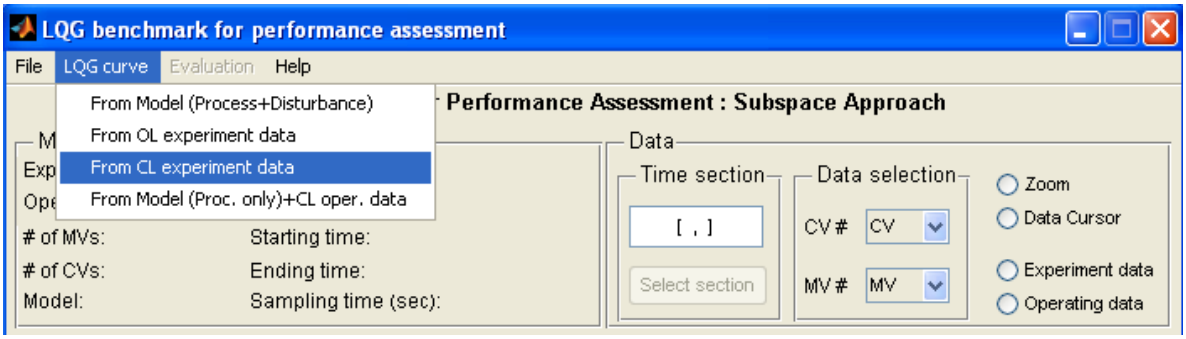

Figure 3: Loading data

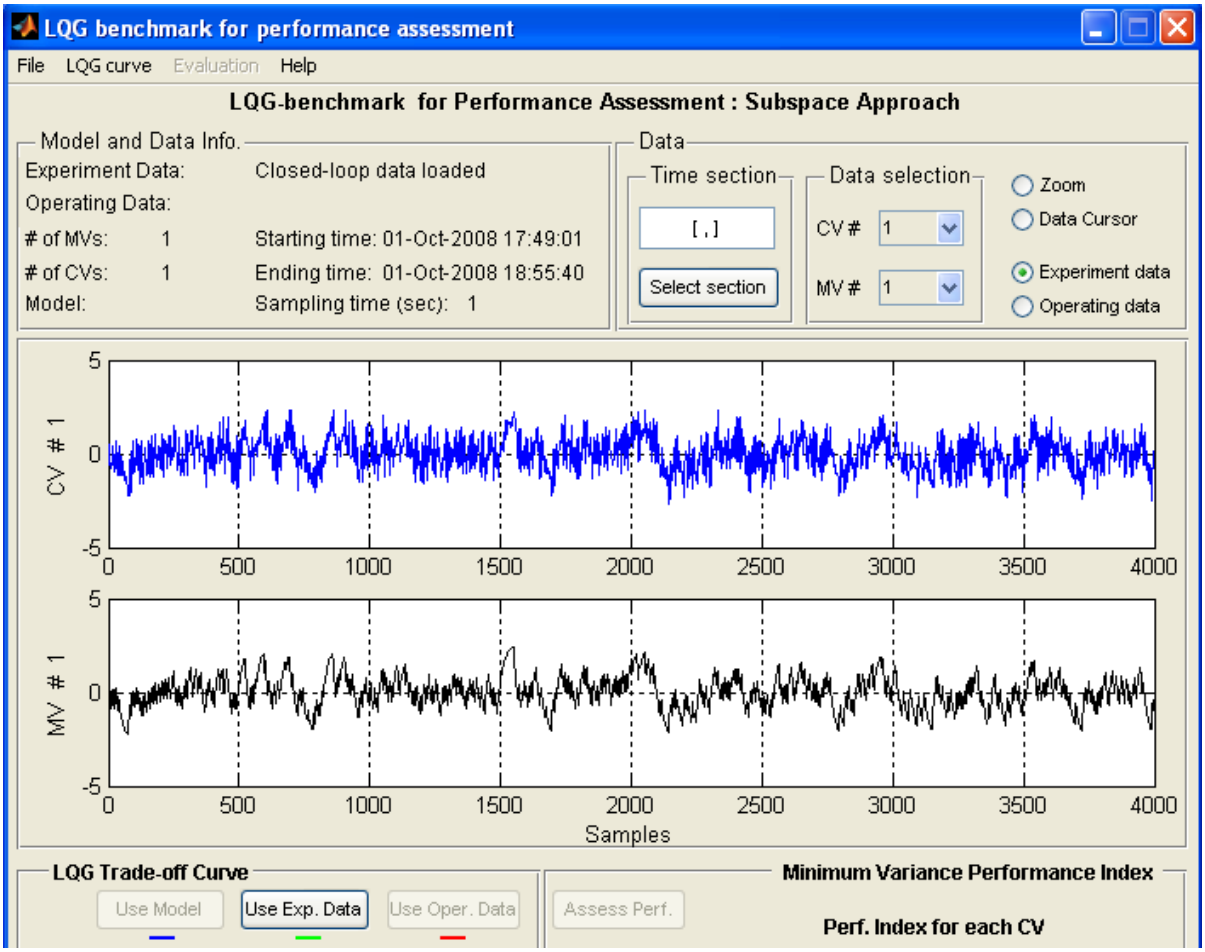

Figure 4: Data information and plots

9) If an open-loop model is loaded, you may press the **Use Model button** to generate the trade-off curve (in blue) as in Figure 5. Depending on the type of process dynamics and number of inputs and outputs, it takes a few seconds to a few minutes to perform the calculation.

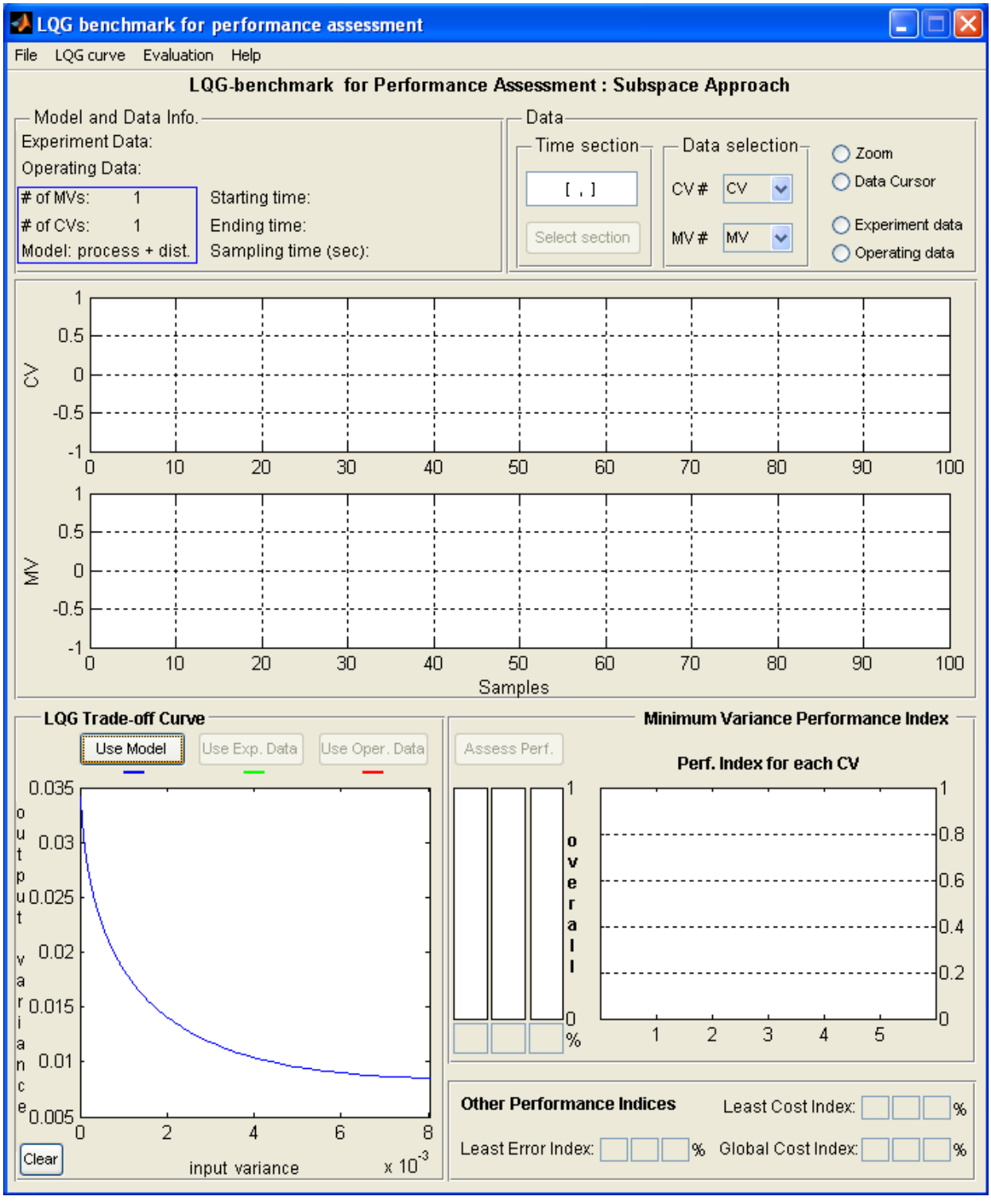

Figure 5: LQG curve calculated using the provided complete model

10) If a set of closed-loop or open-loop experiment is loaded, you can press **Use Exp. Data** to estimate and plot the curve (in green) as in Figure 6. Depending on the type of process

dynamics, number of inputs and outputs and number of data points, it will need a few seconds to a few minutes to perform the estimation.

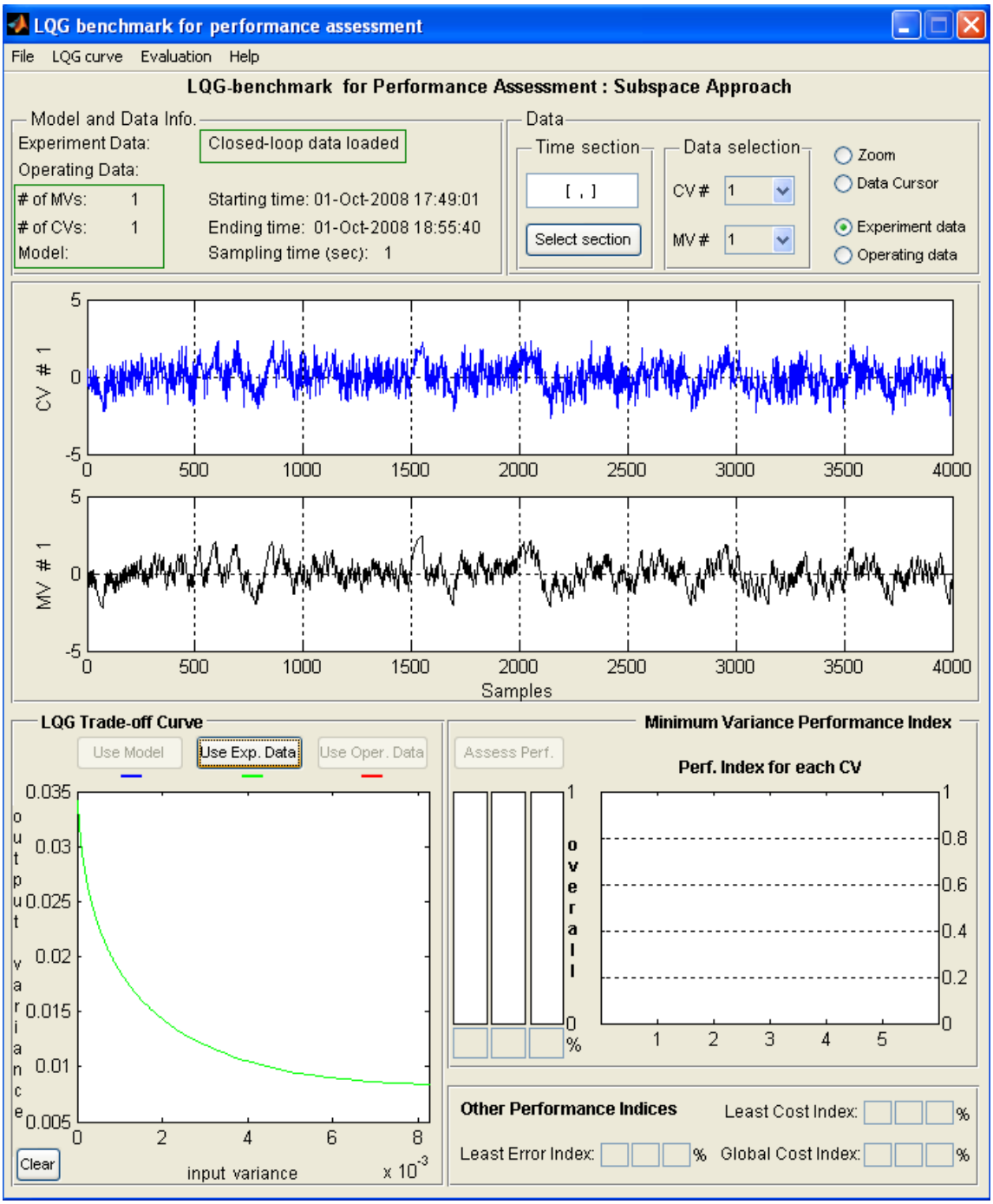

Figure 6: LQG curve estimated form closed-loop experiment data

11) If an open-loop model along with a set of closed-loop operating data is loaded, then you can get the curve by pressing **Use Oper. Data** button. The result will be plotted in red.

12) To perform a comparison between the real and estimated curves (when both model and data are available), you can load a model and a set of identification data and generate both true curve (computed form the model) and estimated curve (estimated from data) by pressing both buttons sequentially.
13) Once the LQG curves have been obtained, you can load a set of closed-loop routine operating data from the **Evaluation** menu to assess its performance. Then, press the **Assess Perf** button to have all the performance indices calculated. The overall performance index and an index for each controlled variable will be shown in the **Minimum Variance Performance Index** panel. Three more indices are available in the **Other Performance Indices** panel, see Figure 7. For the definition of the different indices, please see the Theory section on p. 9.
14) To clear the GUI, go the **File** menu and select **New**.
15) To save the data currently in the GUI, go the **File** menu and select **Save**.

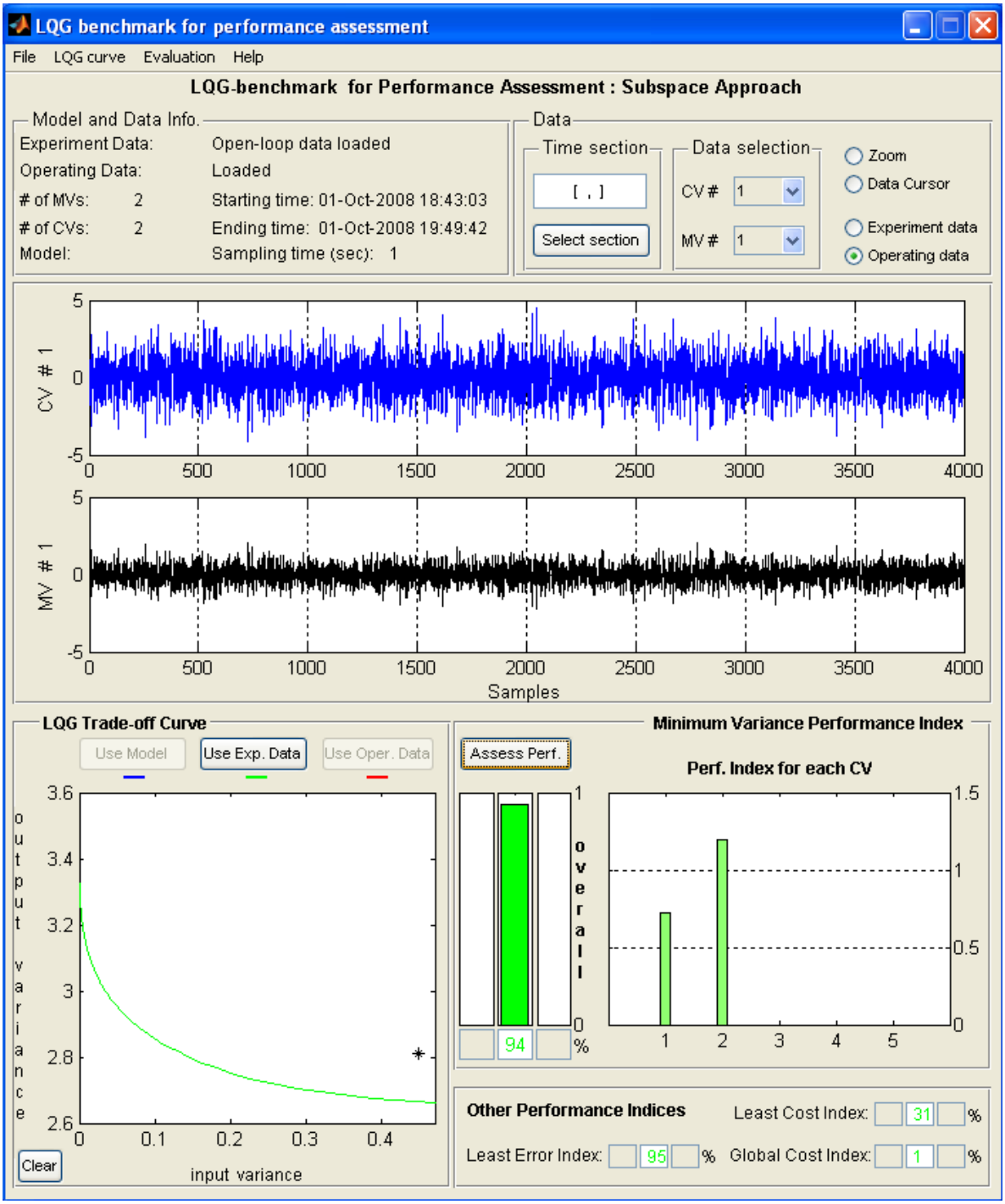

Figure 7: Performance indices obtained for a MIMO process

Detailed Instructions

Theory

The optimal linear-quadratic Guassian (LQG) controller can be used as a benchmark for performance assessment. The question in performance assessment using LQG-benchmark is how far away the current controller"s performance is from the best achievable performance with the same control effort. In other words, the following problem needs to be solved.

Assume that $E\left[u_t^2\right] < \alpha$, what is the minimum value of $E\left[u_t^2\right]$. The solution to this problem is given by the trade-off curve, which can be obtained by solving the LQG problem

$$J(\lambda) = E[y_t^2] + \lambda\, E[u_t^2] \tag{1}$$

for different values of λ. In this way, different solutions for $E[u_t^2]$ and $E[y_t^2]$ can be calculated and then a curve with the optimal $E[y_t^2]$ on the y-axis and $E[u_t^2]$ on the u-axis. Any linear controller can only perform in the region above this curve. In other words, this curve displays the minimal achievable variance of the controlled variable as a function of the variance of the manipulated variable. Performance of any controller can be evaluated comparing current input and output variances with the trade-off curve.

A sample LQG trade-off curve is shown in Figure 8. In this figure, the current working point of the control system is also shown. With this curve and the current input and output variances, it is possible to define some performance indices that compare the performance of current controller to the optimal LQG controller (Huang & Kadali, 2008; Huang & Shah, 1999). A minimum variance control (MVC) benchmark can also be found from this curve. This occurs when $\lambda \rightarrow 0$. Consider that the actual input variance is given as V_u and the output variance as V_y; the optimal output variance corresponding to V_u is given by V_u^0; and the optimal input variance corresponding to V_y is defined as V_y^0. Define the following performance indices for output and input variances:

$$\eta = \frac{V_y^0}{V_y}$$
$$E = \frac{V_u^0}{V_u} \tag{2}$$

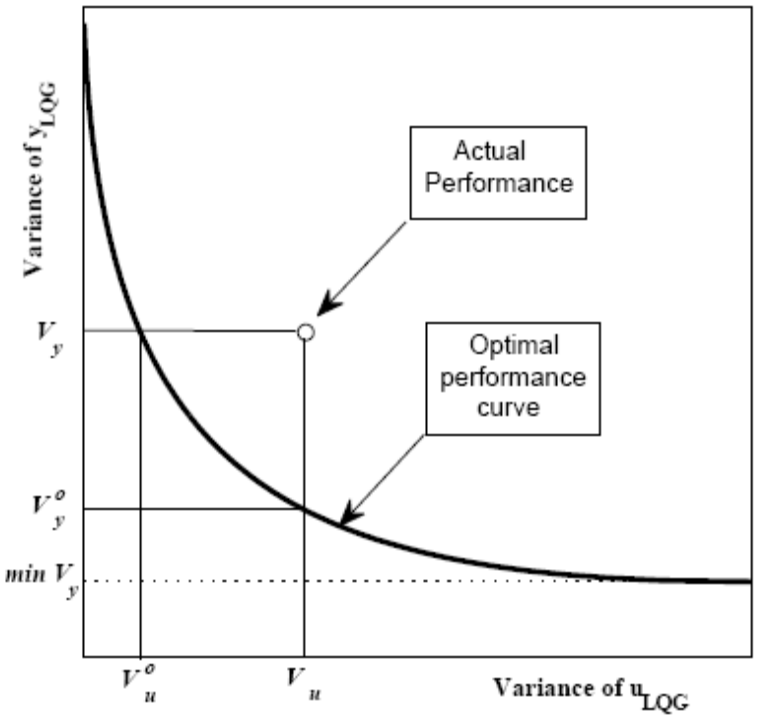

Figure 8: An example LQG trade-off curve

In contrast to the minimum variance control (MVC) benchmark, this approach focuses on both variances of input and output and provides a trade-off curve which represents a feasible range of performance for linear controllers (Huang & Kadali, 2008; Huang & Shah, 1999). Other indices can be obtained from this method. A detailed discussion of the different indices is presented in (Huang, 2003). Similar to the MVC benchmark, this method is based on a process model, which can be obtained using process identification.

In (Huang & Kadali, 2008), a method for calculating the LQG trade-off curve using the subspace framework based on the LQG design given in (Favoreel W. , De Moor, Gevers, & Van Overschee, 1999; Favoreel W. , De Moor, Gevers, & Van Overschee, 1998) was presented. A closed-loop identification method for the estimation of the required open-loop subspace matrices from closed-loop data is also provided in (Kadali & Huang, 2002). An improved version of this identification method was recently proposed in (Danesh Pour, Huang, & Shah, 2009), which improves the consistency of noise variance estimation. This work also presents the method of disturbance model estimation using an available process model and a set of closed-loop routine operating data (no identification experiments required). This GUI is based on the work proposed in (Danesh Pour, Huang, & Shah, 2009).

In order to understand how LQG curve is calculated, assume that the following system is known:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + Ke_t \\ y_t &= Cx_t + Du_t + e_t \end{aligned} \tag{3}$$

In matrix form, Equation (3) can be written as

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ CA^{N-2}D & & \cdots & & D \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} + \begin{pmatrix} I \\ CK \\ \vdots \\ CA^{N-2}K \end{pmatrix} e_0 \tag{4}$$

$$y_{0|N-1} = L_u u_{0|N-1} + L_{e,1} e_0$$

Consider a regulatory LQG control objective function defined as

$$J = \sum_{i=0}^{N-1} \left(y_i^T y_i + u_i^T (\lambda I) u_i \right) \tag{5}$$

Minimizing J will yield following solutions

$$\begin{aligned} u_{0|N-1}^{opt} &= -\left(L_u^T L_u + \lambda I \right)^{-1} L_u^T L_{e,1} e_0 \\ y_{0|N-1}^{opt} &= \left(I - L_u \left(L_u^T L_u + \lambda I \right)^{-1} L_u^T \right) L_{e,1} e_0 \end{aligned} \tag{6}$$

Now, define the following two matrices

$$\begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{N-1} \end{pmatrix} = -\left(L_u^T L_u + \lambda I \right)^{-1} L_u^T L_{e,1} \tag{7}$$

$$\begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{N-1} \end{pmatrix} = \left(I - L_u \left(L_u^T L_u + \lambda I \right)^{-1} L_u^T \right) L_{e,1} \tag{8}$$

It can be shown that all the points for LQG curve can be obtained using following formula

$$\begin{aligned} Cov[u_t] &= \sum_{i=0}^{N-1} \psi_i c\,\mathrm{ov}[e_t] \psi_i^T \\ Cov[y_t] &= \sum_{i=0}^{N-1} \gamma_i c\,\mathrm{ov}[e_t] \gamma_i^T \end{aligned} \tag{9}$$

To calculate LQG curve using the data, L_e and L_u must be calculated first and then the remaining parameters can be estimated.

Data Storage

The file **gen_cmpct_data_LQG.p** can be used to generate data or model objects and save them for use with the GUI. Model and data formats generated by this code are as follows.

Model Storage Format

The model storage file contains information about the multivariate model that is being used to describe the system. It must be entered as a transfer function object (tf) or state space object (ss). The transfer function matrix, for each of the process or disturbance models, can be created using the following steps:

1) Assume that a process with n inputs and m outputs is being analyzed. The transfer function between the j^{th} input and the it^{h} output is a rational function of either s or z, given as Aij / Bij, where A is the numerator and B is the denominator , and both are vector containing in descending order of powers of s or z the corresponding co-efficients of A and B, respectively.
2) It is now necessary to create 2 cell arrays that contain all the information about the process. The first cell array, **A** , is created as follows:
   The first row contains the transfer function between the first output and each of the inputs, while the second row contains the transfer function between the second output and each of the inputs. It should be noted that the curly brackets, {}, are used to create a cell array. In order to access, the (i, j) entry of the array, the command would be $\mathbf{A}\{i, j\}$. A similar cell array containing the values of the denominator, denoted as **B** , is also created.
3) The time delay for the model can be created by forming the matrix **D** , such that the (i, j) entry contains the time delay associated with the transfer function for the ith output and jth input. The array simply contains the numeric values.
4) The continuous transfer object can then be created using the following MATLAB command: **"`>>Atf=tf(A,B,"ioDelay",D)`"**, where **A** is the cell array **A** , **B** is the cell array **B** , and **D** is the time-delay matrix, **D** . To create a discrete time transfer function, the MATLAB command would be **"`>>Atf=tf(A,B,Ts,"ioDelay",D)`"**,

where **Ts** is the sampling time. If it is desired to create a discrete model that contains powers of "z^{-1}", then the command would be

"`>>Atf=tf(A,B,Ts,"ioDelay",D,"Variable","z^-1")`"

5) Creating a MIMO state space model can be easily done by providing A,B,C,D,K matrices to the MATLAB function **"`>>sys=ss(A,B,C,D,Ts)`"**. For continuous-time model, **Ts** should be left blank.

Data Storage Format

The data format is called *Compact* format. Assume that there are ***p*** samples of output, input and setpoint data with a total of ***t*** controlled variables and ***s*** 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 *p*-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 *p*-by-1 cell matrix that contains the time stamp for each of the samples.
4) **dbl_Compact_Data**: This is a $(2p+t)$-by-q double matrix (for CL experiment data) or a $(p+t)$-by-q double matrix (for OL experiment data and CL operating data) that contains the values for each of the tags and sample periods.
5) **dbl_SamplingTime**: This is a scalar double that contains the sampling time for the process.
6) **int_CVNumber**: This is a scalar integer that contains the number of controlled variables in the process, that is, *t*.
7) **int_MVNumber**: This is a scalar integer that contains the number of manipulated variables in the process, that is, *s*.
8) **status**: This is a *p*-by-$(s + t)$ 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 and Model Generation

The data and model storage files can be generated using **gen_cmpct_data_LQG.p**. Based on the step-by-step comments after running the code, you would be able to create your model or data file in a *Compact* format. To generate a complete model (process + disturbance), you have two choices:

1) A `tf`-object as the process model and a `tf`-object as the disturbance model must be ready in the workspace.
2) State-space matrices `A`, `B`, `C`, `D`, and `K` must be ready in the workspace.

Note that you can only use one of these two types of model to generate a *Compact* model. For process model (only process model, no disturbance model),

1) It can be provided by the state-space matrices `A`, `B`, `C`, and `D`.
2) It can be provided as one `tf`-object.

Note that a set of closed-loop routine operating data is required with this choice of model. A set of **closed-loop** or **open-loop** experiment data (identification data) must be available in the workspace, if you want to estimate the LQG curve from experiment data.

To create the closed-loop **operating** data in *Compact* format (for **evaluation** purposes), you have to get your data ready in the workspace and run the code. By running the code once, a single model can be generated and saved, a single set of experiment data, or a single set of operating data. Depending on the choices selected, some of the steps below can be skipped.

Run the code:

If the data and/or model information are ready in the workspace, run the code. The followings will appear on the command window:

```
*********************************************************
Consider a process with m inputs , p outputs and d
disturbances:
The following variables should be available in WORKSPACE:

 1. LQG CURVE REQUIREMENTS:
     - process and disturbance model in the form of
       p x m and p x d transfer function matrices (tf-object)
       And White noise covariance matrix
```

```
 OR
     - process and disturbance model in the form of
       a State-Space model: A,B,C,D,K matrices
       And White noise covariance matrix
 OR
     - only process model
       in SS format (A,B,C,D) of TF format (p x m tf-object)
       And
       CL Operating data; output and Input data (y,u):  N
x(p+m) matrix

 AND / OR

     - CL Excitation data; Output, Input and Setpoint data
(y,u,r): N x (p+m+p) matrix

 AND / OR
     - OL Excitation data; Output and Input data (y,u): N
x(p+m) matrix

 2. PERFOMANCE EVALUATION REQUIRMENT
     - CL Operating data; output and Input data (y,u):  N
x(p+m) matrix

 3. Sampling time

*******************************************************
Press ENTER to continue or type X to exit :
```

Press ENTER to go further. **Step 1** is to provide a complete model (process + disturbance), if applicable. You will see the first step in the command window:

```
1. Provide process and disturbance models....

press ENTER to provide TF model or type X to go to the next
part :
```

Press ENTER, and the following will appear:

```
process model transfer function matrix (tf-object):
```

Give the name that you have defined for your **process** model tf-object, for example, `Gp`, and press enter. The following will appear:

```
disturbance model transfer function matrix (tf-object):
```

Provide the name that you have defined for your **disturbance** model tf-object, for example, `Gd`, and press enter. The following will appear:

```
noise covariance matrix:
```

Enter the input noise covariance matrix. If this step was skipped by typing X, a state-space model will be required

```
press ENTER to provide SS model or type X to go to the next
part :
```

Press ENTER and give the following state-space matrices:

```
state space A matrix:
state space B matrix:
state space C matrix:
state space D matrix:
state space K matrix:
noise covariance matrix:
```

If this step was skipped, then the process model without a no noise model is entered as follows (Step 2):

```
2. Provide process model and CL operating data....
press ENTER to provide TF model or type X to go to the next
part:
```

Press ENTER and the following will appear:

```
process model transfer function matrix (tf-object):
```

By typing X, the following will appear:

```
press ENTER to provide SS model or type X to go to the next
part :
```

If you choose to provide a state-space model, the code will ask for the matrices:

```
state space A matrix:
state space B matrix:
state space C matrix:
state space D matrix:
```

Closed-loop operating data should then be provided:

```
Provide Closed-loop operating data matrix [y u]:
```

Irrespective of whether Steps 1 and 2 are skipped, Step 3 asks for the closed-loop excitation data:

```
3. Provide Closed-loop excitation data matrix [y u r],
Or press ENTER, if you want to skip this part,
```

Give the data arranged in the given order: output, input and setpoint. Keep the []. If ENTER is pressed without providing the closed-loop experiment data, open-loop experiment data can be provided in Step 4:

```
4. Provide Open-loop excitation data matrix [y u],
Or press ENTER, if you want to skip this part,
```

If you want to generate and save the closed-loop operating data, the next command (**Step 5**) allows you to do that:

```
5. Provide Closed-loop operating data matrix [y u]:
```

Note that you cannot generate both experiment and operating data in *Compact* format by running the code once. You will be asked about the number of process outputs for any of these set of data that you provided (if you have not provided any model).

Sampling time will be asked at this stage:

```
Sampling time:
```

Names of the files for saving the *Compact* model or *Compact* data are requested at the last step:

```
The file name to save compact data (e.g. test_cmpct_data):
The file name to save transfer function model (e.g. tf_model):
The file name to save state space model (e.g. ss_model):
The file name to save mpc model (e.g. mpc_model):
```

The gen_cmpct_data_LQG code, will save some .mat files in the current active path of MATLAB which you can be loaded and used in the GUI for analysis.

NOTE: Any files with the same name will be overridden.

Using the Toolbox

Installation

The toolbox can be installed by unzipping the files to the 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_LQGPA”.**

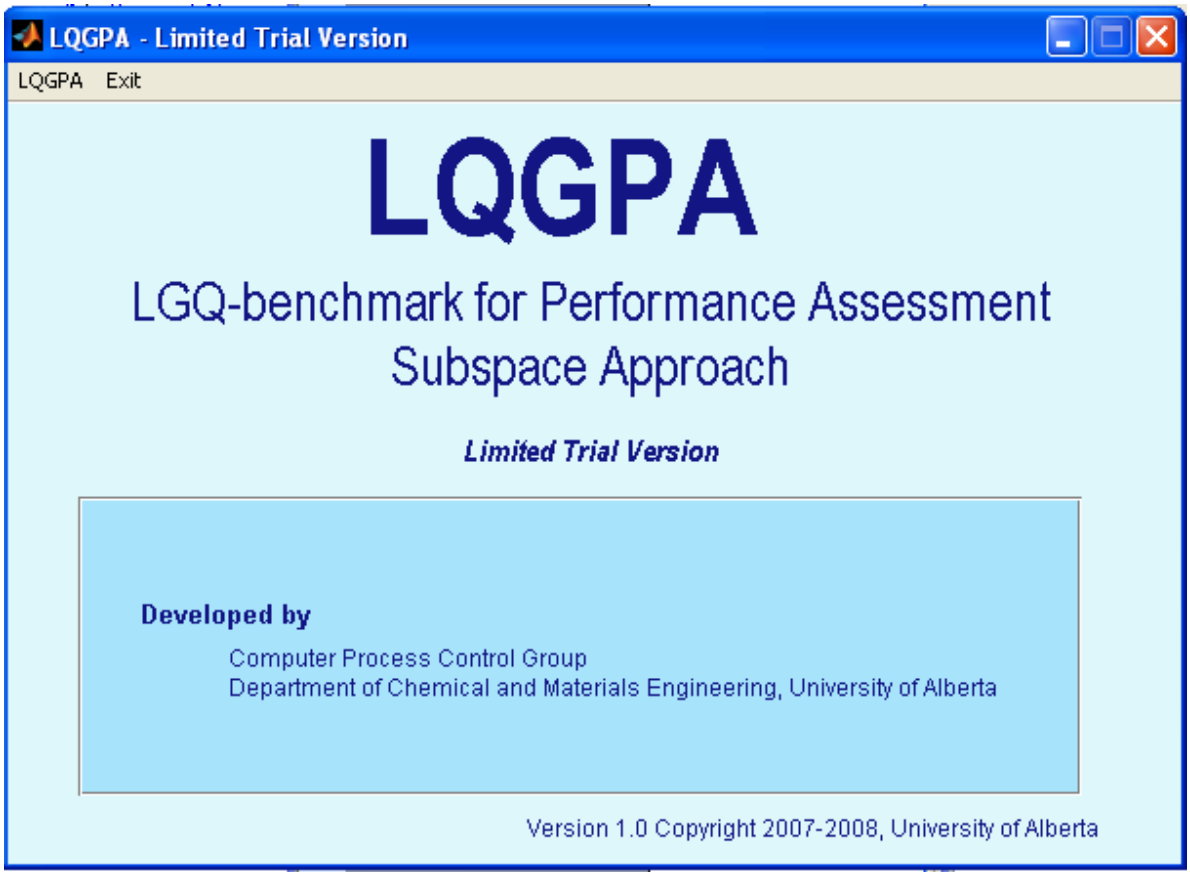

Figure 9: The first GUI that appears

The GUI shown in Figure 9 should appear. This GUI is the main access to the toolbox. To start a session of the toolbox, click the **LQGPA** menu. This will bring up a new GUI, which is shown in Figure 10. In Figure 10, each of the main parts of the GUI is separated by panels and will be discussed separately.

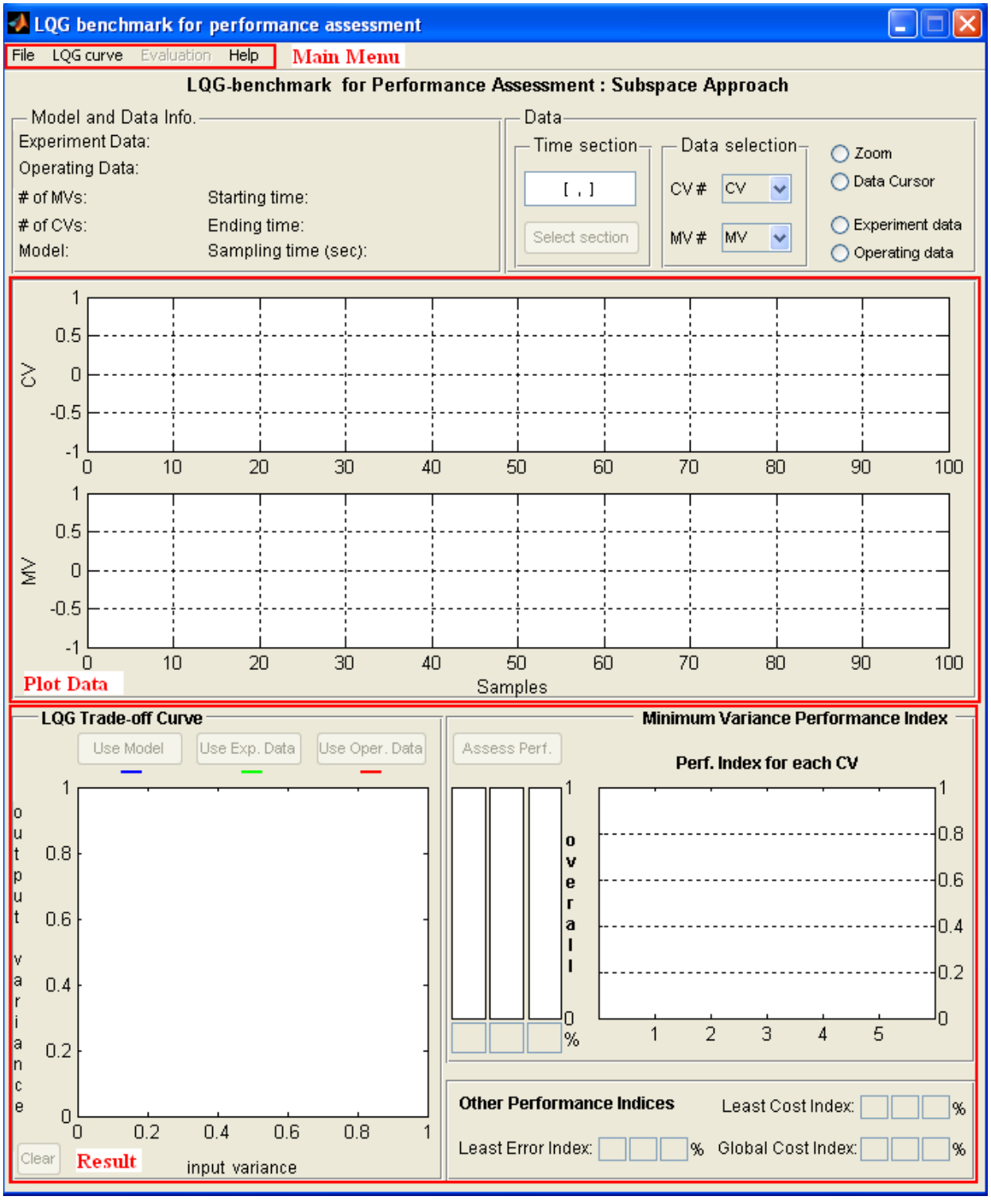

Figure 10: The main GUI for this toolbox

In-depth Discussion of the Toolbox

Section 1: Main Menu

The **Main Menu** consists of the following 3 parts:

1) **File:** Clicking this menu will allow user to choose between 3 options:
   a. **New:** Clear all the data from the current GUI and restart the analysis from a clean layout.
   b. **Open:** This option allows the user to open a previously-saved case. Everything will be shown as it was before.
   c. **Save:** This option allows the user to save the current case. All the models, data, and results will be saved in a MATLAB file.
2) **LQGcurve:** Clicking this menu will allow the user to select different options for closed-loop or open-loop data or models (generated by "gen_cmpct_data_LQG.p") that will be used in the analysis. Four choices are available:

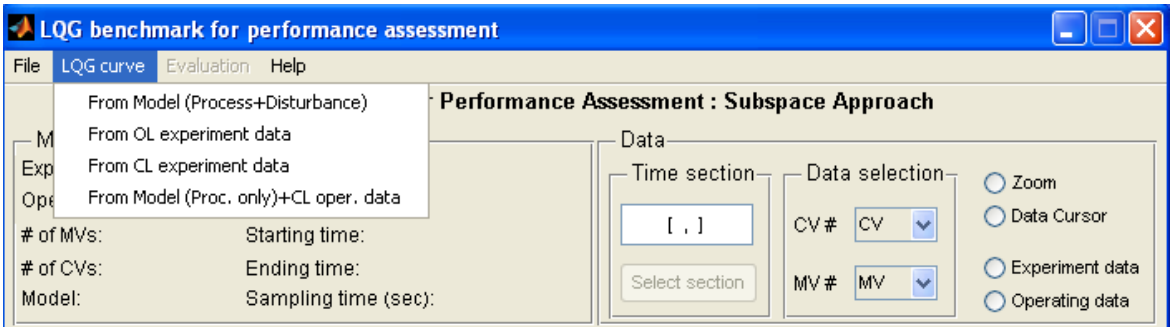

Figure 11: The "LQGcurve" menu from the "Main" menu

   a. **From Model (Process + Disturbance):** This option allows the user to load a complete model of the process which contains both process and disturbance models to be used for the calculation of the LQG curve.
   b. **From OL experiment data:** This assumes that the data to be used for estimation of the curve is a set of open-loop experiment (identification) data stored in the *Compact* format.
   c. **From CL experiment data:** This assumes that the data to be used for estimation of the curve is a set of closed-loop experiment (identification) data stored in the *Compact* format.

d. **From Model (Proc. only)+CL oper. data:** This loads a process model and a set of closed-loop operating data (in one file). Note that when you load this file, the process model and operating data will be used to calculate the curve, so for the evaluation purpose you may want to load another set of operating data (which can be the same set) from the **Evaluation** menu.

3) **Evaluation:** Clicking this menu will allow the user to select and load a set of closed-loop routine operating data (generated by "gen_cmpct_data_LQG.p") for evaluating the performance of current controller. Only one choice is available:

   a. **Load CL operating data:** This allows the user to load a set of closed-loop routine operating data.

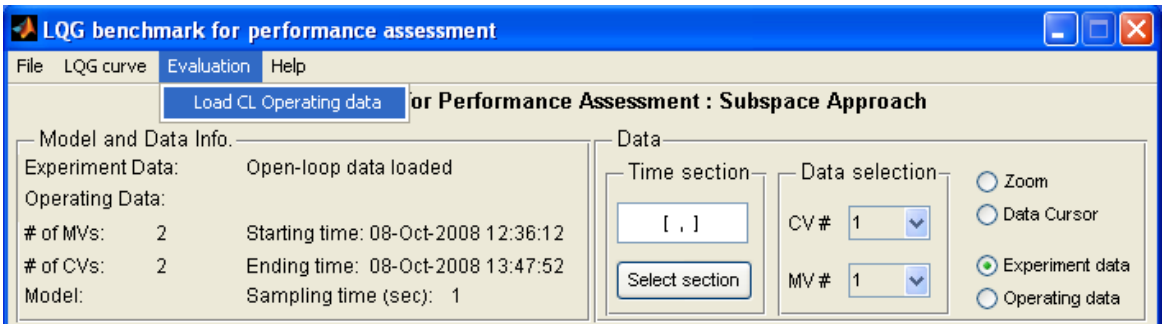

Figure 12: The "Evaluation" menu from the "Main" menu

Section 2: Model and Data Info panel

This panel gives the information about the type of model you have loaded as well as experiment data and operating data, including the number of controller variables (CV), the number of manipulated variables (MV), sampling time, starting time, and ending time.

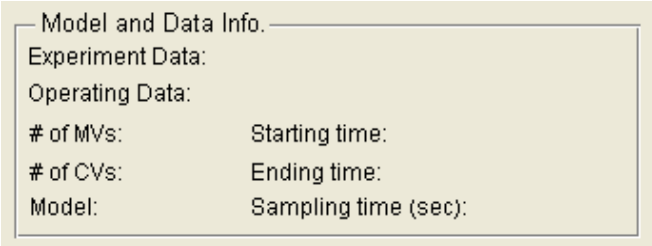

Figure 13: The "Model and Data Info." panel

Section 3: Data panel

There are some tools in this panel which can be used for data selection and display as follows:

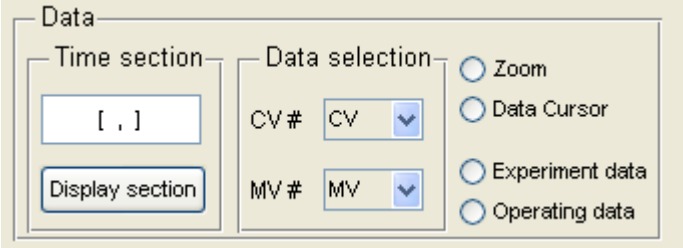

Figure 14: The "Data" panel

1) **Time section:** In this field, the user can specify what part of the total data is to be used in the analysis. The format for this entry is "[start sample value, end sample value]". A comma must separate the 2 values and the end value must be greater than the start value. As well, there must a sufficient number of data samples in order for the computer to estimate a model. Clicking "Select section" will display the selected samples in the "plot data" section.
2) **Data selection:** This subpanel enables you to choose the controlled variable (CV) and manipulated variable (MV) that you want to show in the "plot data" section.
3) **Zoom:** This allows the user to zoom into a certain section of the graph. However, there is no zoom-out button, so you have to double click the left mouse button to get back to the original zoom.
4) **Data Cursor:** This can be used to monitor the exact value of any data points.
5) **Experiment/Operating data:** By choosing one of these two radio buttons, you can choose to show either experiment data or operating data in the "plot data" section.

Section 4: Plot data

Two axes have been provided in this section which allows the user to view the MV and CV data. Each one can only show one set of data, so you should use the **Data selection** subpanel in the **Data** panel to choose the number of CV or MV you want to see.

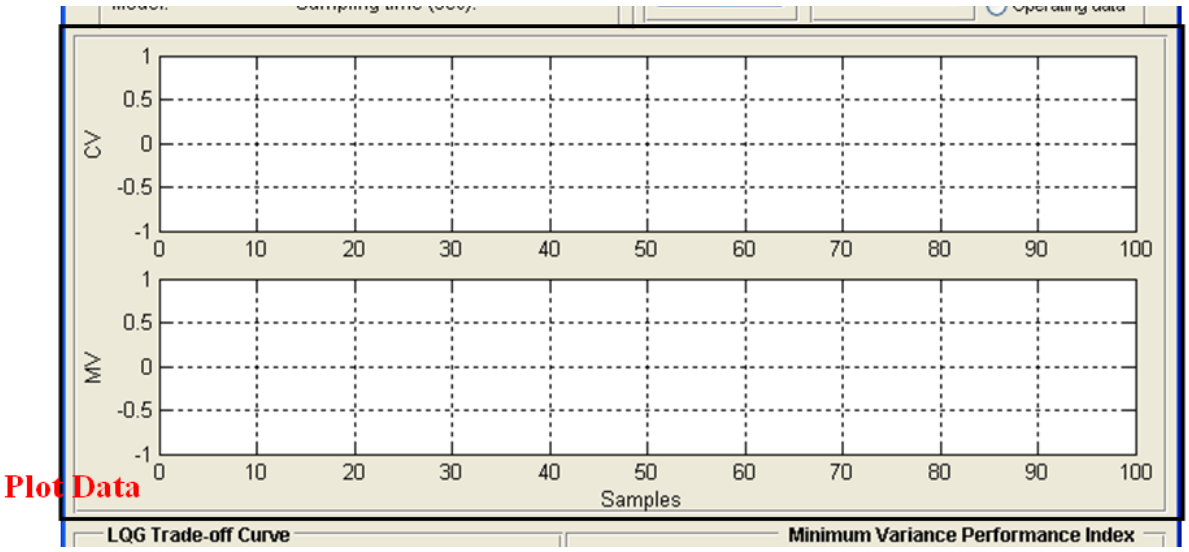

Figure 15: The panel being used for plotting data

Section 5: LQG Trade-off Curve panel

In this panel, the LQG trade-off curve calculated using the model, estimated from data, or both is plotted on the provided graph. Four push buttons are available in this panel:

1) **Use Model:** This button is enabled after you loaded a complete model (Process + Disturbance) in the GUI. Press it to calculate and plot the trade-off curve (in blue).

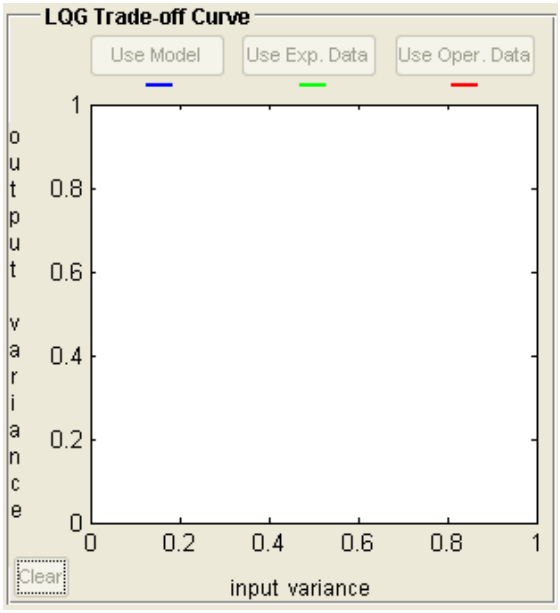

Figure 16: "LGQ Trade-off Curve" panel

2) **Use Exp. Data:** This button is enabled, if you have provided a set of closed-loop or open-loop experiment data to estimate and plot the trade-off curve (in green).

3) **Use Oper. Data:** This button is enabled, if you have provided process model and a set of closed-loop operating data to estimate and plot the trade-off curve (in red).
4) **Clear:** You may use this button to clear the graph.

Section 6: Minimum Variance Performance Index panel

In this panel, the Minimum Variance Control (MVC) benchmark (obtained from the trade-off curve) is used to calculate the MVC performance index. The MVC benchmark calculated automatically takes all the nonminimum phase zeros into account. Further information about the MVC benchmark for nonminimum phase systems can be found in (Huang & Shah, 1999). There are two axes in this panel: one for the overall MVC performance index and the other one for individual performance indices for each CV.

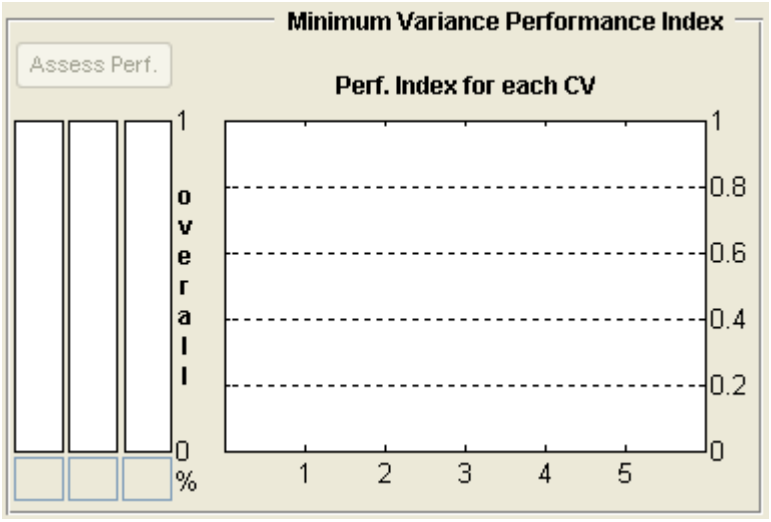

Figure 17: "Minimum Variance Performance Index" panel

As seen, one corresponding push button is provided. Press this button after a trade-off curve has been calculated/estimated and a set of closed-loop operating data is also loaded in to see all the performance indices for the process. All the calculation can be determined using the 3 different curves. Hence, 3 different answers with 3 colors are demonstrated in this panel.

Section 7: Other Performance Indices panel

Three other performance indices can also be found from the LQG trade-off curve which are achievable in this panel. Note that based on the position of actual variance point in relation to the trade-off curve, some of these indices might not be calculable.

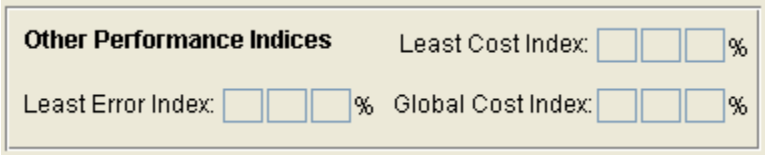

Figure 18: "Other Performance Indices" panel

Example 1

A set of models and data generated by "gen_cmpct_data_LQG" are provided with this toolbox which comes from the following system:

$$x_{t+1} = \begin{pmatrix} 0.6 & 0.6 & 0 \\ -0.6 & 0.6 & 0 \\ 0 & 0 & 0.7 \end{pmatrix} x_t + \begin{pmatrix} 1.6161 \\ -0.3481 \\ 2.6319 \end{pmatrix} u_t + \begin{pmatrix} -1.1472 \\ -1.5204 \\ -3.1993 \end{pmatrix} e_t$$
$$y_t = \begin{pmatrix} -0.4373 & -0.5046 & 0.0936 \end{pmatrix} x_t - 0.7759 u_t + e_t \tag{10}$$

A PI controller,

$$G_c = 0.1 + \frac{0.05}{s} \tag{11}$$

is used for this process. A simple Simulink model has been prepared to collect closed-loop experimental data (see Figure 19). The identification test signal, $\boldsymbol{r}(\boldsymbol{t})$, is designed by "*idinput"* command:

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

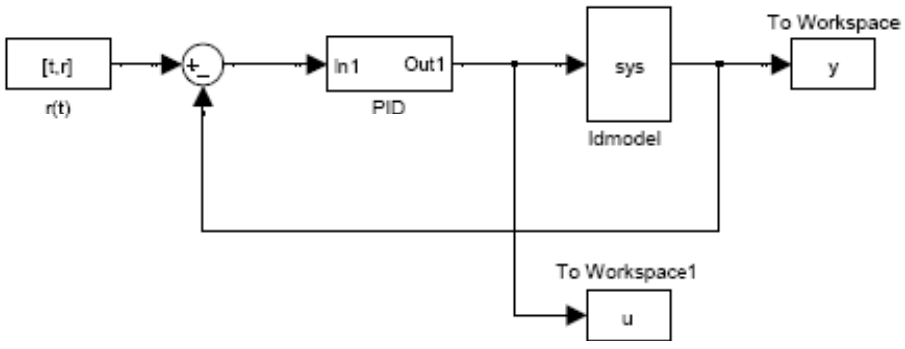

Figure 19: Simulink model for collecting closed-loop data

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. To get a set of routine operating data $r(t)$ must be replaced by a zero column vector of length 6001. A similar model has been used to generate open-loop experiment data.

When model and data variables are present in the workspace, run "gen_cmpct_data_LQG" to create and save model and data files in the *Compact* format. All the files related to this example have names starting with **SISO_**. After loading model and data files in the GUI, the LQG trade-off curve can be estimated and various performance indices can be calculated. The final results are shown in Figure 20, which shows relatively poor performance of the controller in all aspects.

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:

$$G_p = \begin{pmatrix} \dfrac{z^{-1}}{1-0.4z^{-1}} & \dfrac{0.5z^{-2}}{1-0.1z^{-1}} \\ \dfrac{0.3z^{-1}}{1-0.4z^{-1}} & \dfrac{z^{-2}}{1-0.8z^{-1}} \end{pmatrix}$$
$$G_l = \begin{pmatrix} \dfrac{1}{1-0.5z^{-1}} & \dfrac{-0.6z^{-1}}{1-0.5z^{-1}} \\ \dfrac{0.5z^{-1}}{1-0.5z^{-1}} & \dfrac{1}{1-0.5z^{-1}} \end{pmatrix} \qquad (12)$$

The following controller is implemented:

$$G_c = \begin{pmatrix} \dfrac{0.5-0.2z^{-1}}{1-0.5z^{-1}} & 0 \\ 0 & \dfrac{0.25-0.2z^{-1}}{(1-0.5z^{-1})(1+0.5z^{-1})} \end{pmatrix} \qquad (13)$$

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
```

Use a Simulink model similar to Example 1 to generate the data. The same procedure as the previous example is followed to create and save a *Compact* model and data files, load it into the GUI, and obtain the trade-off curve as well as performance indices. All the files related to this example have names started with **MIMO_**. Results are shown in Figure 21 which indicates that the overall control system is working similar to the minimum variance controller, while the very small value of the Global Cost index indicates that the control action could be reduced significantly for the same output variance.

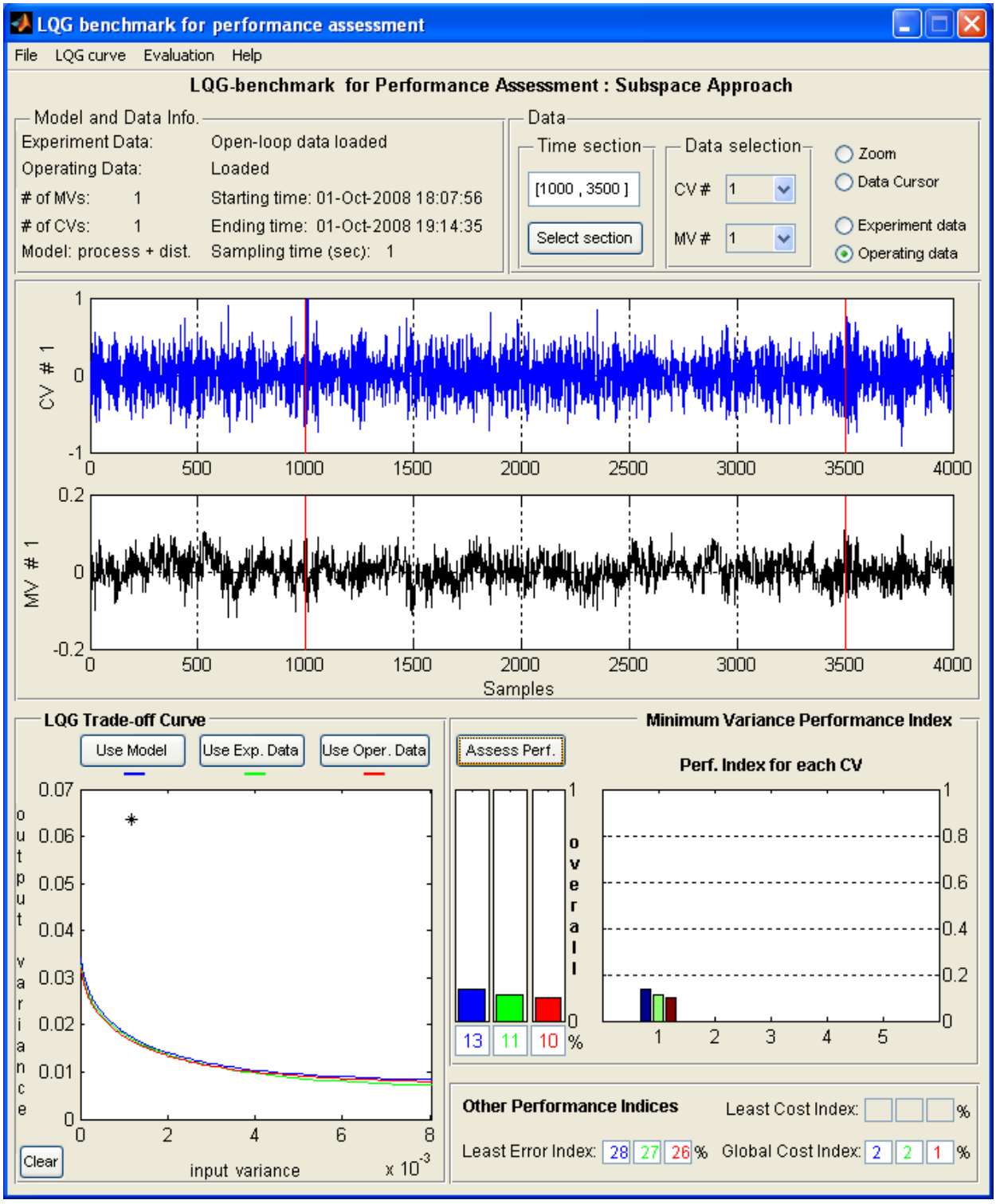

Figure 20: Results of performance assessment for Example 1

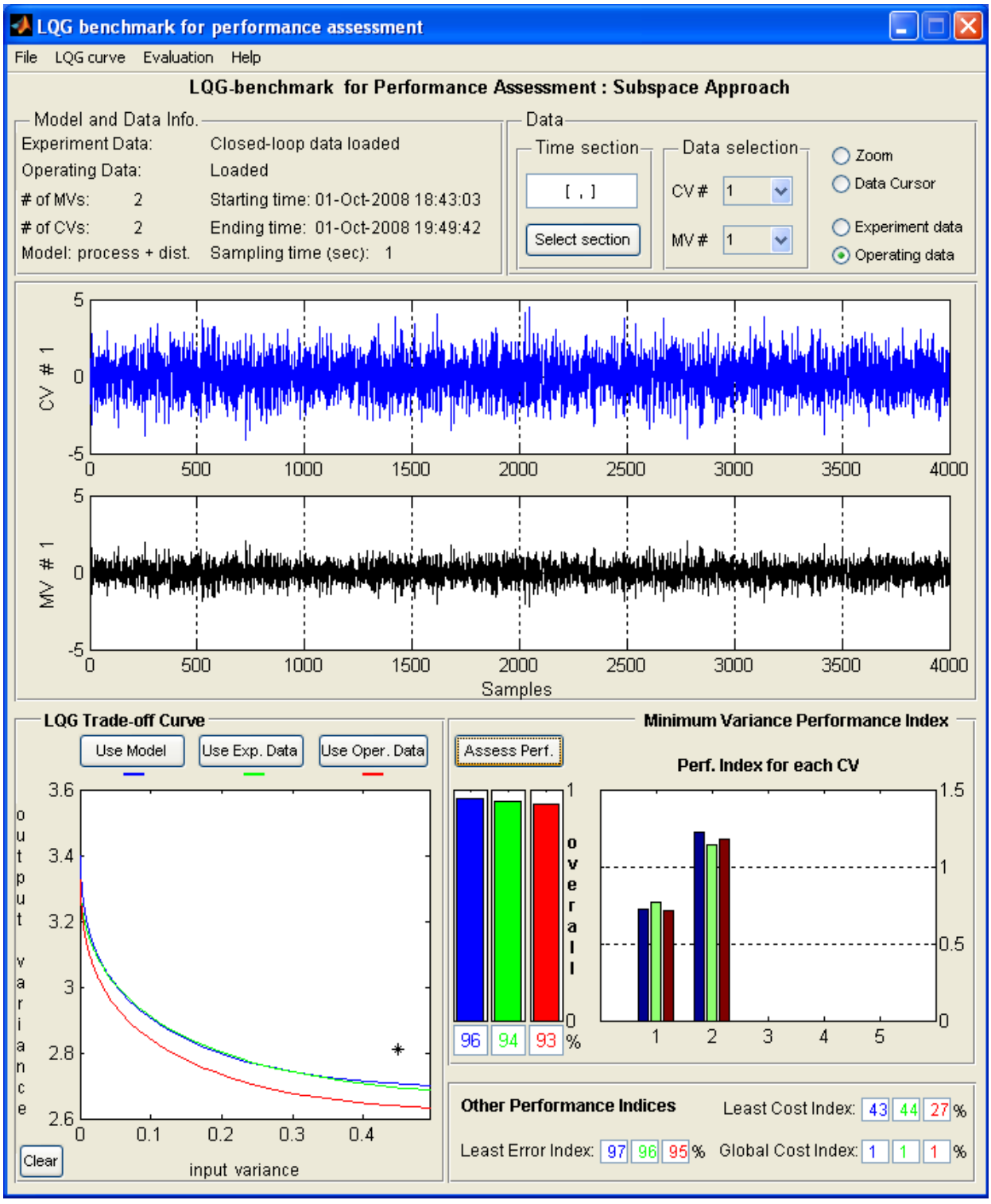

Figure 21: Results of performance assessment for Example 2

References

Danesh Pour, N., Huang, B., & Shah, S. L. (2008). Closed-loop subspace identification for performance assessment. *Journal of Process Control* .

Danesh Pour, N., Huang, B., & Shah, S. L. (2009). Consistency of noise covariance estimation in joint input–output closed-loop subspace identification with application in LQG benchmarking. *Journal of Process Control , 19* (10), 1649-1657.

Favoreel, W., De Moor, B., Gevers, M., & Van Overschee, P. (1999). Closed-loop model-free subspace-based LQG-design. *Proceedings of the 7th Mediterranean Conference on Control and Automation.* Haifa, Israel.

Favoreel, W., De Moor, B., Gevers, M., & Van Overschee, P. (1998). Model-free subspace-based LQG-design. *1999 American Control Conference*, (pp. 3372-3376). San Diego, California, United States of America.

Huang, B. (2003). A pragmatic approach towards assessment of control loop performance. *International Journal of Adaptive Control and Signal Processing , 17* (7-9), 589-608.

Huang, B., & Kadali, R. (2008). *Dynamic Modeling, Predictive Control, and Performance Monitoring.* London, England, United Kingdom: Springer-Verlag.

Huang, B., & Shah, S. L. (1999). *Performance Assessment of Control Loops: Theory and Applications.* Berlin: Springer.

Kadali, R., & Huang, B. (2002). Estimation of the Dynamic Matrix and Noise Model for Model Predictive Control Using Closed-Loop Data. *Industrial & Engineering Chemistry Research , 41* (4), 842-852.