University of Alberta Computer Process Control Group

LMIPA

Limited Trial Version

Written by: CPC Control Group, University of Alberta

Version 2.0

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

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

1) MATLAB 2007b (MATLAB 7.1) or lower. It should be noted that, since the binary files have not been recompiled for the newest version of MATLAB, the programme will not run on MATLAB 2008a.
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 the unzipped files, there is a folder called SolverTools. This folder, along with any subfolders, must be added to the default paths in MATLAB. This is performed as follows:
    a. Start MATLAB.
    b. Go to File→Set Path.
    c. A "Set Path" Window (Figure 1b) will appear. Click the button "Add with subfolders". Its location is shown by a box with the label 1 in Figure 1b. Find the folder "SolverTools" and select it. Click "OK".
    d. Once MATLAB has finished, click on the "Save" button on the "Set Path" window to save the new default paths. Its location is shown by a box with the label 2 in Figure 1b.
    e. The default paths are those paths that MATLAB will search to find a given function.

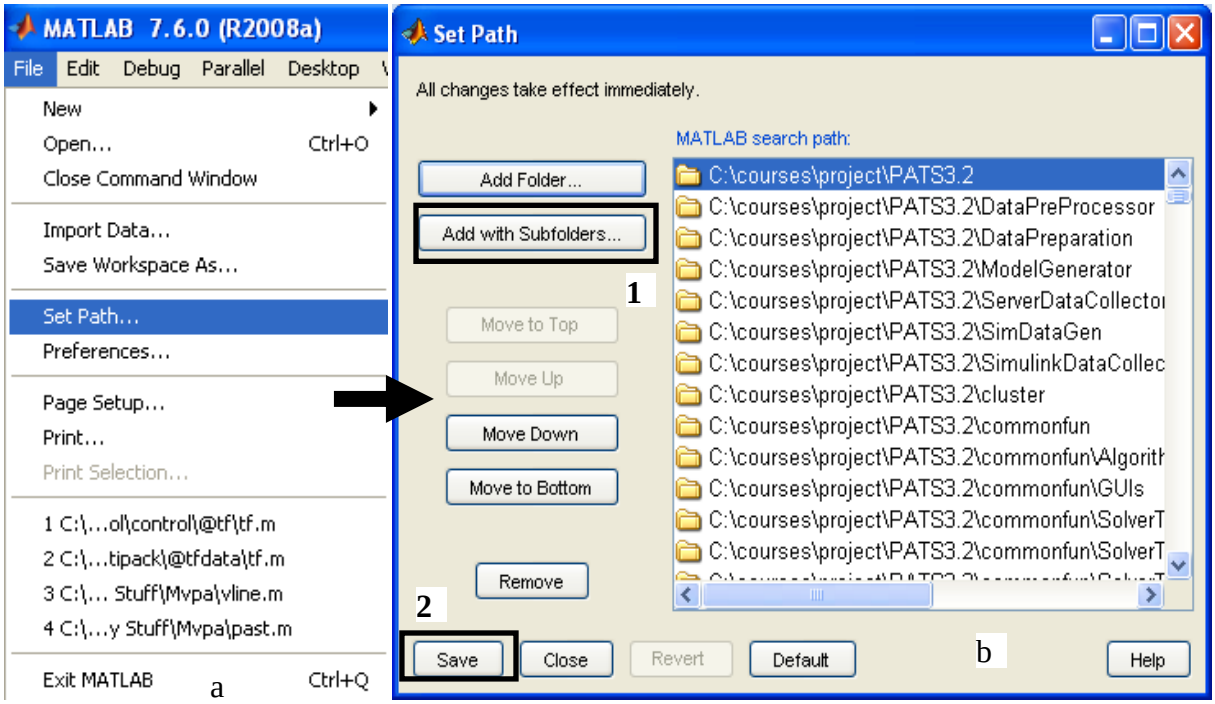

Figure 1: The steps required to add default paths in MATLAB. a) location of the "Set Path" command and b) the "Set Path" Window

3) In order to use the software, point the current directory to the location of the unzipped files.
4) At the command prompt, type "**`>> main_lmipa`**" to start the toolbox. The GUI shown in Figure 2 should appear.
5) Press the "LMIPA" menu. A new GUI will appear that is shown in Figure 3.

Figure 2: The First GUI that appears

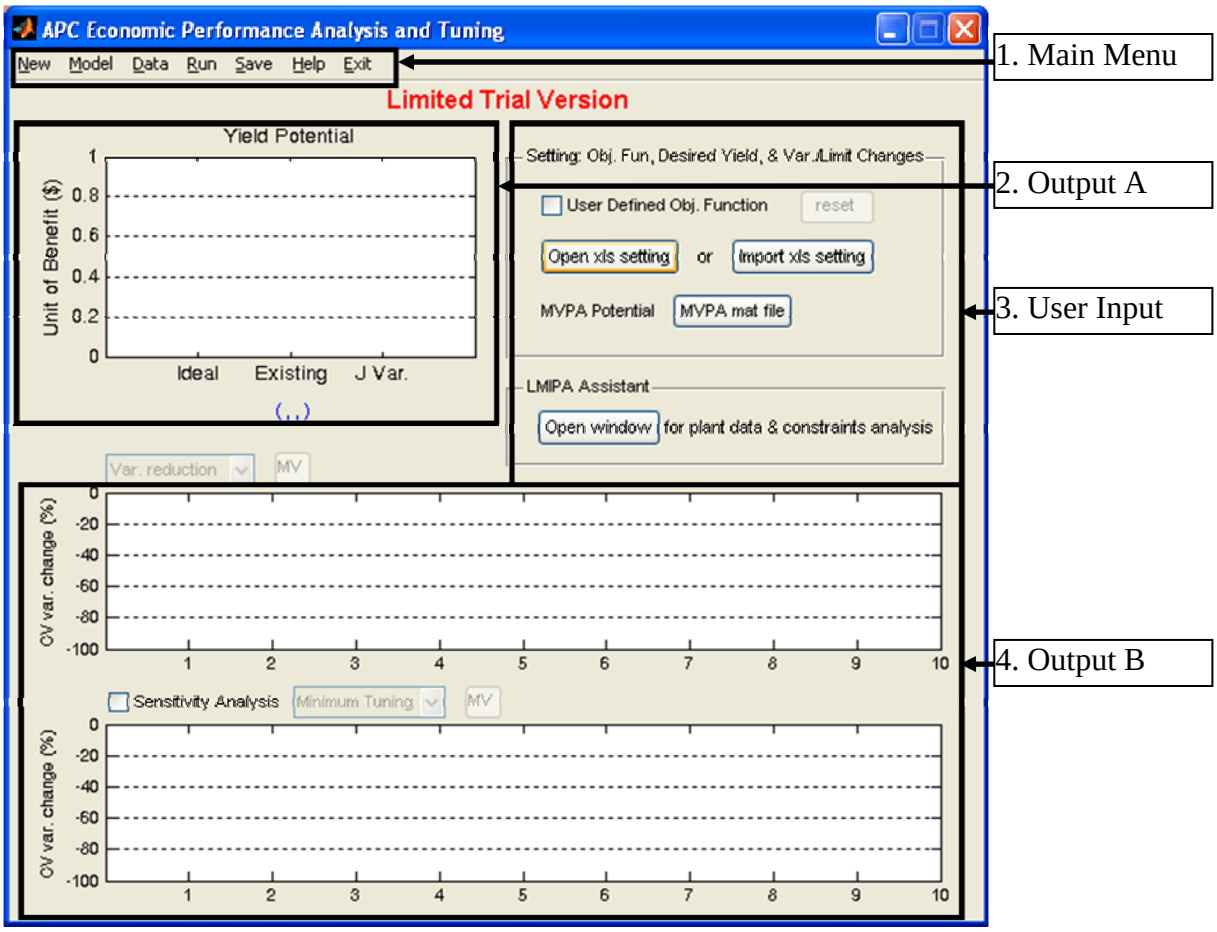

Figure 3: The main GUI for this toolbox, with the main regions highlighted

6) For the purpose of this quick start, we will be using the sample data provided in the zip file. Go to the "Model" Menu located in the Main Menu area, which is denoted as 1 in Figure 3. 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 "Data" Menu located in the same area. Select the "Case_Cmprhnsv_Data.mat" file. Wait patiently as the file is loaded.

7) After a few seconds, the data should be loaded. In order to see a plot of the data, go to the "Open window" button located in area 3 of Figure 3. The resulting window is shown in Figure 4. The abbreviations used in the graph are explained by clicking the "Legend" button located in area 7 of Figure 4. The resulting window is shown in Figure 5. In this case, none of the values will be changed. Thus, click the "Close" button located in area 7 of Figure 4.

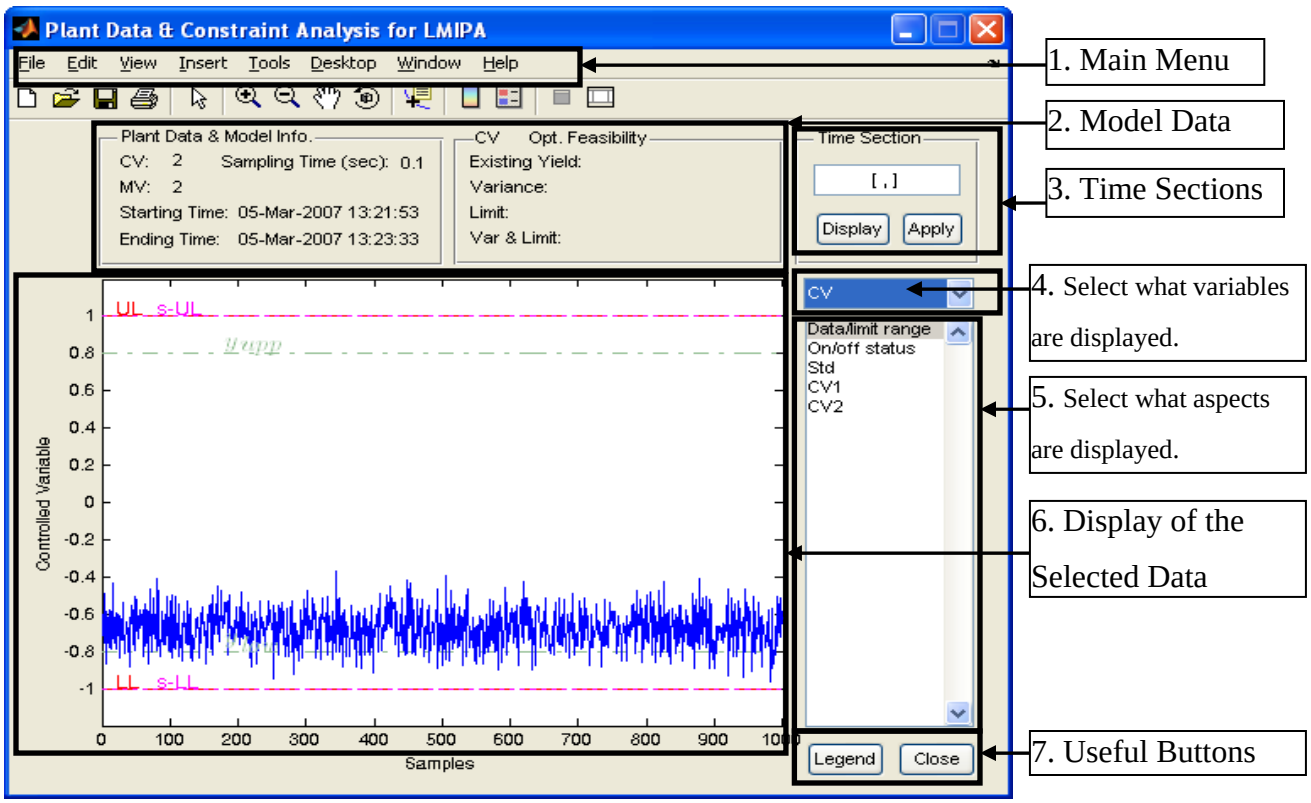

Figure 4: Plant Data & Constraint Analysis for LMIPA

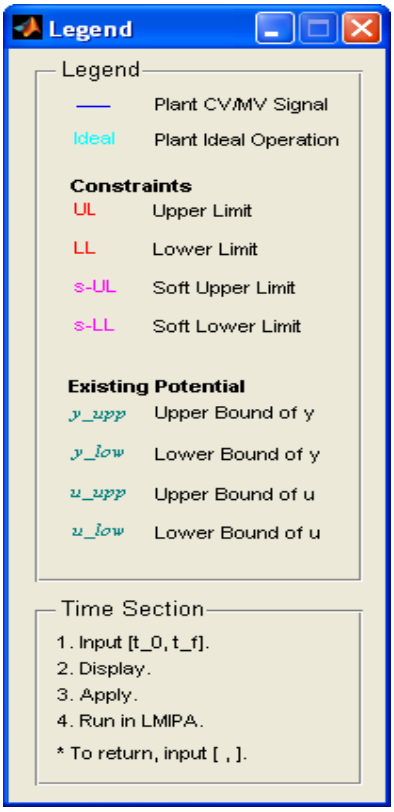

Figure 5: Explanation of the abbreviations used in the Plant Data & Constraint Analysis for LMIPA window

8) At this point, the “Run” menu is selected from area 1 of Figure 3. After a few minutes, the screen-shot shown in Figure 6 will appear.

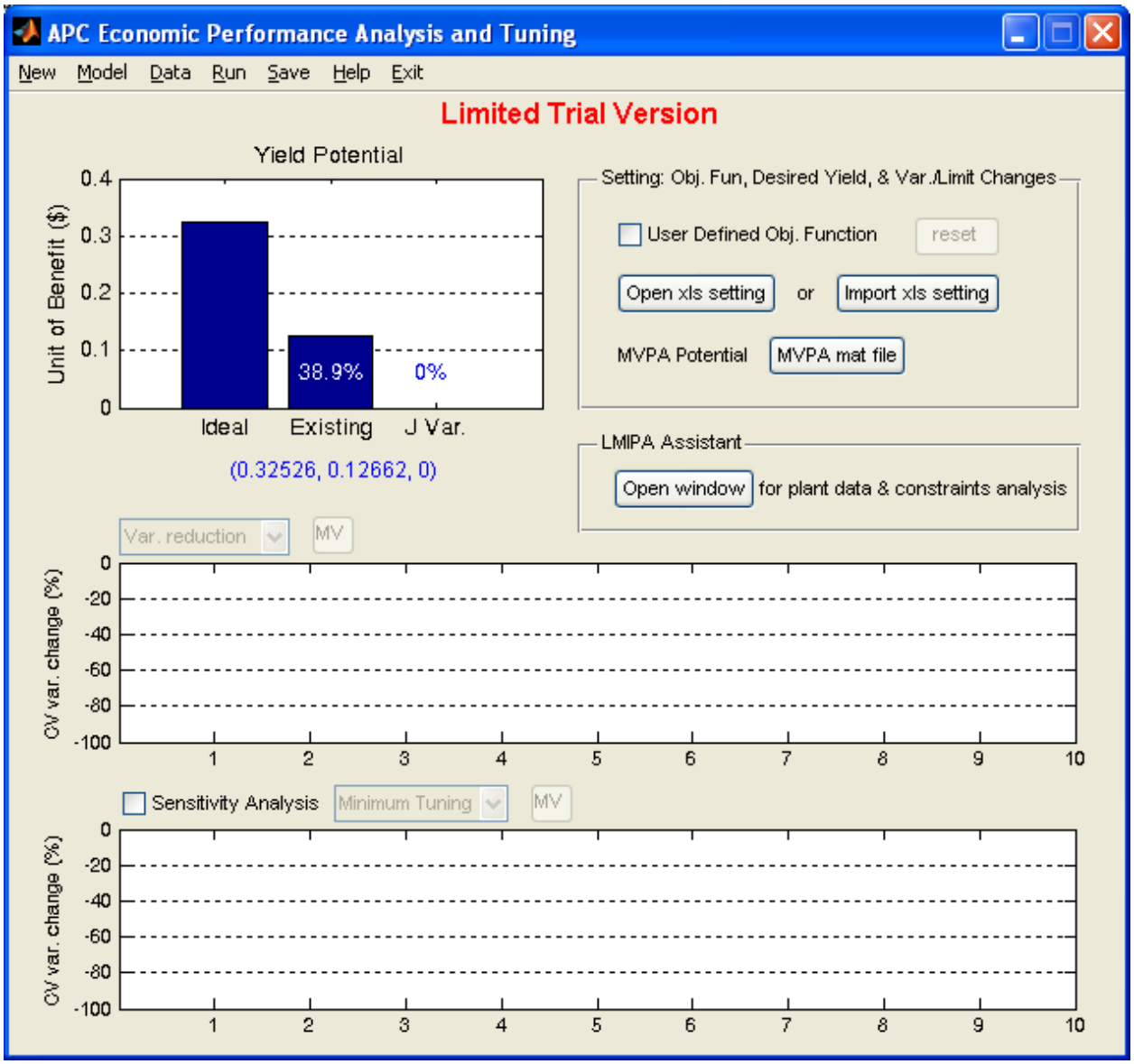

Figure 6: Screen shot after clicking "Run"

9) The histogram located in area 2 of Figure 6 consists of 3 bars, which show the yield potential under different conditions. The first bar shows the “ideal” yield potential, while the second bar shows the actual (existing) benefit obtained using the given controller. As well, actual benefit as a fraction of the ideal benefit is shown as a percentage. Finally, the third bar shows the controller potential if the given reduction in variance can be reached or if the constraints can be relaxed to the given value. Since in this example, no target

reductions or relaxations were given, this value is equal to zero. The actual values are shown below the histogram.

10) In order to assess the model, given target reductions in the variance or relaxation of the constraints, click on the "Open xls settings" button located in area 3 of Figure 3. This will open an Excel spreadsheet, which is shown in Figure 7.

	A	B	C	D	E	F
1		Var. (%)	IsValid	Limit (%)	IsValid	
2	J Ratio	0		0		
3	Lambda	0		0		
4	CV1	0	1	0	1	
5	CV2	0	1	0	1	
6	MV1	0	1	0	1	
7	MV2	0	1	0	1	
8						

Figure 7: Cropped view of the Excel spreadsheet

11) The spreadsheet shown in Figure 7 contains 4 columns:

a. Var. (%), which allows the user to set the percent reduction in the given variable,
b. Limit (%), which allows the user to set the percent relaxation in the given variable.
c. IsValid, which consists of 2 columns that determine whether or the given variables are to be considered in the analysis.

It should be noted that the *J*-ratio represents what amount of the ideal benefit can is desired, while λ represents the linear weighting co-efficient.

12) For the purposes of this demonstration, it will be desired to examine which constraints could be changed in order to achieve 60% of the ideal benefit potential. Enter the value 60 into cell D2 of the spreadsheet and click "Save" to save the results. *If "save" is not clicked, then the most recent results will not be used.*

13) Click on the "Run" menu as before in order for the results to appear. The new results are shown in Figure 8.

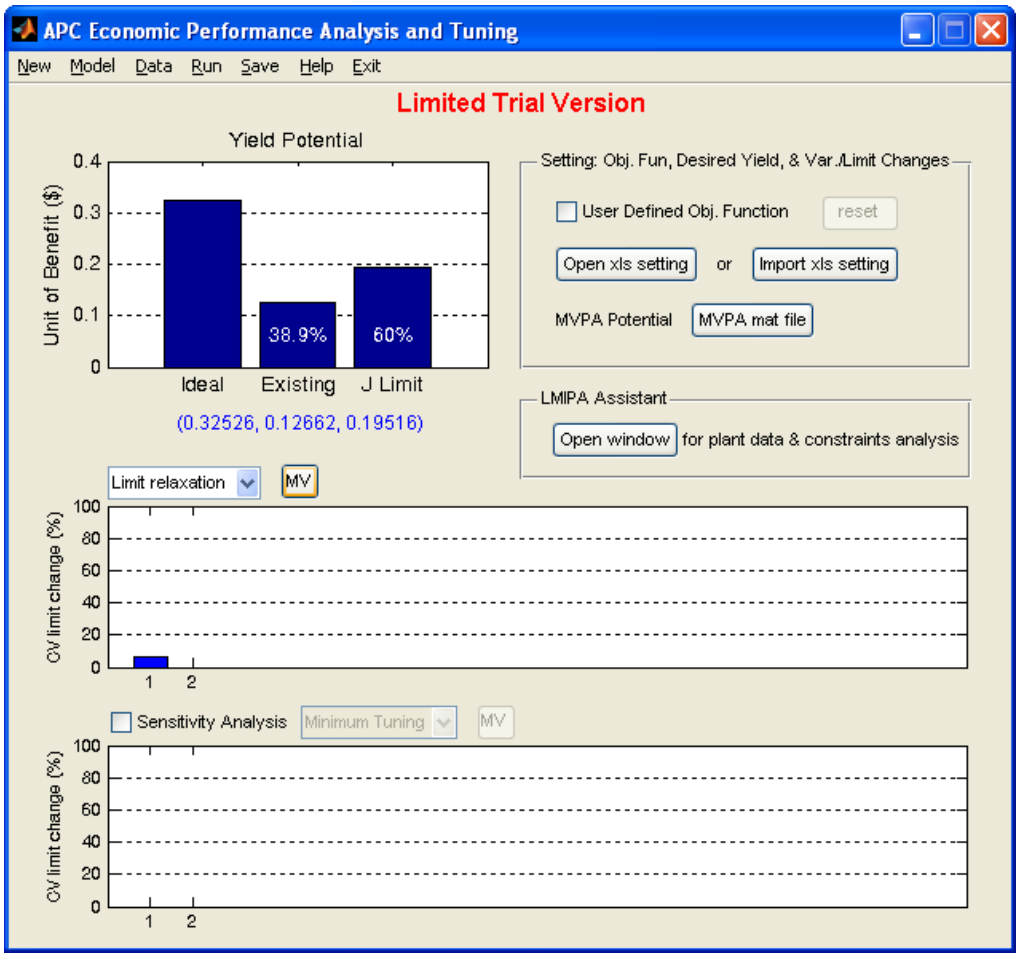

Figure 8: Screen-shot after desiring a *J* 60% of the ideal

14) Figure 8 shows in the top histogram of area 4 (Output B), the amount of relaxation in the limit values for the control (CV) and manipulated variables (MV). This can be toggled by clicking the MV (or CV) button located to the right of the drop-down menu at the top of the graph. In this example, approximately a 10% relaxation, that is an increase in the upper limit, of the limit is required.

15) Next, sensitivity analysis will be performed to determine which variables, if any, are sensitive to changes. This can be achieved by clicking the “Sensitivity Analysis” checkbox that is located above the second histogram. After selecting it, press the “Run” menu to obtain the results. The results are shown in Figure 9.

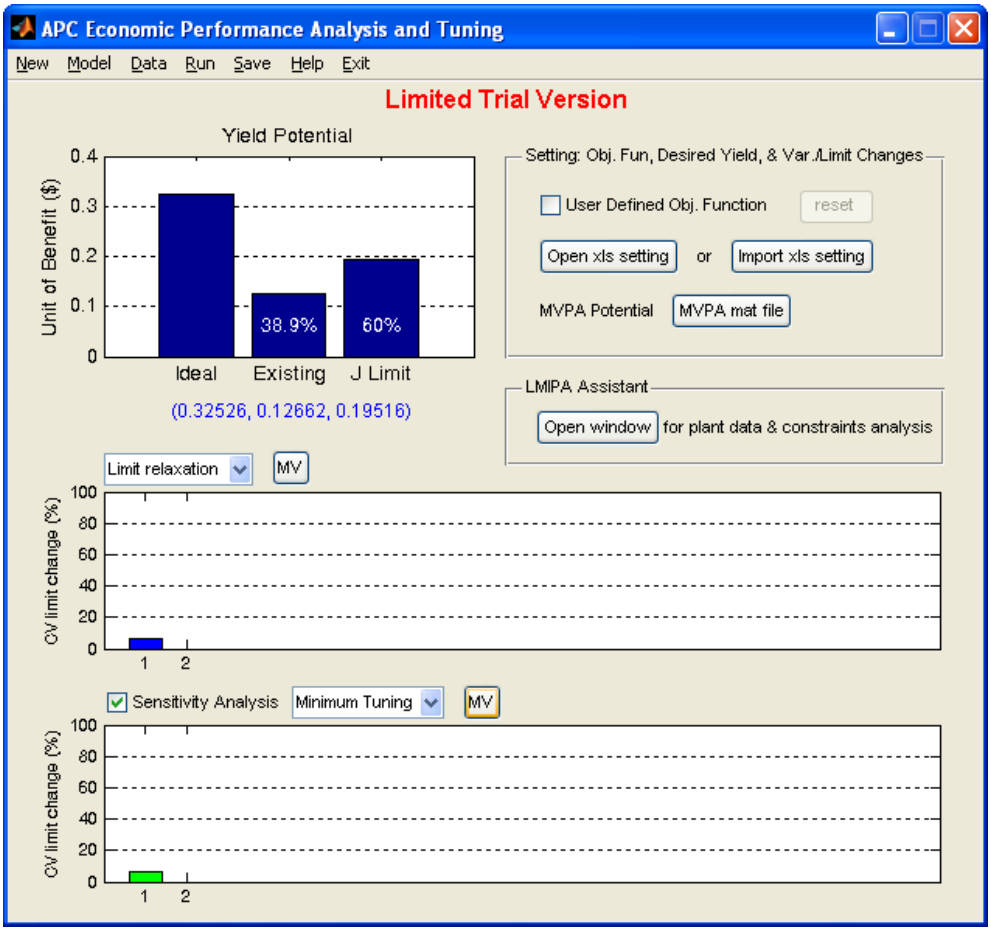

Figure 9: Screen-shot after Sensitivity Analysis is performed

16) This suggests that manipulated variable 1 should be changed first to obtain the desired change.

17) The results can be saved as a MATLAB file by clicking “Save,” which is located in the Main Menu area. The programme can be quit by clicking on “Exit” twice 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),

$$\min J \tag{2.1}$$

where *J* is defined as

$$J = \frac{1}{N_L}\sum_{k=1}^{N_L} J(k) \tag{2.2}$$

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

$$J(k) = \sum_{i=1}^{p}\left[b_{y_i}(k)\bar{y}_i + a_{y_i}^2(k)\left(\bar{y}_i - y_{s_i}(k)\right)^2 \right] + \sum_{i=1}^{m}\left[b_{u_i}(k)\bar{u}_i + a_{u_i}^2(k)\left(\bar{u}_j - u_{s_j}(k)\right)^2 \right] \tag{2.3}$$

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:

$$\sum_{j=1}^{m} K_{ij}\Delta\bar{u}_j = \Delta\bar{y}_i \qquad \forall i = 1,2,...,p$$

$$\begin{aligned} L_{y_i}(k) - \lambda_{y_i}\alpha_{y_i}(k) + 2\sigma_{y_i}\left(1+\nu_i\right) \le \bar{y}_i \le H_{y_i}(k) + \mu_{y_i}\alpha_{y_i}(k) - 2\sigma_{y_i}\left(1+\nu_i\right) \qquad & \forall i = 1,2,...,p \\ L_{u_j}(k) - \lambda_{u_j}\alpha_{u_j}(k) + 2\sigma_{u_j} \le \bar{u}_j \le H_{u_j}(k) + \mu_{u_j}\alpha_{u_j}(k) - 2\sigma_{u_j} \qquad & \forall j = 1,2,...,m \end{aligned} \tag{2.4}$$

where

$$\begin{aligned} \bar{y}_i &= \bar{y}_{0_i} + \Delta\bar{y}_i, & \bar{y}_{0_i} &= \frac{1}{N_L}\sum_{k=1}^{N_L} y_{0_i}(k) & \forall i = 1,2,...,p \\ \bar{u}_j &= \bar{u}_{0_j} + \Delta\bar{u}_j, & \bar{u}_{0_j} &= \frac{1}{N_L}\sum_{k=1}^{N_L} u_{0_j}(k) & \forall j = 1,2,...,m \end{aligned} \tag{2.5}$$

$\bar{y}_0$ 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, ν 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 $\overline{y}_i = \overline{y}_{0_i}$ or $\Delta\overline{y}_i = 0$, while for constraint (2.4c), replace it by $\overline{u}_j = \overline{u}_{0_j}$ or $\Delta\overline{u}_j = 0$. 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):

$$\min_{\left\{\overline{y}_i, \overline{u}_j, \lambda_{y_i}, \lambda_{u_j}, \mu_{y_i}, \mu_{u_j}\right\}} \left(\sum_{i=1}^{p} \lambda_{y_i} + \mu_{y_i} + \sum_{j=1}^{m} \lambda_{u_j} + \mu_{u_j} \right) \tag{2.6}$$

subject to

$$\begin{cases} \overline{\lambda}_{y_i} \geq \lambda_{y_i} \geq 0 & flag_{\lambda_{y_i}} = 1 \\ \lambda_{y_i} = 0 & flag_{\lambda_{y_i}} = 0 \end{cases} \tag{2.7}$$

$$\begin{cases} \overline{\mu}_{y_i} \geq \mu_{y_i} \geq 0 & flag_{\mu_{y_i}} = 1 \\ \mu_{y_i} = 0 & flag_{\mu_{y_i}} = 0 \end{cases} \tag{2.8}$$

$$\begin{cases} \overline{\lambda}_{u_j} \geq \lambda_{u_j} \geq 0 & flag_{\lambda_{u_j}} = 1 \\ \lambda_{u_j} = 0 & flag_{\lambda_{u_j}} = 0 \end{cases} \tag{2.9}$$

$$\begin{cases} \overline{\mu}_{u_j} \geq \mu_{u_j} \geq 0 & flag_{\mu_{u_j}} = 1 \\ \mu_{u_j} = 0 & flag_{\mu_{u_j}} = 0 \end{cases} \tag{2.10}$$

$$\frac{J_0 - J}{\Delta J_I} = R_C \tag{2.11}$$

$$\begin{aligned} L_{y_i}(k) - \lambda_{y_i}\alpha_{y_i}(k) + 2\sigma_{y_i} \leq \overline{y}_i \leq H_{y_i}(k) + \mu_{y_i}\alpha_{y_i}(k) - 2\sigma_{y_i} \\ L_{u_j}(k) - \lambda_{u_j}\alpha_{u_j}(k) + 2\sigma_{u_j} \leq \overline{u}_j \leq H_{u_j}(k) + \mu_{u_j}\alpha_{u_j}(k) - 2\sigma_{u_j} \end{aligned} \tag{2.12}$$

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

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):

$$\min_{\{\overline{y}_i, \overline{u}_j, \nu_i\}} \left(\sum_{i=1}^{p} |\nu_i| \right) \tag{2.13}$$

subject to

$$\begin{cases} \underline{\nu}_i \leq \nu_i \leq 0 & flag_{\nu_i} = 1 \\ \nu_i = 0 & flag_{\nu_i} = 0 \end{cases} \tag{2.14}$$

$$\frac{J_0 - J}{\Delta J_I} = R_C \tag{2.11}$$

$$\begin{aligned} L_{y_i}(k) - \lambda_{y_i}\alpha_{y_i}(k) + 2\sigma_{y_i}(1+\nu_i) \leq \overline{y}_i \leq H_{y_i}(k) + \mu_{y_i}\alpha_{y_i}(k) - 2\sigma_{y_i}(1+\nu_i) \\ L_{u_j}(k) - \lambda_{u_j}\alpha_{u_j}(k) + 2\sigma_{u_j} \leq \overline{u}_j \leq H_{u_j}(k) + \mu_{u_j}\alpha_{u_j}(k) - 2\sigma_{u_j} \end{aligned} \tag{2.12}$$

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

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

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:

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

   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

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

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 co-efficients.

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 co-efficients are uniformly 1.
12) Next, type “[0.5;0.5;0.5;0.5]” and press enter. It will be assumed that the quadratic co-efficients 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...

$$\#1:\ \exp(-1*s) * \frac{2s+1}{s^2+3s+1}$$

$$\#2:\ \exp(-3*s) * \frac{4s}{s^2+3s+1}$$

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

Installation

The toolbox can be installed by first unzipping the files to any desired location. Next, the "solverTools" must be added to the search paths of MATLAB. This can be accomplished in the following steps:

a. Start MATLAB.
b. Go to File→Set Path.
c. A "Set Path" Window (Figure 10b) will appear. Click the button "Add with subfolders". Its location is shown by a box with the label 1 in Figure 10b. Find the folder "SolverTools" and select it. Click "OK".
d. Once MATLAB has finished, click on the "Save" button on the "Set Path" window to save the new default paths. Its location is shown by a box with the label 2 in Figure 10b.
e. The default paths are those paths that MATLAB will search to find a given function.

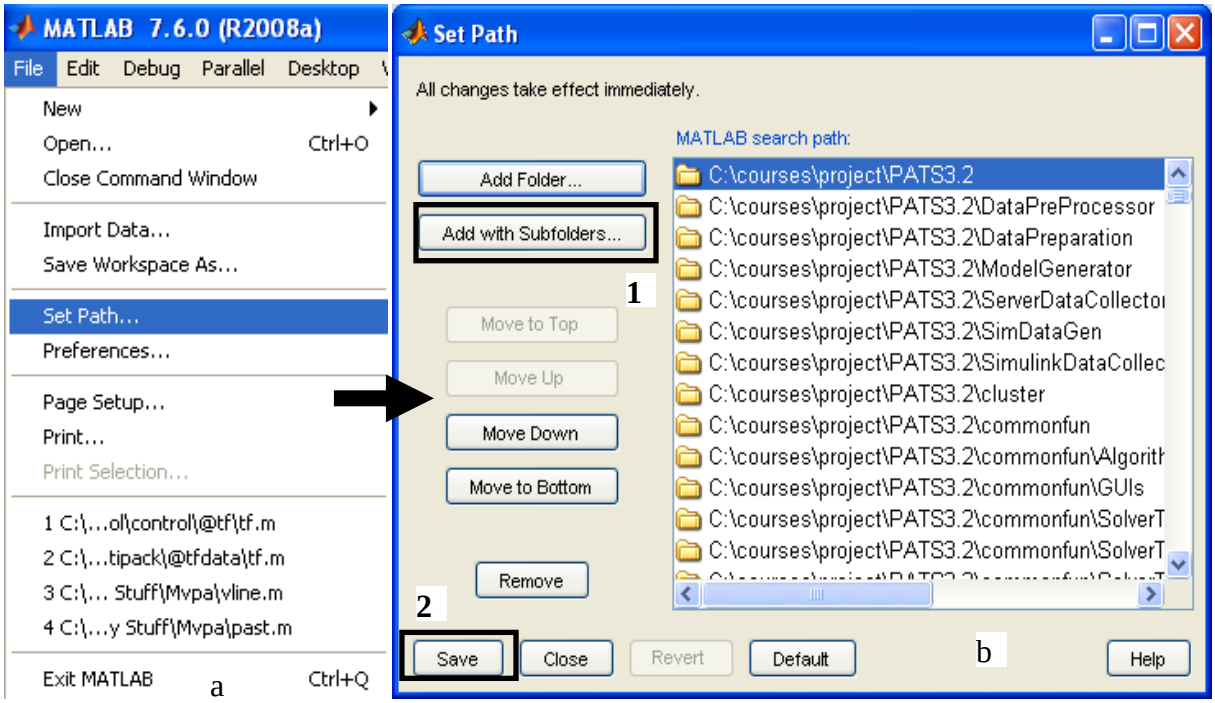

Figure 10: The steps required to add default paths in MATLAB. a) location of the "Set Path" command and b) the "Set Path" Window

Starting 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 11 should appear. This GUI is the main access to the toolbox. To start a session of the toolbox, click on the "UVPA" menu. This will bring up a new GUI, which is shown in Figure 12. In Figure 12, each of the main parts of the GUI is highlighted and will be discussed separately.

Figure 11: The First GUI that appears

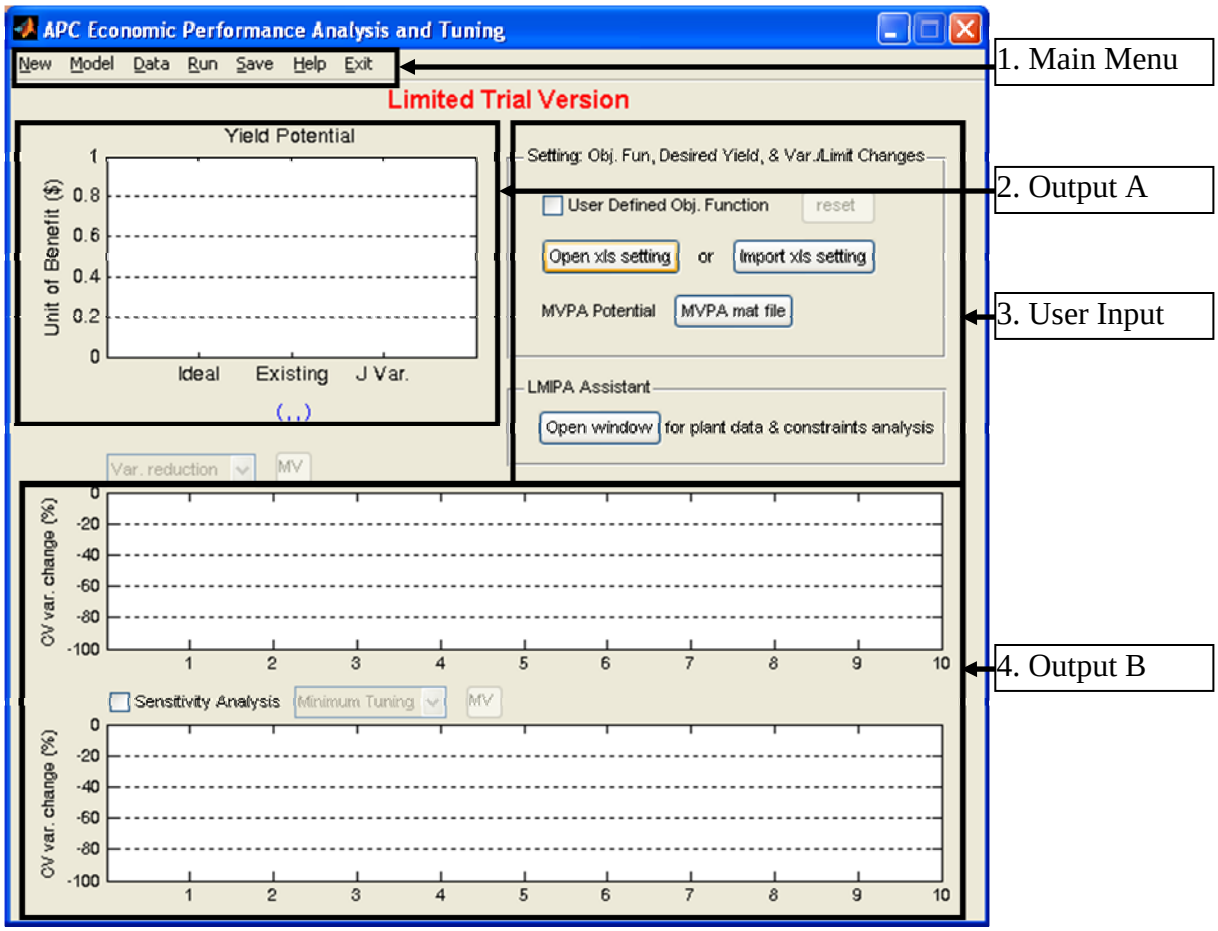

Figure 12: The main GUI for this toolbox, with the main regions highlighted

In-depth Discussion of the Toolbox

Section 1: Main Menu

A close-up of the Main Menu section is given in Figure 13. The Main Menu consists of the following 7 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) **Model**: Clicking this menu will allow the user to select the open-loop model or gain model file that will be used in the analysis.
3) **Data**: Clicking this menu allows the user to load data in the comprehensive data format.
4) **Run**: Clicking this menu will cause the toolbox to analysis the data that has been selected.
5) **Save**: Clicking this menu will allow the current analysis to be saved as a ".mat" file.
6) **Help**: Clicking this menu will allow the user to access help about using the toolbox. Currently, this is not available.

7) **Exit**: Clicking this menu will cause the toolbox to close.

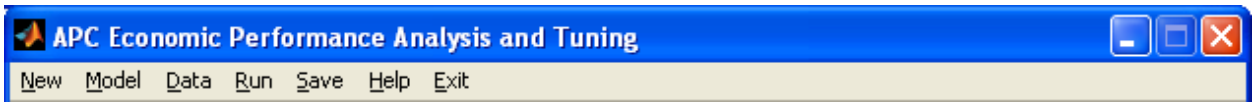

Figure 13: Close-up of the Main Menu

Section 2: Output A

In this section, the potential functions and their values are displayed. The value for each of the functions is also given.

Section 3: User Input

Here the user can enter information about the process. There are 4 distinct areas for the user to consider:

1) **User Defined Objective Function**: This allows the user to define the objective function that will be used.
2) **Open xls setting or Import xls setting**: This allows the user to enter the desired percentage changes in variance or constraint relaxation. This is entered in a spreadsheet that is shown in Figure 14. The "IsValid" column allows the user to determine whether the given variable will be used in the optimisation. It can have only one of two values: 0 (do not consider) or 1 (consider).

	A	B	C	D	E	F
1		Var. (%)	IsValid	Limit (%)	IsValid	
2	J Ratio	0		0		
3	Lambda	0		0		
4	CV1	0	1	0	1	
5	CV2	0	1	0	1	
6	MV1	0	1	0	1	
7	MV2	0	1	0	1	
8						

Figure 14: Cropped view of the spreadsheet that appears

3) **MVPA Potential**: This allows the user to import an MVPA potential .mat file, which contains the minimum variance performance indices. These values show whether or not further control can be performed for the given variables.
4) **LMIPA Assistant**: This allows the user to select the data that will be used in the analysis. Figure 15 shows the window that appears. It consists of 7 different areas:

a. **Main Menu (Section 1)**: This is the generic MATLAB menu and will not concern us here.
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 Sections (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 16, 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).

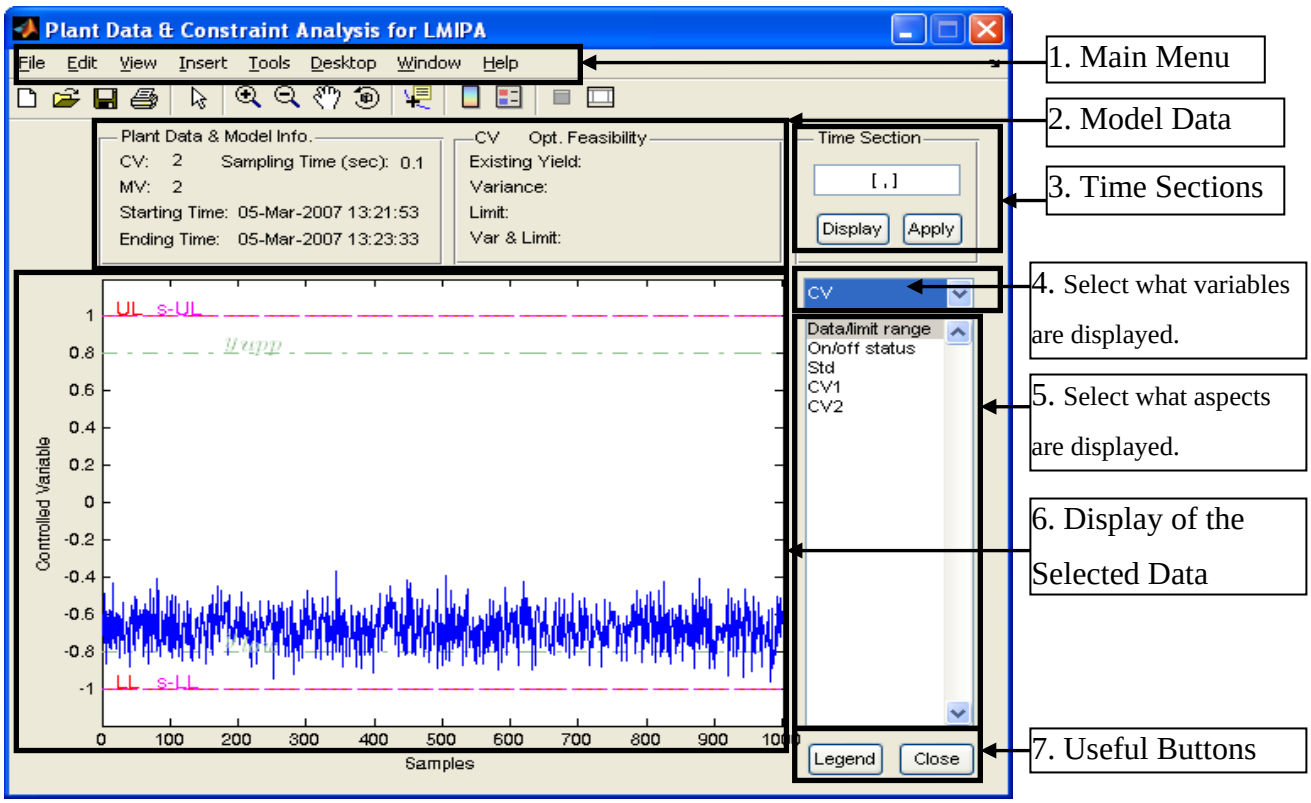

Figure 15: Plant Data & Constraint Analysis for LMIPA

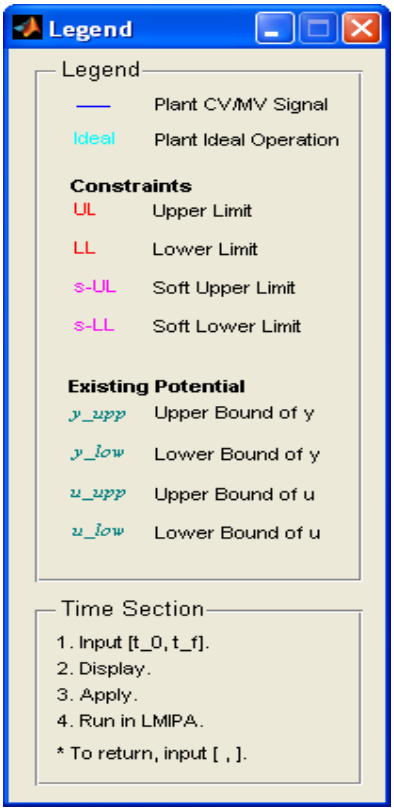

Figure 16: Explanation of the abbreviations used in the Plant Data & Constraint Analysis for LMIPA window

Section 4: Output B

This section presents 2 graphs that present the results of the analysis by the toolbox. The information above each of the 2 graphs, allows the user to:

1) Determine whether or not to perform sensitivity analysis (only over the second graph). Checking the check box will select sensitivity analysis.
2) Determine what type of control analysis is being performed. The drop-box contains the following entries: minimum tuning, limit relaxation, and variance reduction. Not all of the selections are available at a given point.
3) Cycle between controlled (CV) and manipulated (MV) variables.

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

Now, that it has been explained how to perform the calculations manually with minimal computer help, the programme that was designed to do this will be explained. 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 Menu and selecting the "**`Case_model.mat`**" file.

After a few seconds, load the data by clicking the Data Menu and selecting the "Compact" option. Select the file entitled "**`Case_Cmprhnsv_Data.mat`**". Next, click on the "Run" menu to obtain the screen shown in Figure 17.

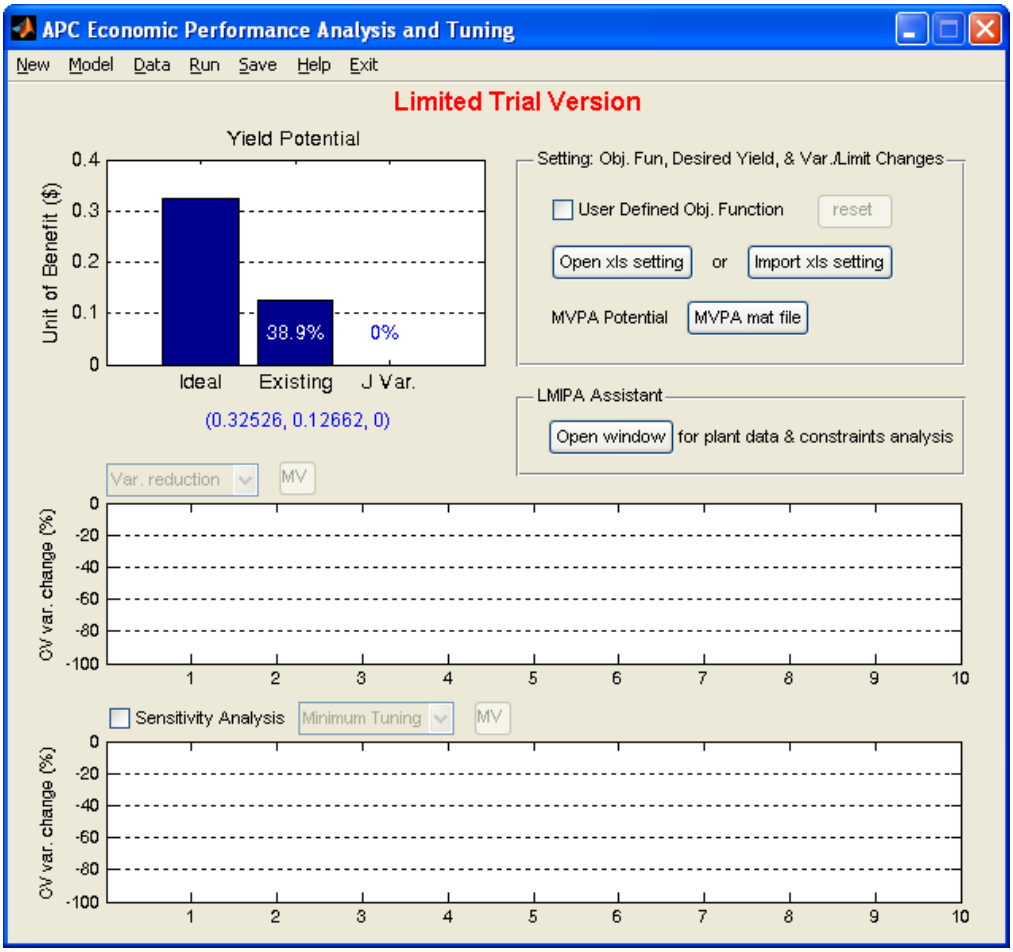

Figure 17: Screen shot 1 for the example

Figure 17 shows that the ideal benefit potential is 0.33, while the current (existing) benefit potential is 0.13, or 38.9%, of the ideal.

Now click on the “Open xls setting” button. It will be desired to see what constraint relaxation needs to be performed in order to achieve 60% of the ideal benefit potential. In order to do this, enter a value of 60 in D2 of the spreadsheet. Save the spreadsheet, or else the values are not updated. Finally, click on the “Run” menu on the main window. The results shown in Figure 18 should appear.

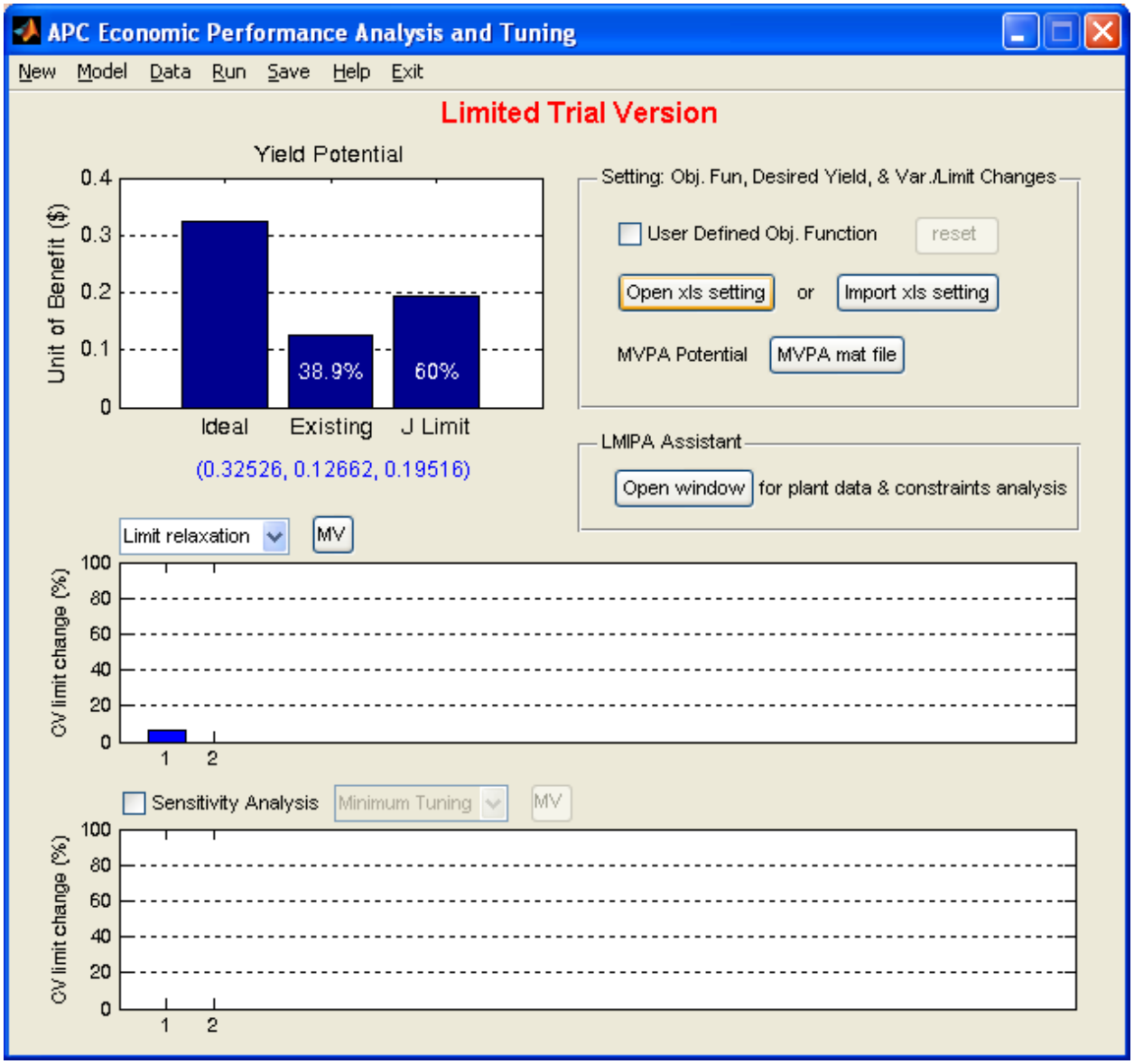

Figure 18: Screen shot 2 for the example

From Figure 18, it can be seen that in order to achieve 60% of the ideal benefit potential, the first control variable's (CV) constraint needs to be increased by about 5%. It would now be interesting to determine the sensitivity of the variables to a change. This can be accomplished by selecting the "Sensitivity Analysis" check box, which is located just above the second graph. Click on the "Run" menu to accomplish to obtain the results. Figure 19 shows the results.

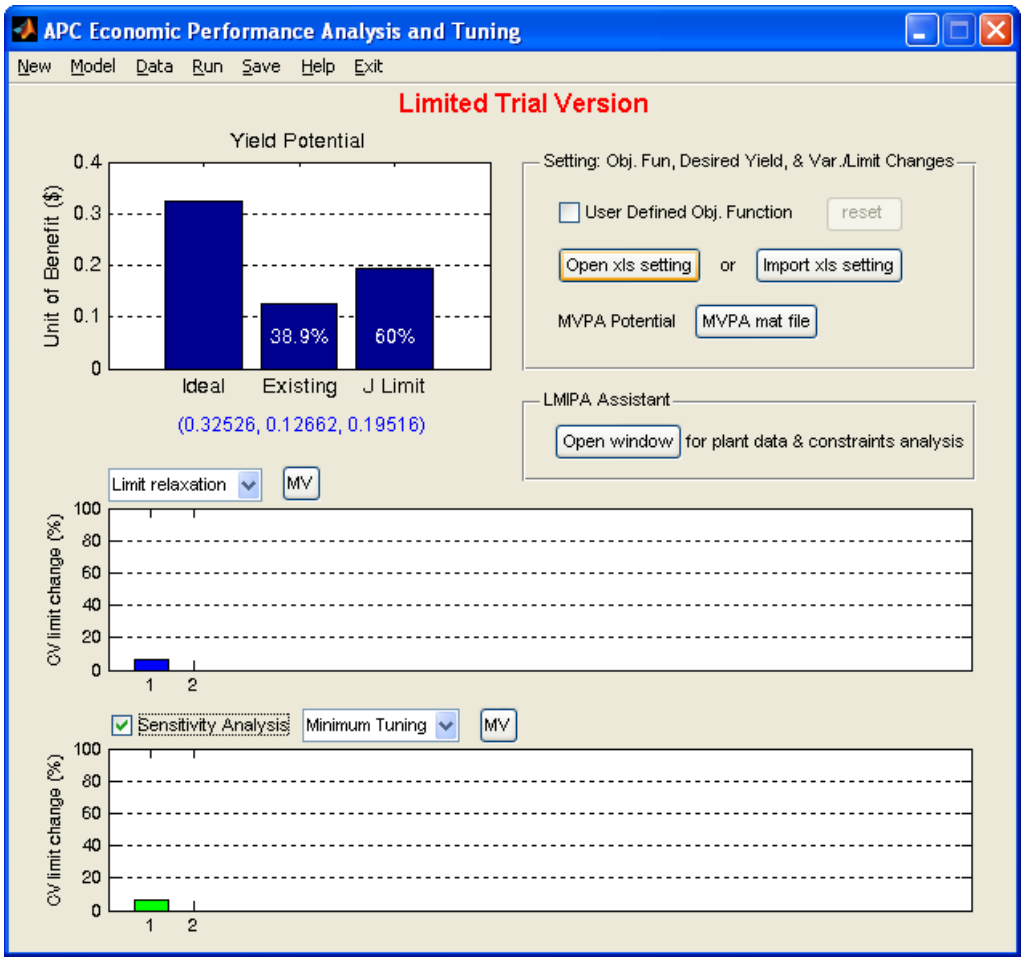

Figure 19: Screenshot for example after performing sensitivity analysis

Based on Figure 19, it can be concluded that at minimum tuning the first control variable's constraint should be increased by approximately 5%.

Another possible approach is to determine how much the benefit potential can increase if the variance is decreased by 60%. Thus, return to the Excel spreadsheet and enter zero in D2, which previously contained 60. Now enter in B2 the value of 60. Save the spreadsheet. Return to the main LMIPA GUI and press "Run". The results are shown in Figure 20. The results suggest a decrease of 60% in the overall variance will give a decrease of 40% in the variance of the first control variable. Similarly to the above, it is possible to perform sensitivity analysis by clicking the "Sensitivity Analysis" checkbox and pressing the "Run" Menu again. The results are shown in Figure 21 and suggest that the process is quite sensitive to changes.

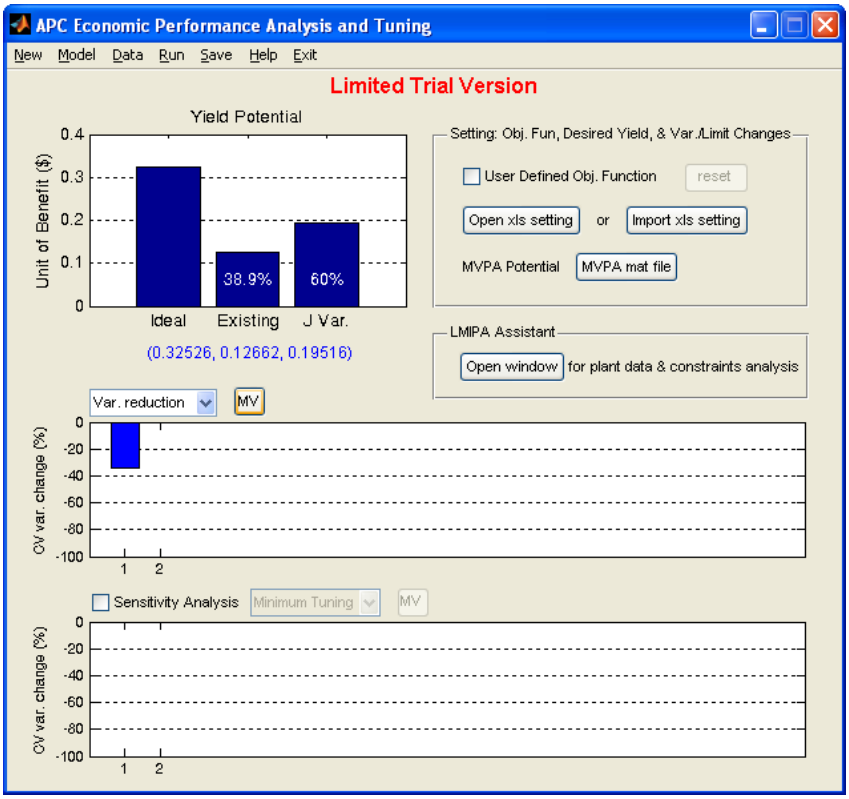

Figure 20: Screenshot after performing variance reduction

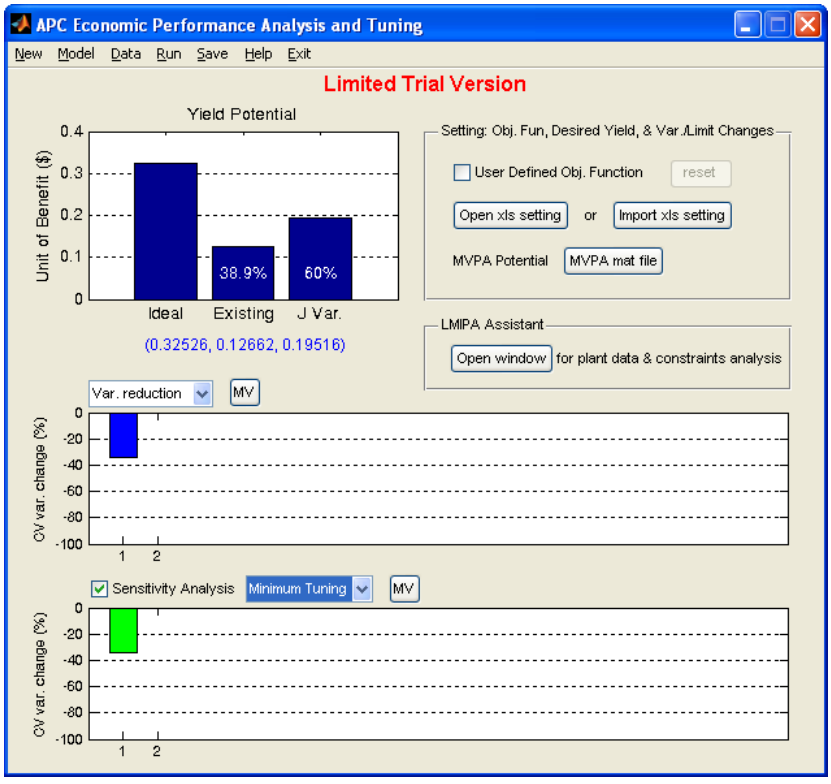

Figure 21: Screenshot of Variance Reduction and Sensitivity Analysis

Part II: Time Section in LMIPA

In order to enter time section in LMIPA, the following steps should be followed:

1) Click on the "LMIPA Assistant".
2) Go to section 3 (Time Section) of Figure 15, 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 22 should appear. Other control variables and manipulated variables can be selected. Once you have confirmed that the desired section has been selected. Click the "Apply" button, which located in section 3 below the textbox. Finally, click on "Close" button that is located in section 7 of Figure 15.

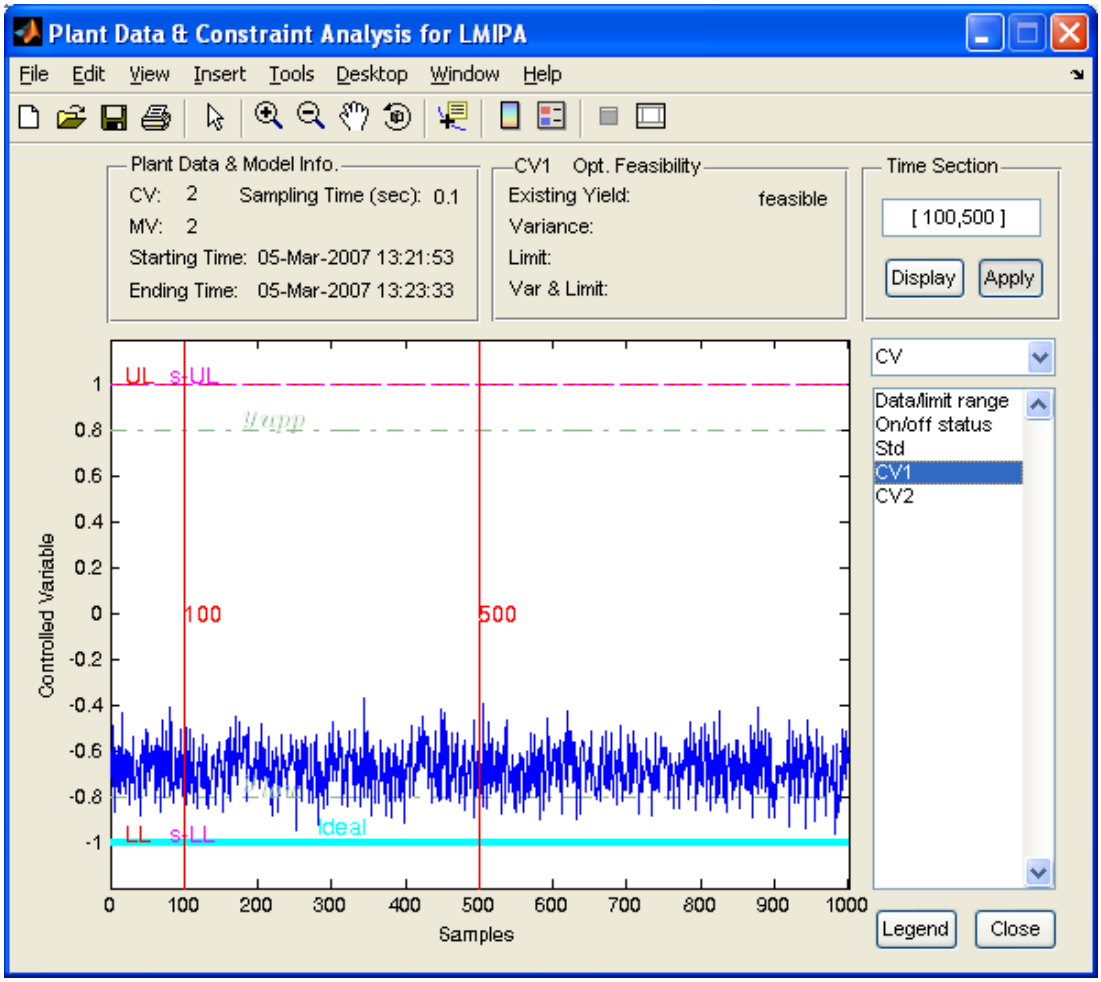

Figure 22: Showing the selected time sections for CV1

3) Click on the Run menu in the main window. After some time, the figure shown in Figure 23 should appear. As can be seen, the results are similar to those previously obtained. This should be the case in most circumstances.

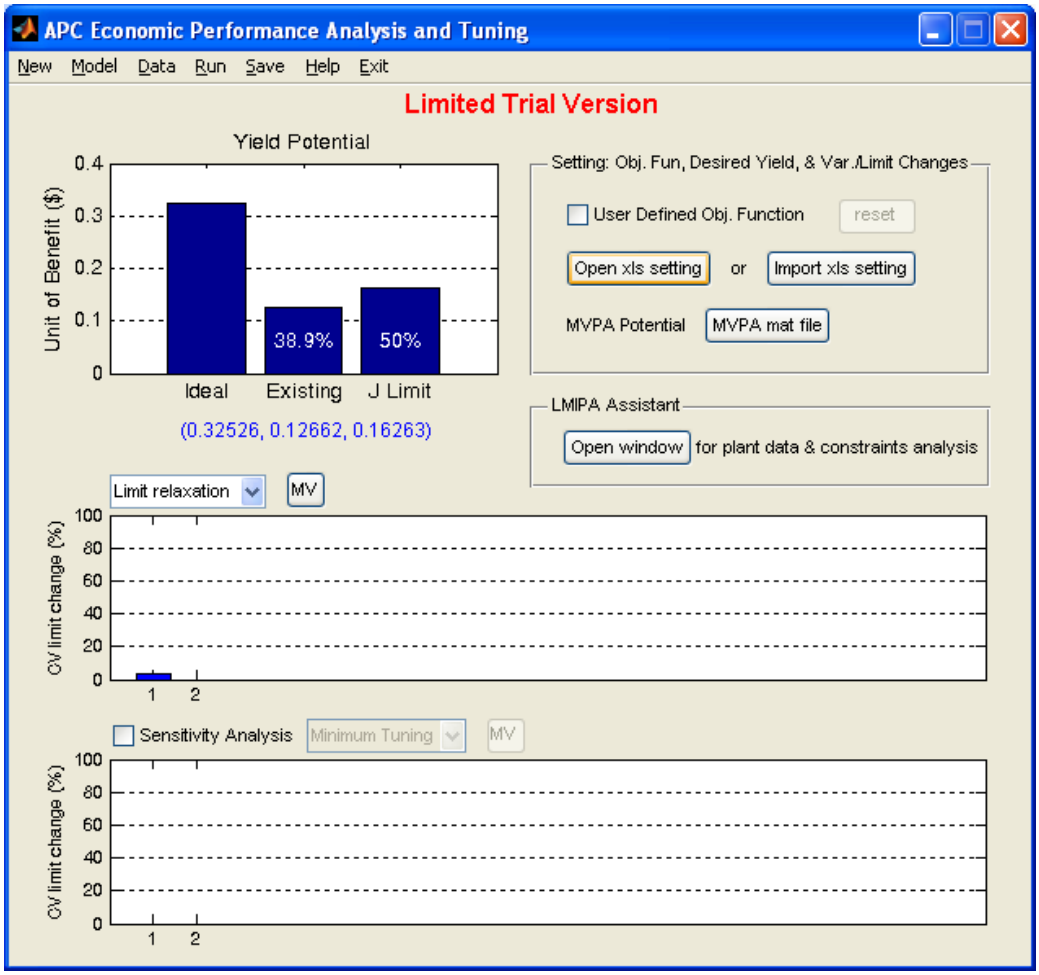

Figure 23: Results after using a given time section

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.