University of Alberta Computer Process Control Group

LMIPA

Limited Trial Version

Written by CPC Control Group, University of Alberta

Version 4.0

Table of Contents

List of Figures

List of Tables

Introduction

The LMIPA toolbox was developed by the Computer Process Control Group at the University of Alberta to allow linear-matrix-inequality-based economic assessment of controller performance using MATLAB.

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

System Requirements

1) MATLAB. The programme has been tested on Matlab 7.11.0 (R2010b) and lower versions.
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) In order to use the software, point the current directory to the location of the unzipped files.
3) At the command prompt, type "**`>> main_lmipa`**" to start the toolbox. The GUI shown in Figure 1 should appear.
4) Press the "LMIPA" menu. A new GUI will appear as shown in Figure 2.

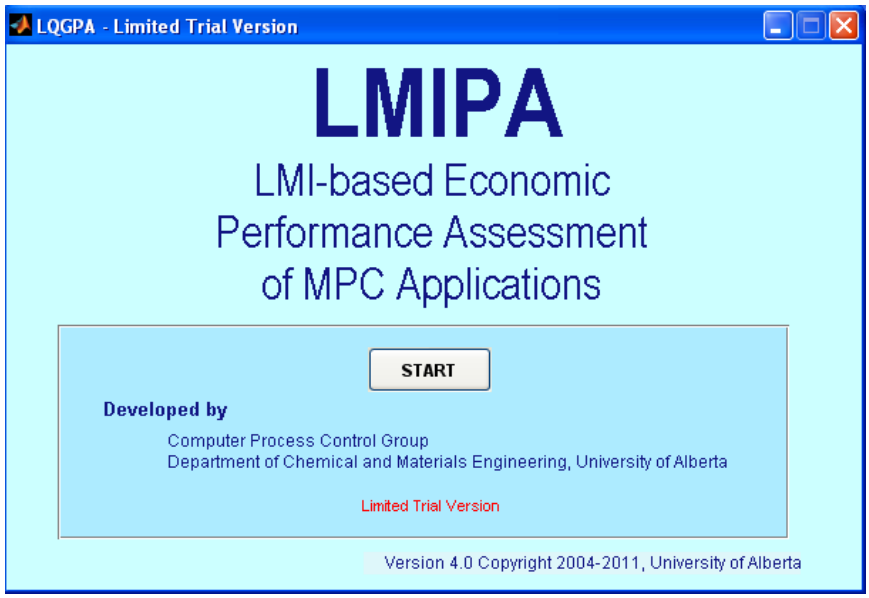


Figure 1: The First GUI that appears

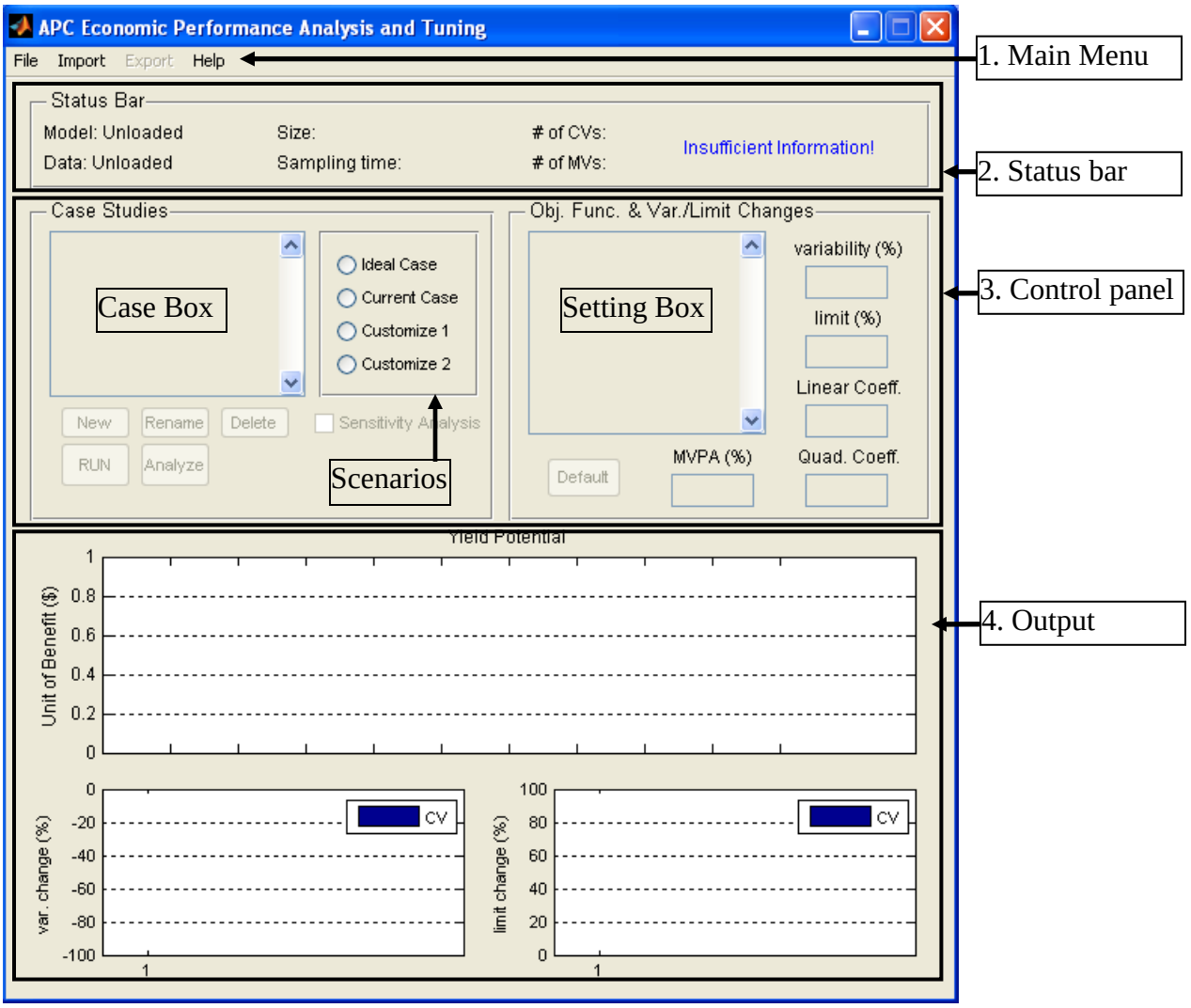


Figure 2: The main GUI for this toolbox, with the main regions highlighted

5) For the purpose of this quick start, we will use the sample data provided in the zip file. Go to the "Import" Menu located in Area 1 in Figure 2 and select "Model". Select the "Case_Model.mat" file. Be patient as the file is loaded. This file contains the open-loop, process transfer function, or the gain matrix. Then go to the "Import" Menu and select "Data." Select the "Case_Cmprhnsv_Data.mat" file. Wait patiently as the file is being loaded.
6) At this moment, sufficient information has been loaded. In the left panel of Area 3, there is a list box, called Case box, which allows us to manage different case studies. Pressing the "New" button will create a new case study. For each case study, 4 different scenarios can exist. The first scenario (Ideal Case) is the **ideal yield potential**, while the second

one (Current Case) is the **current potential**. As each scenario is selected, an explanation in green will appear below the list.

7) Select the Ideal Case and press the “RUN” button.
8) Create a new case by clicking the “New” button. Once done, press the “RUN” button. Figure 3 shows the result.

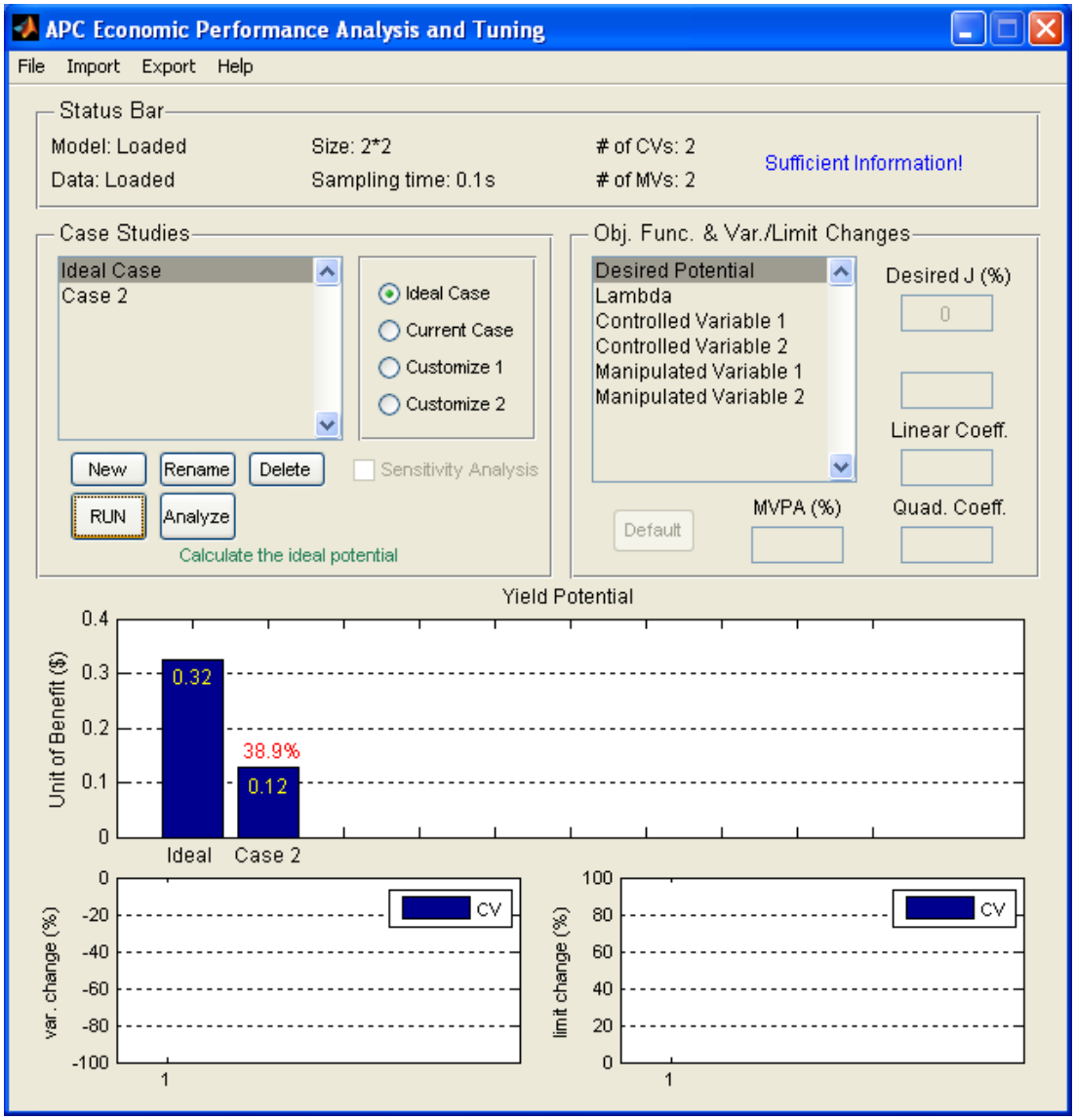


Figure 3: Screen shot after running ideal and current cases

9) As shown in Figure 3, the actual benefit as a fraction of the ideal benefit is displayed in red as a percentage above the corresponding bar in Area 4.
10) Now let us assume we want to find out what variability percentage we should apply to the variables in order to have some desired potential, which corresponds to Customize 2.

Therefore, create a new case study and select Customize 2. In the right panel of Area 3 in the Setting box select “Desired potential.” As well, enter 60% in the textbox “Desired J (%).”

11) Click on “RUN” to run the case. The results are shown in Figure 4. The results show that in order to increase the benefit potential to 60% of the ideal case, the variability in Controller Variable 1, must be decreased by about 34.5%.

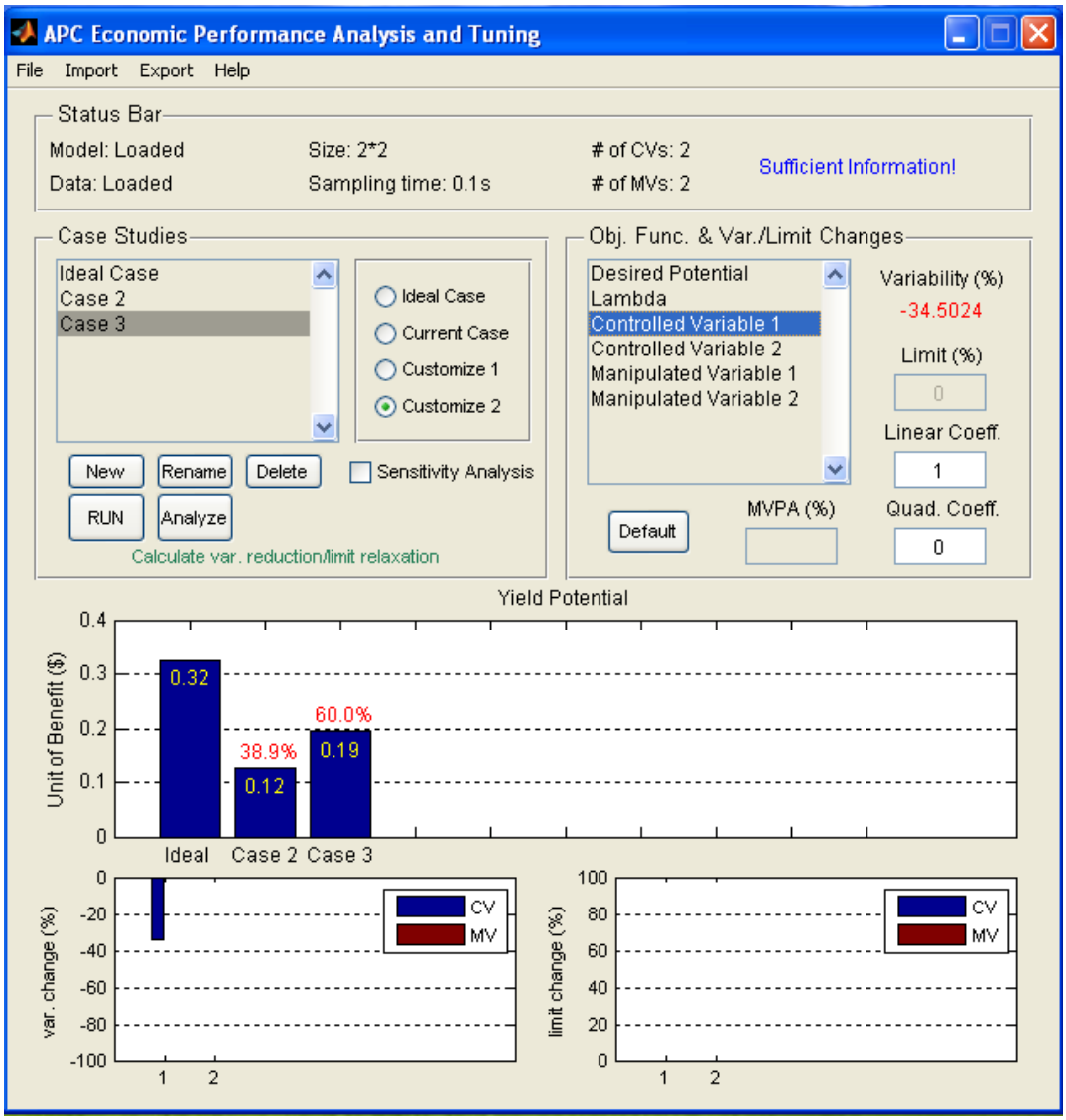


Figure 4: Results obtained after desiring a J 60% of the ideal

8) The results can be saved as either a Matlab or Excel file using the “Export” menu. All the cases that have been run will be saved. The entire project can also be saved a single Matlab file using File->Save (Ctrl + S). This file can be reloaded to continue the analysis

at a later time. After loading, everything will be restored to what it was at the time of saving.

9) The programme can be quit by clicking on File->Exit and then Exit to close the 2 windows that previously appeared.

Detailed Instructions

Theory

Linear Matrix Inequality Performance Analysis (LMIPA) is based on minimising the economic objective function, *J*, that is (Lee, Huang, & Tamayo, 2008),

$$\min J \tag{1}$$

where *J* is defined as

$$J = \frac{1}{N_L}\sum_{k=1}^{N_L} J(k) \tag{2}$$

N_L is the total number of data points collected, and $J(k)$ is defined as

$$J(k) = \sum_{i=1}^{p}\left[b_{y_i}(k)\bar{y}_i + a_{y_i}^2(k)\left(\bar{y}_i - y_{s_i}(k)\right)^2 \right] + \sum_{i=1}^{m}\left[b_{u_i}(k)\bar{u}_i + a_{u_i}^2(k)\left(\bar{u}_j - u_{s_j}(k)\right)^2 \right] \tag{3}$$

a_y is the quadratic co-efficient for the i^{th} control variable (CV), b_y is the linear co-efficient for the i^{th} control variable, a_u is the quadratic co-efficient for the j^{th} manipulated variable (MV), b_u is the linear co-efficient for the j^{th} manipulated variable, y_s is the set point (target) value for the control variable, u_s is the set point (target) value for the manipulated variable, $\bar{y}$ and $\bar{u}$ are the optimal mean operating points of the control and manipulated variables, *p* is the number of control variables, and *m* is the number of manipulated variables. The optimisation is performed subject to the following constraints:

$$\begin{gathered}
\sum_{j=1}^{m} K_{ij}\Delta\bar{u}_j = \Delta\bar{y}_i \qquad \forall i = 1,2,...,p \\
L_{y_i}(k) - \lambda_{y_i}\alpha_{y_i}(k) + 2\sigma_{y_i}\left(1+\nu_i\right) \le \bar{y}_i \le H_{y_i}(k) + \mu_{y_i}\alpha_{y_i}(k) - 2\sigma_{y_i}\left(1+\nu_i\right) \qquad \forall i = 1,2,...,p \\
L_{u_j}(k) - \lambda_{u_j}\alpha_{u_j}(k) + 2\sigma_{u_j} \le \bar{u}_j \le H_{u_j}(k) + \mu_{u_j}\alpha_{u_j}(k) - 2\sigma_{u_j} \qquad \forall j = 1,2,...,m
\end{gathered} \tag{4}$$

where

$$\bar{y}_i = \bar{y}_{0_i} + \Delta\bar{y}_i, \qquad \bar{y}_{0_i} = \frac{1}{N_L}\sum_{k=1}^{N_L} y_{0_i}(k) \qquad \forall i = 1,2,...,p$$
$$\bar{u}_j = \bar{u}_{0_j} + \Delta\bar{u}_j, \qquad \bar{u}_{0_j} = \frac{1}{N_L}\sum_{k=1}^{N_L} u_{0_j}(k) \qquad \forall j = 1,2,...,m \tag{5}$$

$\bar{y}_0$ and $\bar{u}_0$ are the current mean operating points of the control and manipulated variables and *y* and *u* are the routine operating process data, $\Delta\bar{y}_0$ and $\Delta\bar{u}_0$ are the proposed shift in the mean operating conditions, *L*(*k*) is the lower limit (LL) for the given variable, *H*(*k*) is the upper limit for the given variable, α is half the constraint region for the given variable, σ is the standard deviation for the variable, v is the percent change in variability, μ is the percent change in the upper limit, and λ is the percent change in the lower limit. The last 3 variables are set by the user.

Based on the above equations, 3 special cases are distinguished:

1) Base Case Operation, J_0: This is calculated by replacing $\bar{y}_i$ with $\bar{y}_{0_i}$ and $\bar{u}_j$ with $\bar{u}_{0_j}$. This gives the base case potential.
2) Ideal Benefit Potential, ΔJ_I: It is assumed that v_i = -1, $\lambda_{y_i} = \mu_{y_i} = \lambda_{u_j} = \mu_{y_j} = 0$, $\forall i = 1,2,...,p$ and $\forall j = 1,2,...,m$. This gives the benefit, J_I, for the case where there is no process variability. The ideal benefit potential is then found as $\Delta J_I = J_I - J_0$.
3) Current (existing) Benefit Potential, ΔJ_E: It is assumed that v_i = 0, $\lambda_{y_i} = \mu_{y_i} = \lambda_{u_j} = \mu_{y_j} = 0$, $\forall i = 1,2,...,p$ and $\forall j = 1,2,...,m$. It should be noted that if constraints (2.4b) and (2.4c) become unfeasible, then they are replaced as follows. For constraint (2.4b), replace it by $\bar{y}_i = \bar{y}_{0_i}$ or $\Delta\bar{y}_i = 0$, while for constraint (2.4c), replace it by $\bar{u}_j = \bar{u}_{0_j}$ or $\Delta\bar{u}_j = 0$. This gives the benefit, J_E, for the case where there are no changes in either process variability or the constraints. The real benefit potential is then found as $\Delta J_E = J_E - J_0$.

In practice, it is desired to see which variables should be tuned to achieve the given goals. Thus, there is a need to perform sensitivity analysis. The brute force method would be consider each of the variables separately, perturb it, and observe the results. However, this naive method fails for 2 reasons. Firstly, it is assumed that the interaction between the variables is nonexistent. Thus, there is no need to consider groups of variables (cross-terms). In practice, the individual variables can strongly influence each other. Secondly, for a large number of variables, this search

would take too long to perform. Therefore, 2 separate algorithms have been developed that seek to determine the sensitivities.

The first algorithm is used to determine sensitivity analysis for constraint tuning. In this case, the following set of equations is solved (Lee, Huang, & Tamayo, 2008):

$$\min_{\left\{\overline{y}_i,\overline{u}_j,\lambda_{y_i},\lambda_{u_j},\mu_{y_i},\mu_{u_j}\right\}}\left(\sum_{i=1}^{p}\lambda_{y_i}+\mu_{y_i}+\sum_{j=1}^{m}\lambda_{u_j}+\mu_{u_j}\right) \tag{6}$$

subject to

$$\begin{cases}\overline{\lambda}_{y_i}\geq\lambda_{y_i}\geq 0 & flag_{\lambda_{y_i}}=1\\ \lambda_{y_i}=0 & flag_{\lambda_{y_i}}=0\end{cases} \tag{7}$$

$$\begin{cases}\overline{\mu}_{y_i}\geq\mu_{y_i}\geq 0 & flag_{\mu_{y_i}}=1\\ \mu_{y_i}=0 & flag_{\mu_{y_i}}=0\end{cases} \tag{8}$$

$$\begin{cases}\overline{\lambda}_{u_j}\geq\lambda_{u_j}\geq 0 & flag_{\lambda_{u_j}}=1\\ \lambda_{u_j}=0 & flag_{\lambda_{u_j}}=0\end{cases} \tag{9}$$

$$\begin{cases}\overline{\mu}_{u_j}\geq\mu_{u_j}\geq 0 & flag_{\mu_{u_j}}=1\\ \mu_{u_j}=0 & flag_{\mu_{u_j}}=0\end{cases} \tag{10}$$

$$\frac{J_0-J}{\Delta J_I}=R_C \tag{11}$$

$$\begin{aligned}L_{y_i}(k)-\lambda_{y_i}\alpha_{y_i}(k)+2\sigma_{y_i}\leq\overline{y}_i\leq H_{y_i}(k)+\mu_{y_i}\alpha_{y_i}(k)-2\sigma_{y_i}\\ L_{u_j}(k)-\lambda_{u_j}\alpha_{u_j}(k)+2\sigma_{u_j}\leq\overline{u}_j\leq H_{u_j}(k)+\mu_{u_j}\alpha_{u_j}(k)-2\sigma_{u_j}\end{aligned} \tag{12}$$

$$\forall i=1,2,...,p$$
$$\forall j=1,2,...,m$$

where $\overline{\lambda}$ represents the given constant upper bound for the variable λ, $\overline{u}$ represents the given constant upper bound for the variable *u*, *flag* is a binary variable that represents whether or not the given variables bounds can be changed, and R_C is the given target benefit potential ratio which has a value between 0 and 1.

The second algorithm is used to perform sensitivity analysis for variability tuning. In this case, the following set of equations is solved (Lee, Huang, & Tamayo, 2008):

$$\min_{\{\bar{y}_i,\bar{u}_j,v_i\}}\left(\sum_{i=1}^{p}|v_i|\right) \tag{13}$$

subject to

$$\begin{cases}\underline{v}_i \le v_i \le 0 & flag_{v_i}=1\\ v_i = 0 & flag_{v_i}=0\end{cases} \tag{14}$$

$$\frac{J_0 - J}{\Delta J_I} = R_C \tag{15}$$

$$\begin{gathered}L_{y_i}(k)-\lambda_{y_i}\alpha_{y_i}(k)+2\sigma_{y_i}\left(1+v_i\right)\le \bar{y}_i \le H_{y_i}(k)+\mu_{y_i}\alpha_{y_i}(k)-2\sigma_{y_i}\left(1+v_i\right)\\ L_{u_j}(k)-\lambda_{u_j}\alpha_{u_j}(k)+2\sigma_{u_j}\le \bar{u}_j \le H_{u_j}(k)+\mu_{u_j}\alpha_{u_j}(k)-2\sigma_{u_j}\\ \forall i=1,2,...,p\\ \forall j=1,2,...,m\end{gathered} \tag{16}$$

where $\underline{v}$ represents the given constant lower bound for the variable v.

Data Storage

For LMIPA, the comprehensive data storage format must be used. For quick generation of the comprehensive data storage objects, the file "gen_cmprhnsv_Data.p" can be used.

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). This matrix can be created using the following steps:

1) Assume that a process with *n* inputs and *m* outputs is being analysed. The transfer function between the j^{th} input and the i^{th} output is a rational function of either *s* or *z*, given as A_{ij} / B_{ij}, where *A* is the numerator and *B* is the denominator. Let *A'* be a row vector containing in descending order of powers of *s* or *z* the corresponding co-efficients of the numerator *A*. Similarly, let *B'* be the corresponding row vector of co-efficients of *s* or *z* for the denominator.
2) It is now necessary to create 2 cell arrays that contain all the information about the process. The first cell array, $\mathbb{A}$, is created as follows:

$$\mathbb{A} = \{A'_{11}, A'_{12}, \cdots, A'_{1n};$$
$$A'_{21}, A'_{22}, \cdots, A'_{2n};$$
$$\vdots \qquad\qquad \vdots \;;$$
$$A'_{m1}, A'_{m2}, \cdots, A'_{mn}\}$$

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 $\mathbb{A}\{i, j\}$. A similar cell array containing the values of the denominator, denoted as $\mathbb{B}$, is also created.

3) The time delay for the model can be created by forming the matrix $\mathbb{D}$, such that the (i, j) entry contains the time delay associated with the transfer function for the i^{th} output and j^{th} 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 $\mathbb{A}$, **B** is the cell array $\mathbb{B}$, and **D** is the time-delay matrix, $\mathbb{D}$. 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) If a gain matrix is being used, then the transfer function would simply consist of constant co-efficients. This can be created in MATLAB by assigning a value of "1" to each of the entries in **B** and the gain values would be the entries in **A**.

Output Data Storage Formats

The comprehensive data storage method must be used.

Comprehensive

Assume that there are p samples of output data with a total of t controlled variables and s manipulated variables with a total of q = 8 tags. In the comprehensive 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*(*t* + *s*)-by-1 cell matrix that contains the name of each of the tags that are presented in the process. The first *q* tags represent the first variable, while the next *q* represent the second variable and so on. For this toolbox, the tags are given as:
    a. **cell_char_TagList(1)**: This is the tag for the PV values.
    b. **cell_char_TagList(2)**: This is the tag for the low limits.
    c. **cell_char_TagList(3)**: This is the tag for the high limits.
    d. **cell_char_TagList(4)**: This is the tag for the soft low limits.
    e. **cell_char_TagList(5)**: This is the tag for the soft high limits.
    f. **cell_char_TagList(6)**: This is the tag for the linear optimisation co-efficients.
    g. **cell_char_TagList(7)**: This is the tag for the quadratic optimisation co-efficients.
    h. **cell_char_TagList(8)**: This is the tag for the target values.
3) **cell_char_TimeStamp**: This is a *p*-by-1 cell matrix that contains the time stamp for each of the samples.
4) **dbl_Comprehensive_Data**: This is a *p*-by-*q*(*t* + *s*) double matrix that contains the values for each of the tags and sample periods. Each of the columns contains the data in the same order as the cell_char_TagList cell array does.
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 Generation

The data storage files for multivariate analysis can be generated using the *p*-file "gen_cmprhnsv_data.p". The following steps should be followed to create a compact data set.

1) Start MATLAB and point the directory to the location of the above binary file.
2) At the command prompt create the following matrix, which contains the 3 samples of the 2 controlled variables: ""**>> W=[1 2;3 4;5 6]**".
3) It will be assumed for the sake of this example that the process to be entered can be described as

$$y = \begin{bmatrix} \frac{2.3}{s^2+1}e^{-5s} & \frac{2s+1}{s^2+3s+1}e^{-1s} \\ \frac{5}{s+1}e^{-1s} & \frac{4s}{s^2+3s+1}e^{-3s} \end{bmatrix}$$

Thus, the required arrays and matrices for the transfer function will be created as follows:

a) Form the 4 *A*-matrices as "**>>A11=[2.3]; A12=[2 1]; A21=[5]; A22=[4 0];**".

b) Form the 4 *B*-matrices as "**>>B11=[2 0 1]; B12=[1 3 1]; B21=[1 1]; B22=[1 3 1];**".

c) Create the *A*-cell array as "**>>A={A11,A12;A21,A22};**".

d) Create the *B*-cell array as "**>>B={B11,B12;B21,B22};**".

e) The time delay matrix can be created as "**>>D=[5 1;1 3];**". It should be noted that the negative signs in the time delay have been ignored.

f) The transfer function object is then created using the following command: "**Atf=tf(A,B, 'ioDelay',D)**". The resulting transfer function should match the one given above.

4) Type at the command prompt: ""**>> gen_cmprhnsv_Data**". The following should appear

Consider a process with m inputs and p outputs

1. p x m plant gain/continuous transfer matrix/discrete transfer matrix
2. Output data (y): N x p matrix
3. Input data (u): N x m matrix
4. Sampling time
5. Low limit data: (p + m) x 1 vector
6. High limit data: (p + m) x 1 vector
7. Linear coefficients: (p + m) x 1 vector
8. Quadratic coefficients: (p + m) x 1 vector
9. Target coefficients: (p + m) x 1 vector

Gain/continuous transfer matrix/discrete transfer matrix:

5) Type “Atf” or the name of the transfer function object that was created above and press enter. This is the transfer function object that contains the open-loop plant model or gain matrix. For the gain matrix, it would be simply a transfer function with constant coefficients.
6) Next, type “W” and press enter. This is the sampled data matrix for the given process.
7) Then, type “[0 1;0 1;0 1]” and press enter. This is the sampled input data matrix for the given process.
8) Following this, type “0.1” and press enter. It will be assumed that the sampling time is 0.1 seconds.
9) Next, type “[0;0;0;0]” and press enter. It will be assumed that the lower limit is 0 for all the process variables.
10) Then, type “[10;10;10;10]” and press enter. It will be assumed that the upper limit is 10 for all the process variables.
11) Following this, type “[1;1;1;1]” and press enter. It will be assumed that the linear coefficients are uniformly 1.
12) Next, type “[0.5;0.5;0.5;0.5]” and press enter. It will be assumed that the quadratic coefficients are uniformly 0.5.

13) Then, type "[0.75; 0.75; 0.75; 0.75]" and press enter. It will be assumed that the target co-efficients are uniformly 0.75.
14) Enter a name for the output data storage file, say, for example, "test_data.mat" and press enter. **It should be noted that files that have the same name as those currently present in the directory will be overwritten without any warning**.
15) Enter a name for the model storage file, say, for example, "test_model.mat" and press enter. **It should be noted that files that have the same name as those currently present in the directory will be overwritten without any warning**.
16) In order to view the results, type "**`>>load test_model.mat`**" followed by ""**`>>sys_MPC_Model`**". The results should be identical to that displayed below:

```
 Transfer function from input 1 to output...
                          2.3
 #1:  exp(-5*s) * ------------
                     2 s^2 + 1
                          5
 #2:  exp(-1*s) * -----------
                       s + 1


 Transfer function from input 2 to output...
                       2 s + 1
 #1:  exp(-1*s) * ------------------
                    s^2 + 3 s + 1


                          4 s
 #2:  exp(-3*s) * ----------------
                    s^2 + 3 s + 1
```

17) In order to view the other file type "**`>>load test_data.mat`**". This will load each of the entries into the workspace of MATLAB. Entering the name of each of the entries given below should give the same results as are presented in Table 1.

Table 1: Summary of the Parameters Generated using the "gen_cmprhnsv_Data" command

Entry	Value
Controller_status	[1 1 1]
cell_char_TagList	['CV1' 'CV1.N(1)' 'CV1.N(2)' 'CV1.X(17)' 'CV1.X(18)' 'CV1.X(8)' 'CV1.X(7)' 'CV1.X(9)' 'CV2' 'CV2.N(1)' 'CV2.N(2)' 'CV2.X(17)' 'CV2.X(18)' 'CV2.X(8)' 'CV2.X(7)' 'CV2.X(9)' 'MV1' 'MV1.N(1)' 'MV1.N(2)' 'MV1.X(17)' 'MV1.X(18)' 'MV1.X(8)' 'MV1.X(7)' 'MV1.X(9)' 'MV2' 'MV2.N(1)'

	'MV2.N(2)' 'MV2.X(17)' 'MV2.X(18)' 'MV2.X(8)' 'MV2.X(7)' 'MV2.X(9)']
cell_char_TimeStamp	It should give the current time incremented by 0.1, 0.2, and 0.3
dbl_Compact_Data	*This matrix is too large to display here.*
dbl_SamplingTime	0.1000
int_CVNumber	2
int_MVNumber	2
status	[1 1 2 2 1 1 2 2 1 1 2 2]

Using the Toolbox

The toolbox can be accessed from MATLAB using the following sequence of commands. First MATLAB itself should be started and the directory pointed to the folder containing the files for this toolbox. Next, at the command prompt type "**`>> main_lmipa`**". The GUI shown in Figure 5 should appear. This GUI is the main access to the toolbox. To start a session of the toolbox, click on the "LMIPA" menu. This will bring up a new GUI, which is shown in Figure 6. In Figure 6, each of the main parts of the GUI is highlighted and will be discussed separately.

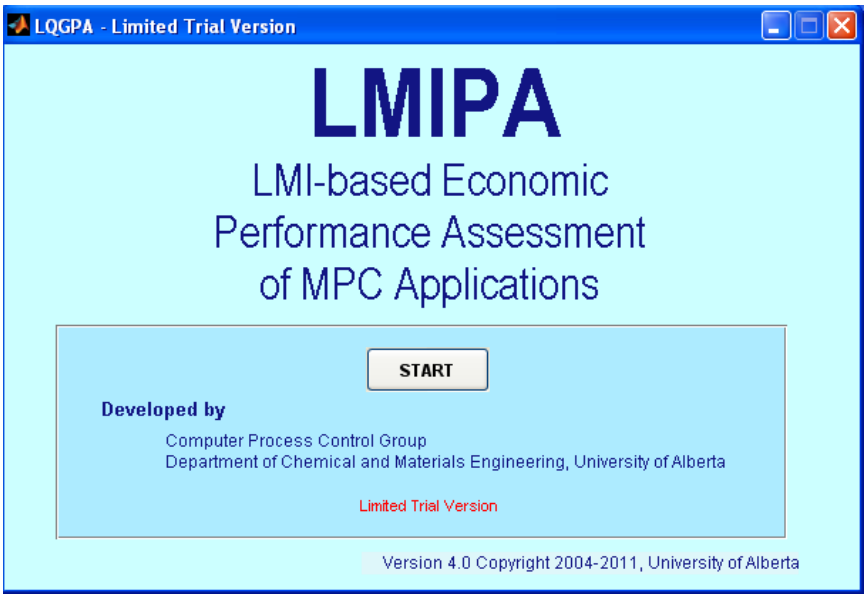


Figure 5: The First GUI that appears

As shown in Figure 6, there are 4 different sections in the main GUI.

Section 1: Main Menu

A close-up of the Main Menu section is given in Figure 7. The Main Menu consists of the following 4 major parts:

1) **File**: Clicking this menu will give 4 options:
    a. **New (Ctrl + N)**: This will clear all the data from the current GUI and allow the user to restart the analysis from a clean layout.
    b. **Open (Ctrl + O)**: This will load a previously saved project file.
    c. **Save (Ctrl + S)**: This will save the current project so that the analysis can be continued later.
    d. **Exit (Ctrl + Q)**: This will close the programme without saving anything.

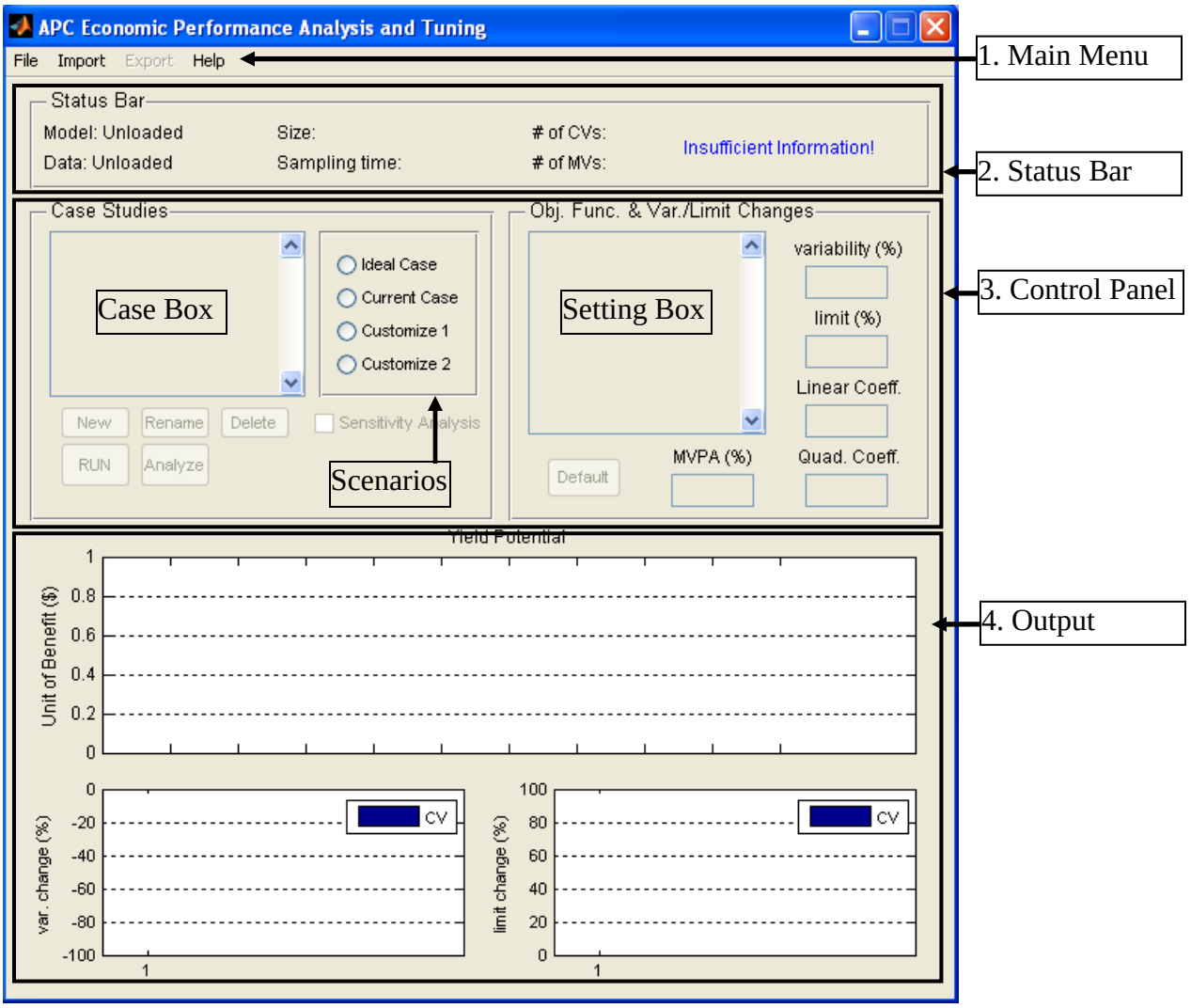


Figure 6: The main GUI for this toolbox, with the main regions highlighted

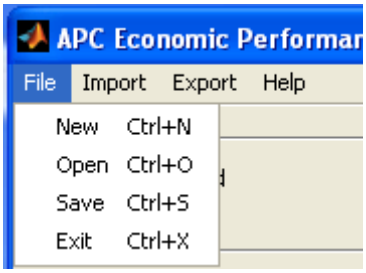


Figure 7: Close-up of the Main Menu

2) **Import**: Clicking this menu will give 2 options:
    a. **Model (Ctrl + M)**: This will allow the user to load the appropriate model for the process.
    b. **Data (Ctrl + D)**: This will allow the user to load the data for analysis.
3) **Export**: This menu is disabled until there is some information to export. In other words, at least one case study must be solved. When active, clicking this menu will bring up 2 options:
    a. **Send to .xls**: This will allow the user to send the results to an Excel (2003) file.
    b. **Send to .mat**: This will allow the user to send the results to a Matlab file.
4) **Help**: This menu contains information about the current version of the programme.

Section 2: Status Bar

This section shows whether or not the required information has been loaded, as well as some basic information about the loaded information, including model size and sampling time. A message will be displayed giving the current status. Only if a "Sufficient Information!" is displayed the rest of the panels will be enabled.

Section 3: Control Panel

This section consists of 2 panels. The left-hand panel manages all the case studies and scenarios, while the right-hand panel contains the system optimisation parameters that can be changed as necessary. In the left-hand panel, the following components are found:

1) **Case box**, which lists all the currently defined case studies. Selecting a given case study will display the associated parameters and information.
2) **"New" button**, which will create a new case study and place it in the list box with a default name. For the sake of simplicity, no more than 10 case studies can be created.
3) **"Rename" button**, which will allow the user to rename an existing case study. The new name can consist of any Latin characters (with or without accents); other scripts will not be displayed. This change will be applied to all sections showing this case's name such as the plots.
4) **"Delete" button**, which will allow the user to delete an existing case study. In order for this button to work, at least one case study must be present; otherwise this button is disabled.

5) **“RUN” button**, which will run the selected case study and display the results.
6) **“Analyze” button**, which will open the LMIPA assistant that allows for more detailed analysis of the data. More details about this can be found in the section LMIPA Assistant.
7) **Scenario radio buttons:** These 4 radio buttons allow the user to select which type of problem is to be analysed in the given case study. Each case study can only consider a single scenario. The 4 scenarios are:
    a. **Scenario 1 (Ideal Case)**, which considers the calculation of the ideal potential, which is equivalent to the Ideal Benefit Potential case in the Theory section.
    b. **Scenario 2 (Current Case)**, which considers the calculation of the current potential, which is equivalent to the Current (existing) Benefit Potential case in the Theory section.
    c. **Scenario 3 (Customize 1)**, which considers the calculation of the potential under the assumption that both the variability and process limits can be changed. The user can enter some selected values for variability and limit of each variable in the setting box and then have the corresponding potential calculated easily.
    d. **Scenario 4 (Customize 2)**, which considers the calculation of the required reductions in the variabilities and process limits given a desired potential. It should be noted that in this case, the objective function requires that a linear weighting be provided to set the importance of the given variables. This can be accomplished by selecting the “Lambda” box from the right-hand list box and entering values for “Var. weight” and “Lim. weight”. These weighting must be numbers between 0 and 100.
8) **“Sensitivity Analysis” checkbox,** which considers performing all the tasks given in Customize 2 through a different objective function which is discussed in sensitivity analysis literature.

In the right-hand panel, the following components are found:

1) **Settings list box**, which contains all the different options for the scenarios, which require user input.
    a. **Desired potential** which allows the user to specify the desired potential.

b. **Lambda** which allows the user to specify the variability weighting and process limit weighting in the format of a number between 0 and 100.

c. **Controlled Variable #**, which allows the user to specify the linear and quadratic coefficients in the objective function associated with the given controlled variable. As well, the variability and limits in percents of unity can be set. There will be as many entries as there are controlled variables. Not all values can be set for all scenarios.

d. **Manipulated Variable #**, which allows the user to specify the linear and quadratic coefficients in the objective function associated with the given manipulated variable. As well, the variability and limits in percents of unity can be set. There will be as many entries as there are manipulated variables. Each time we want to change a parameter, we may select one item in this list box and change the appropriate value in edit boxes placed around it. Not all values can be set for all scenarios.

e. **Notes:** Since the variability is specified as a reduction from the current values, its value must be negative. The original period for variability and limit are [-1, 0] and [0, 1] respectively. Here for the sake of simplicity, numbers are selected from [-100, 0] and [0, 100]. Hence, -100 and 100 correspond to -1 and 1.

2) **Default**, which sets the parameter values to the initial default values.

3) **MVPA (%)**, which allows for the manipulated variables, the specification of the multivariate performance assessment index as a percent.

Section 4: Output

This section has three plots that show the results obtained from the solved cases. The upper plot shows the calculated potential for each case and its value as percentage of the ideal case's value. The other two plots show variables variability and limit that are calculated for the last scenario.

LMIPA Assistant

Clicking the "Analyze" button will open the LMIPA assistant, which allows the user to select the data that will be used in the analysis. Figure 8 shows the window that appears. It consists of 7 different areas:

a. **Figure Tools (Section 1)**: This is the generic MATLAB toolbar for figures.
b. **Model Data (Section 2)**: This section contains information about the model, such as the number of controlled and manipulated variables, sampling time, and feasibility.
c. **Time Section (Section 3)**: This allows the user to select the time section of the data that will be used. 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. It is possible to try more than 1 section simultaneously. Each sample section is separated by a semicolon (;). For example "[1,100; 40, 500]" will consider 2 sections; the first ranging from sample 1 to sample 100 and the second from sample 40 to sample 500. Clicking "Display" will display the selected samples in the data plot area (Section 6). "Apply" button must be pressed in order for the data to be saved.
d. **Select Variables (Section 4)**: This drop-down menu allows the user to select which variables will be displayed.
e. **Data Selected (Section 5)**: This allows the user to select what aspect of the selected variable is to be displayed. It can be noted that "std" represents the standard deviation of the individual data. The data is displayed in the following manners:
    i. Histogram: The On/Off status and Std graphs.
    ii. Line Graphs: The Data/Limit Range and all the individual CV values.
f. **Data Display (Section 6)**: This displays the selected data. The abbreviations are explained in Figure 9, which can be obtained by clicking the "Legend" button located in Section 7.
g. **Useful Buttons (Section 7)**: This contains 2 buttons that display the legend (Legend) or close the given window (Close).

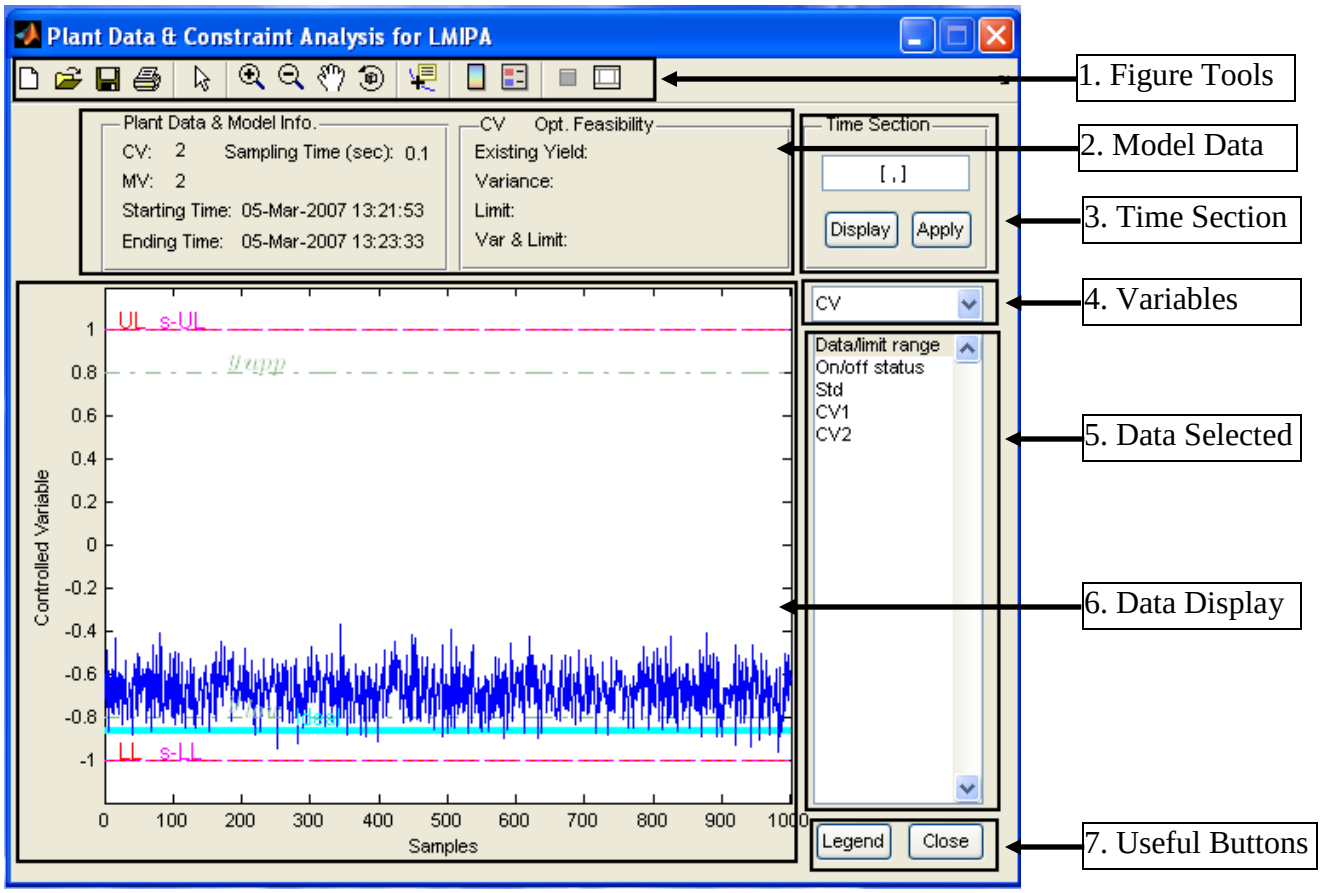


Figure 8: Plant Data & Constraint Analysis for LMIPA

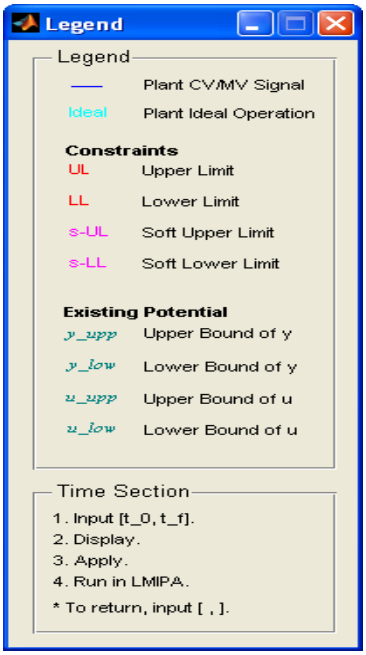


Figure 9: Explanation of the abbreviations used in the Plant Data & Constraint Analysis for LMIPA window

Examples

The following example will present how the analysis can be performed using MATLAB. The example is based on the data provided in the unzipped folder, which consists of 2 manipulated variables and 2 control variables.

Part I: Using the Programme to Obtain the Results

The first step is to load the programme, "**`>>main_lmipa`**", and click on "LMIPA" in the window that appears.

Next, load the model into the programme. This can be accomplished by clicking on the "Model" from "Import Menu" and selecting the "**`Case_model.mat`**" file. After a few seconds, load the data by clicking on "Data" from "Import Menu". Then select the file entitled "**`Case_Cmprhnsv_Data.mat`**". By now the first case has already been created automatically. Next, click on the "Run" push button to obtain the screen shown in Figure 10.

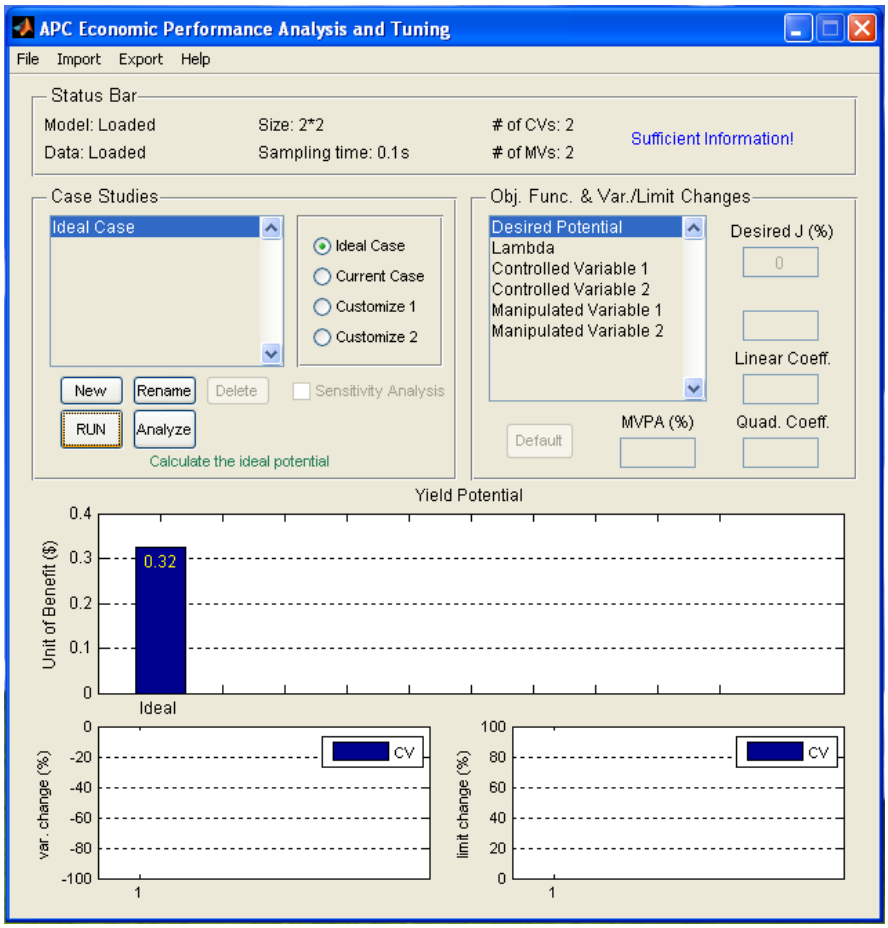


Figure 10: Screen shot 1 for the example

Figure 10 shows the ideal benefit potential is 0.32. In order to know the potential yield in the current system, a case with scenario 2 (Current Case) is needed. Hence, we must click on the "New" push button and select "Current Case". We can also change its name of the case to "Realistic" and finally run it. Results are shown in Figure 11.

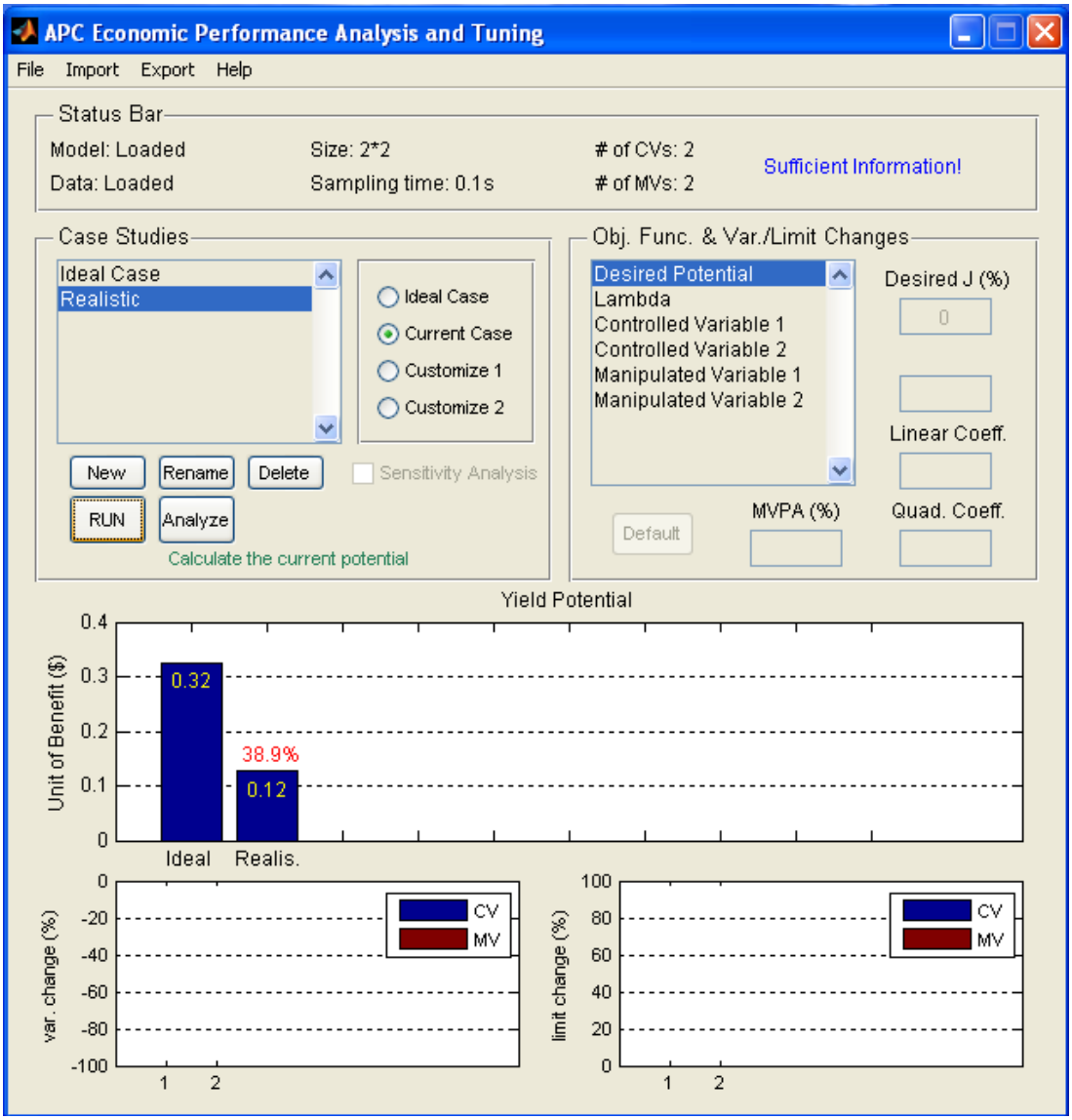


Figure 11: Screen shot 2 for the example

Now let us assume the objective is to find the potential of the system if the variability and process limits change. In this case, a new case is selected and the third scenario "Customize 1"is chosen. On the right side of the control panel, select Controlled Variable 1 and set its variability and limit to be -30% and 5% respectively. Note that since we are interested in a reduction in

variability, the variability of the variables must be negative. The other parameters remain unchanged. Running this case will produce Figure 12.

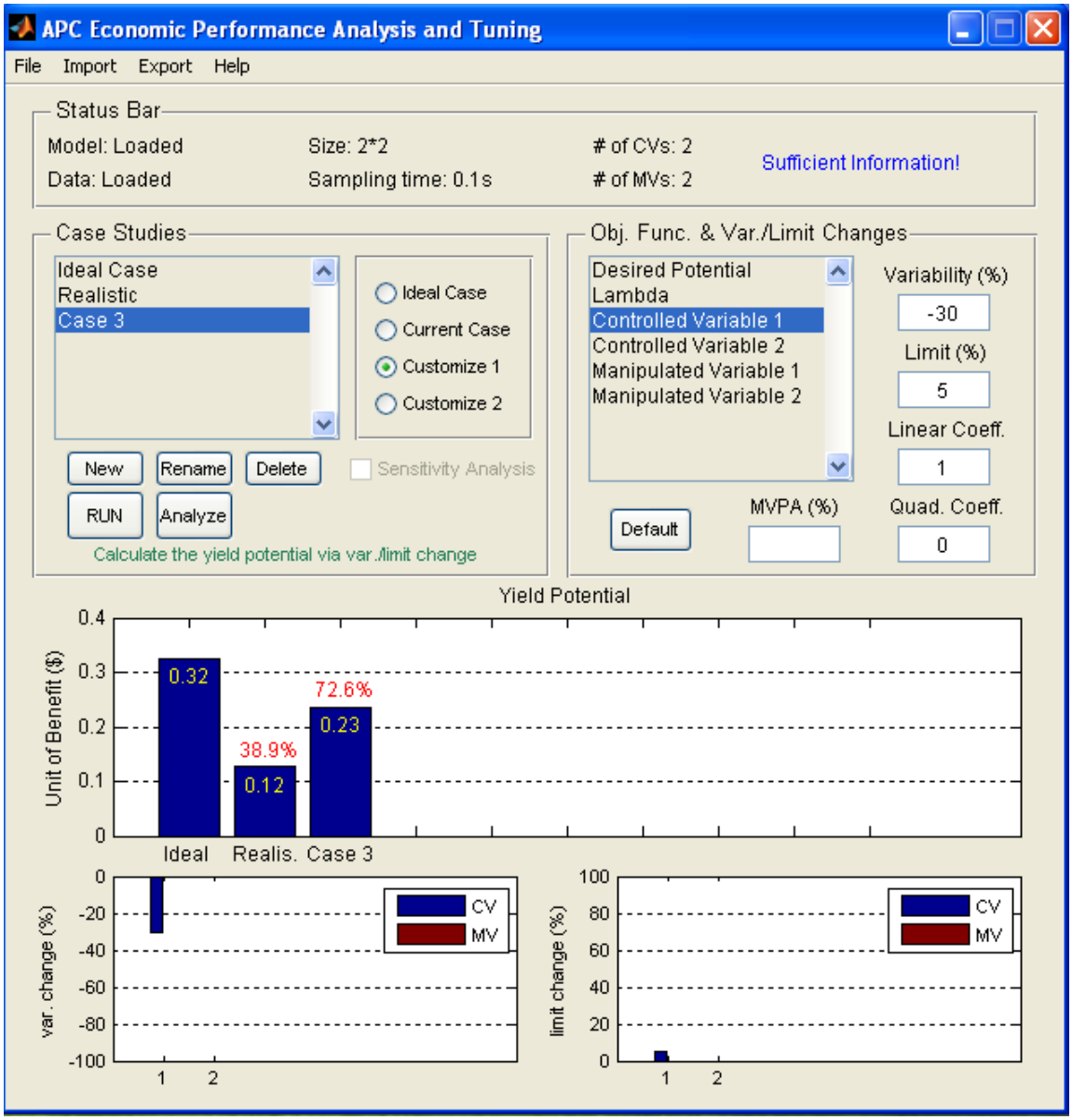


Figure 12: Screen shot 3 for the example

It can be seen that changing the process variability by -30% and the process limits by 5% increases to 0.23, which is 72.6% of the ideal case.

Next, assume that we would like to achieve 80% of the ideal potential by changing only the variability of the variables. This is what Scenario 4 (Customize 2) is concerned with. Therefore, create a new case and select Scenario 4. Select “Desired Potential” from the right-hand list box and enter 80 into the textbox labelled “Desired J (%)”. Note that since we are only

interested in the change of variability, Lambda weighting coefficients must be 100 and 0 for variability and limit respectively. The results are shown in Figure 13.

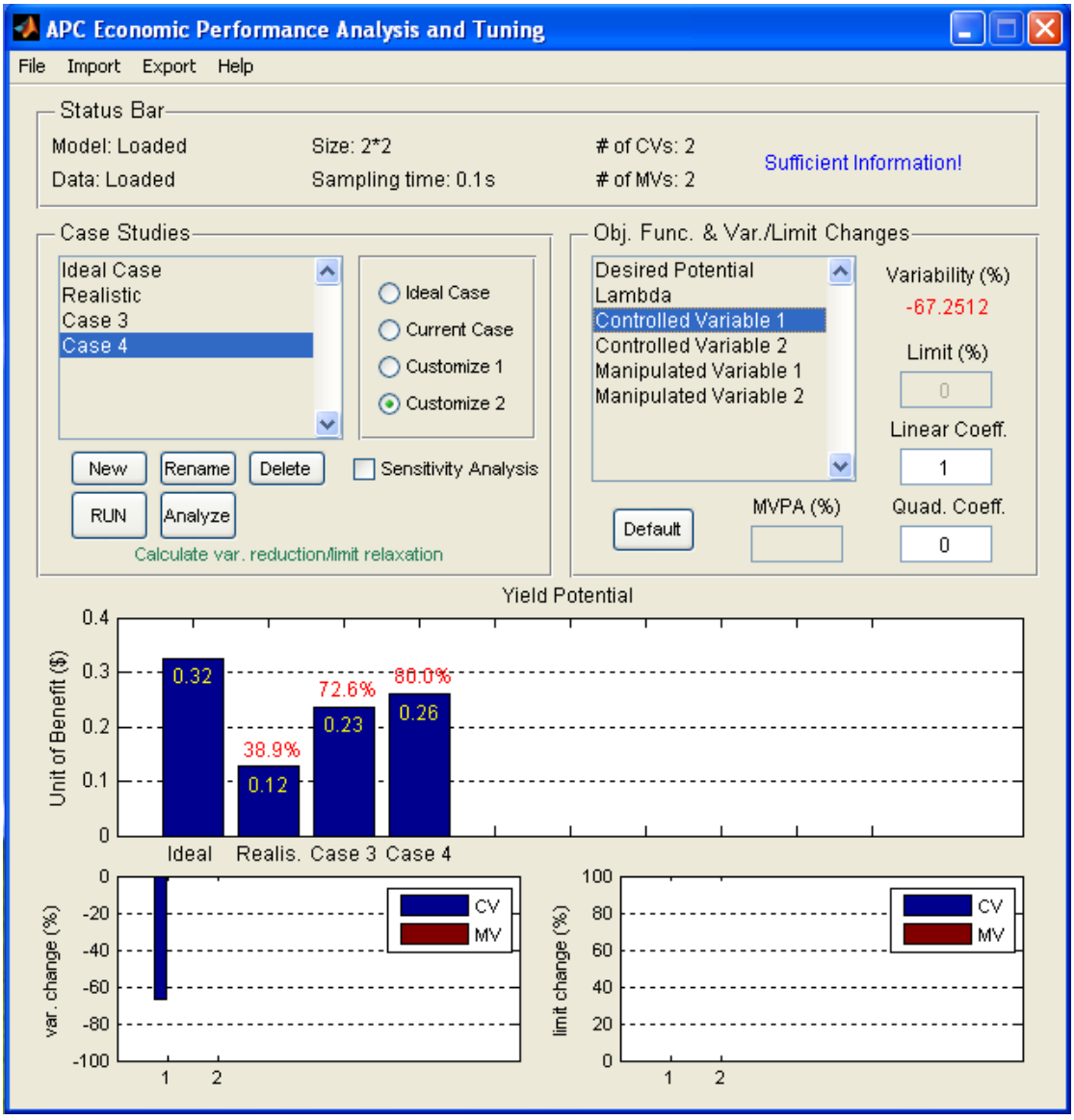


Figure 13: Screen shot 4 for the example

It can be seen that the variability of controller variable 1 must decrease by 67.3% in order to achieve a potential 80% of the ideal one, that is, setting v_1 = -0.6725 and v_2 = 0 will bring about this change.

For the last example, Scenario 4 is tried again to achieve a potential that is 50% of the ideal potential. This time both variable variability and limit values (%) are used to do the optimization over. For the objective function, two linear weight parameters are required.

Therefore, set 20 for variability and 80 for limit for “Lambda”. After running this case, Figure 14 will appear. This shows that by changing the limits a small amount, the potential that is 50% of the ideal can be achieved.

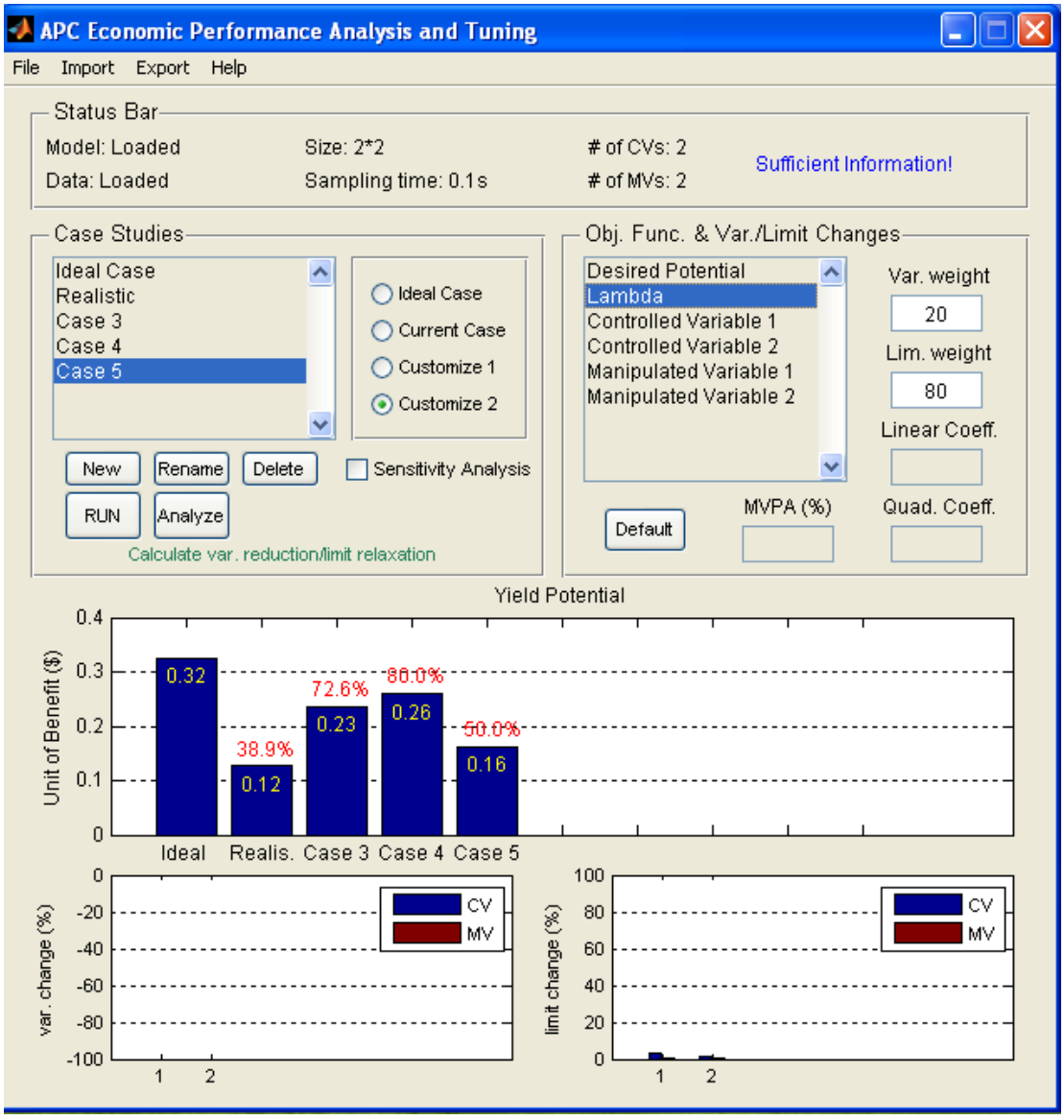
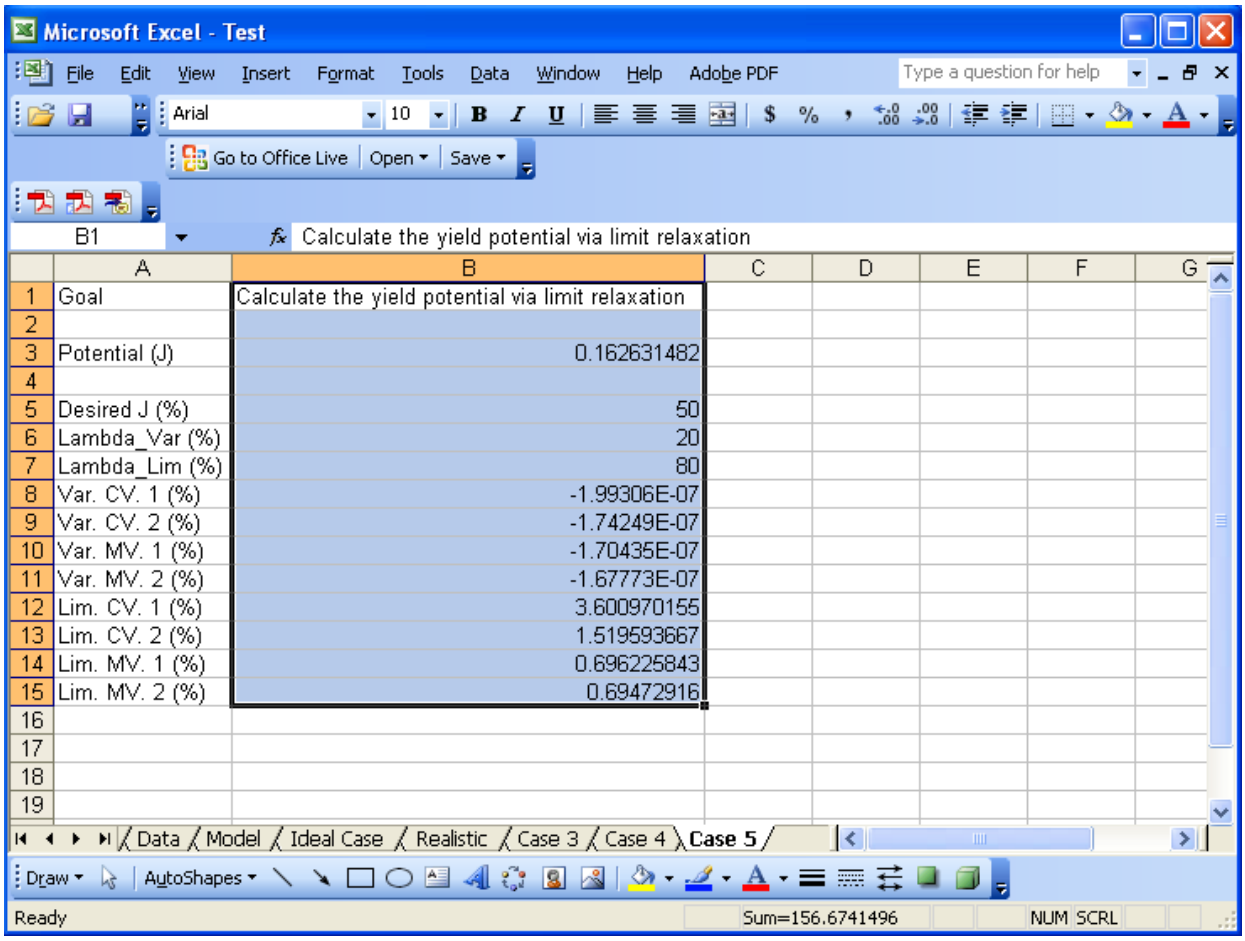


Figure 14: Screen shot 5 for the example

This procedure could be repeated for more cases. The project as a single file using the Save option from the “File” menu. As well, the results can be exported to Excel to MATLAB files using the “Export” menu. In the Excel file, each worksheet will contain the solved values for a given case. 2 sheets will also contain the information regarding model and data that were used. Figure 15 shows the Excel file produced from the results obtained in this example.



	A	B
1	Goal	Calculate the yield potential via limit relaxation
2		
3	Potential (J)	0.162631482
4		
5	Desired J (%)	50
6	Lambda_Var (%)	20
7	Lambda_Lim (%)	80
8	Var. CV. 1 (%)	-1.99306E-07
9	Var. CV. 2 (%)	-1.74249E-07
10	Var. MV. 1 (%)	-1.70435E-07
11	Var. MV. 2 (%)	-1.67773E-07
12	Lim. CV. 1 (%)	3.600970155
13	Lim. CV. 2 (%)	1.519593667
14	Lim. MV. 1 (%)	0.696225843
15	Lim. MV. 2 (%)	0.69472916

Figure 15: Excel screen shot for the example

Part II: Time Section in LMIPA

In order to enter time section in LMIPA, click on the “Analyze” push button. After appearing LMIPA assistant, go to section 3 (Time Section) of Figure 8 and enter into the box: “[100, 500]”. Press the “Display”, which is located below the box. Select “CV1” in section 4. The figure shown in Figure 16 should come up. Other controlled variables and manipulated variables can be selected. Once you have confirmed that the desired section has been selected. Click the “Apply” button, which is located in section 3 below the textbox. Finally, you may run the case again and this time only data from selection time period will be used for analysis.

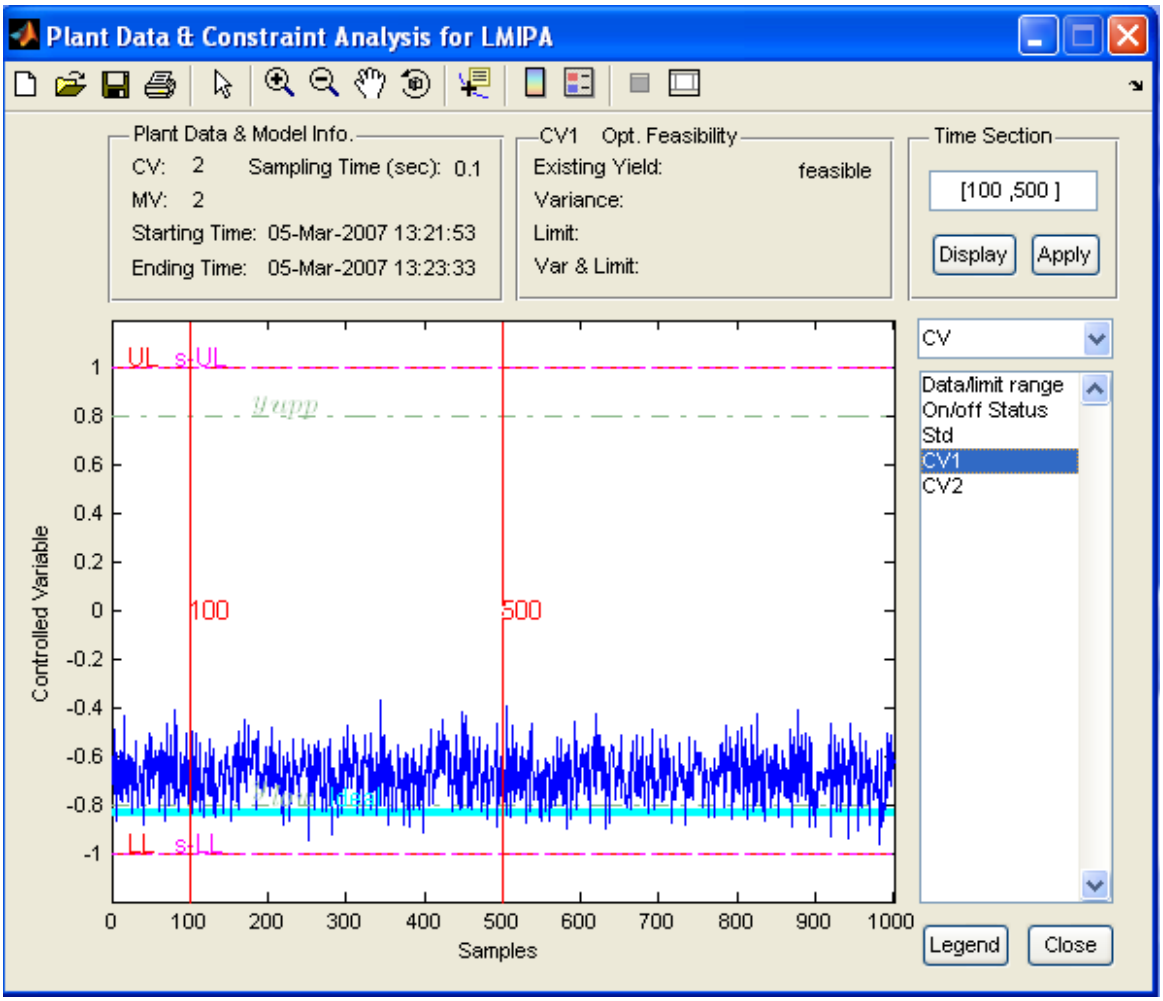


Figure 16: Showing the selected time sections for CV1

References

Lee, K. H., Huang, B., & Tamayo, E. C. (2008). Sensitivity analysis for selective constraint and variabliltiy tuning in performance assessment of industrial MPC. *Control Engineering Practice* (16), 1195-1215.