University of Alberta Computer Process Control Group

Closed-loop Valve Stiction Detection and Quantification

Limited Trial Version

Written by CPC Control Group, University of Alberta

Version 2.5

Table of Contents

Introduction 1

System Requirements 1

Quick Start 1

Detailed Instructions 4

- Theory 4
- Data Storage 7
  - Data Generation 8
- Optimisation Subroutine 10
- Using the Toolbox 11
  - Installation 11
  - Starting the Toolbox 12
  - In-depth Discussion of the Toolbox 12
- Examples 16
  - Part I: Using the Programme to Obtain Results 16
  - Part II: Some Other Issues Using the Toolbox: Time Sections and Downsampling 19

References 21

List of Figures

Figure 2: The main GUI for this toolbox, with the main regions highlighted 2
Figure 3: Screen shot after loading the data 3
Figure 4: Screen shot after clicking "Run" 4
Figure 5: Phase plot for the typical behaviour of valve with stiction (after (Shoukat Choudhury, Thornhill, & Shah, 2005)) 5
Figure 6: f_D as a function of f_S. The crucial regions are also highlighted (after (Lee, Ren, & Huang, 2008)). 6
Figure 7: Decision chart for determining the appropriate model (after (Lee, Ren, & Huang, 2008)) 7
Figure 10: Close-up of the main menu 12
Figure 9: The main GUI for this toolbox, with the main regions highlighted 13
Figure 11: Colour scheme used in the MSE plot 15
Figure 12: Screen shot 1 for the example 17
Figure 13: Screen shot 2 for the example 18
Figure 14: Second-order model results 19
Figure 15: Results of using a range of time sections 20

List of Tables

Table 1: Summary of the parameters generated using the "gen_cmpct_Data_Stiction" command for the first loop 9
Table 2: Summary of the parameters generated using the "gen_cmpct_Data_Stiction" command for the second loop 9

Introduction

The Closed-loop Valve Stiction Detection and Quantification toolbox is developed by the Computer Process Control Group at the University of Alberta as a tool to detect and characterise the stiction phenomenon in pneumatic control valves.

A “Quick Start” approach to use this toolbox will be presented, along with a detailed section containing step-by-step guide for users to benefit from different features of the toolbox the best.

System Requirements

In order to run this toolbox 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 quick use 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 “**`>> UA_stiction_online`**” to start the toolbox. The GUI shown in Figure 1 should appear.

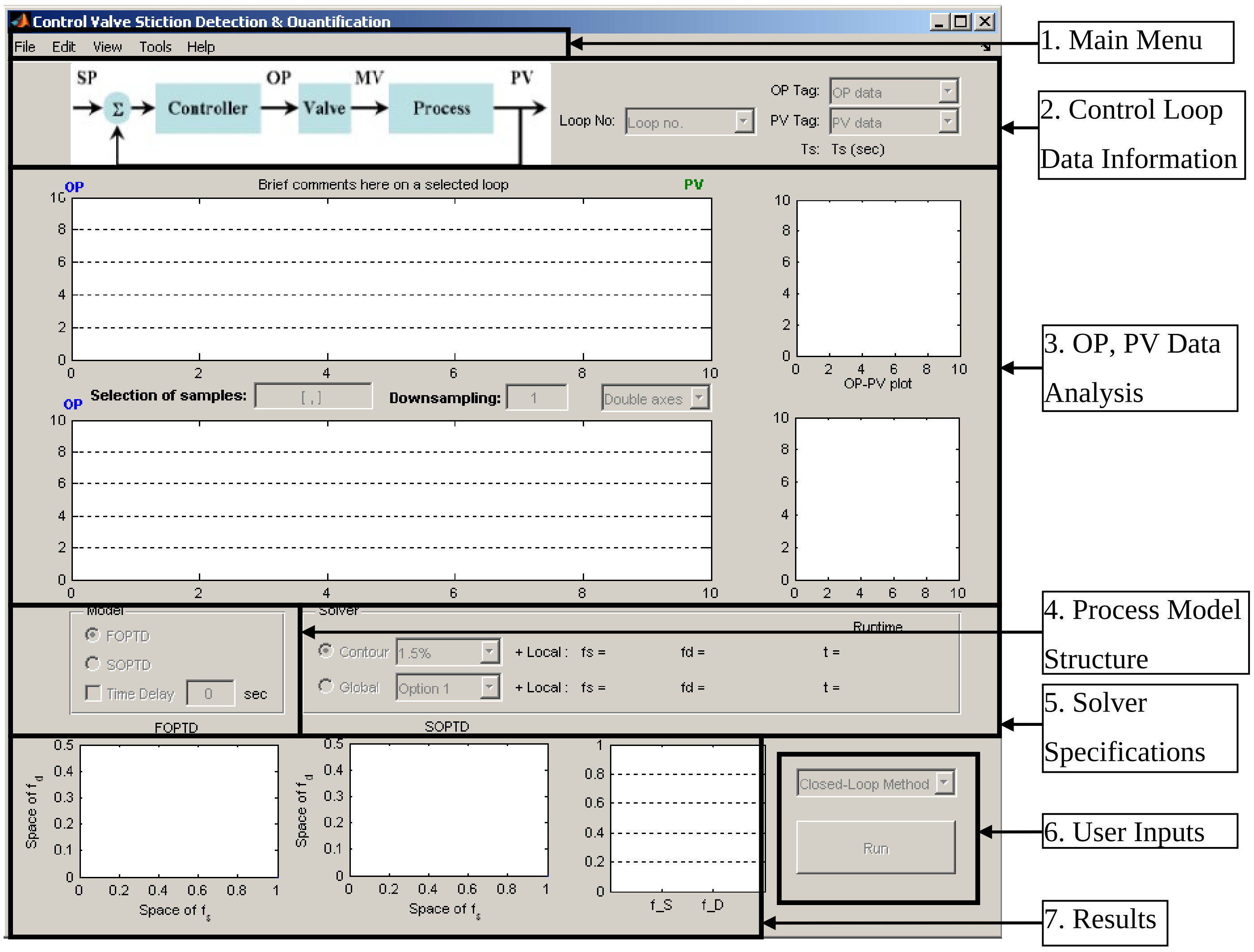
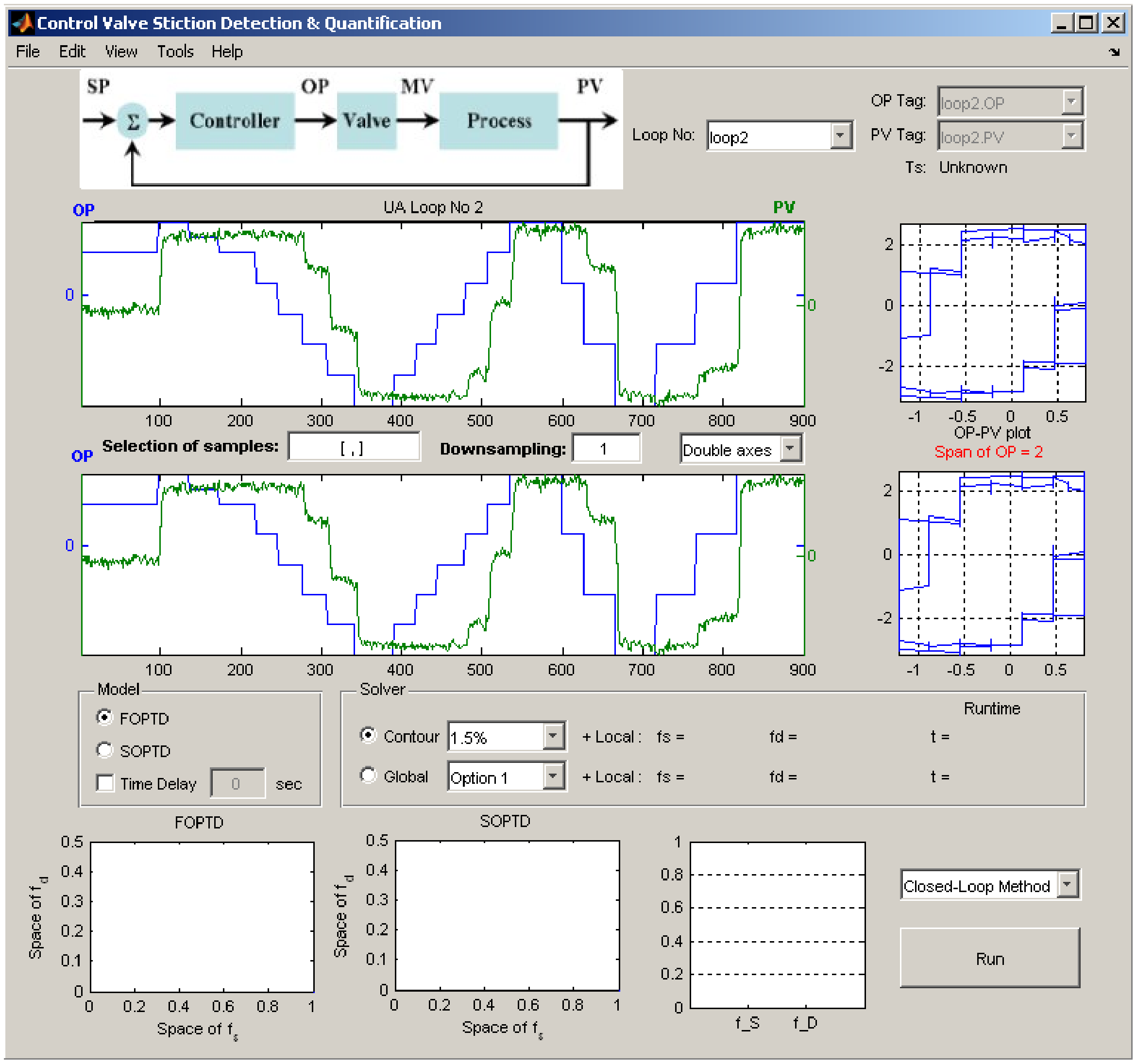


Figure 1: The main GUI for this toolbox, with the main regions highlighted

4) For the purpose of this quick start, we will be using the sample data provided in the zip format. Go to File→Open located in the Main Menu area, which is denoted as 1 in Figure 1. Select the "UA_cmpct_Data_stiction.mat" file. Be patient as the file is loaded. This file contains the closed-loop data for the process. Go to the drop box labelled "Loop No:" in Section 2 of Figure 1. Select any of the control loops in order to display the data. For this example, "Loop 4" will be selected.

5) After a few seconds, graphs of the loaded data will appear in region 3 of Figure 1. The resulting screenshot is shown in Figure 2.

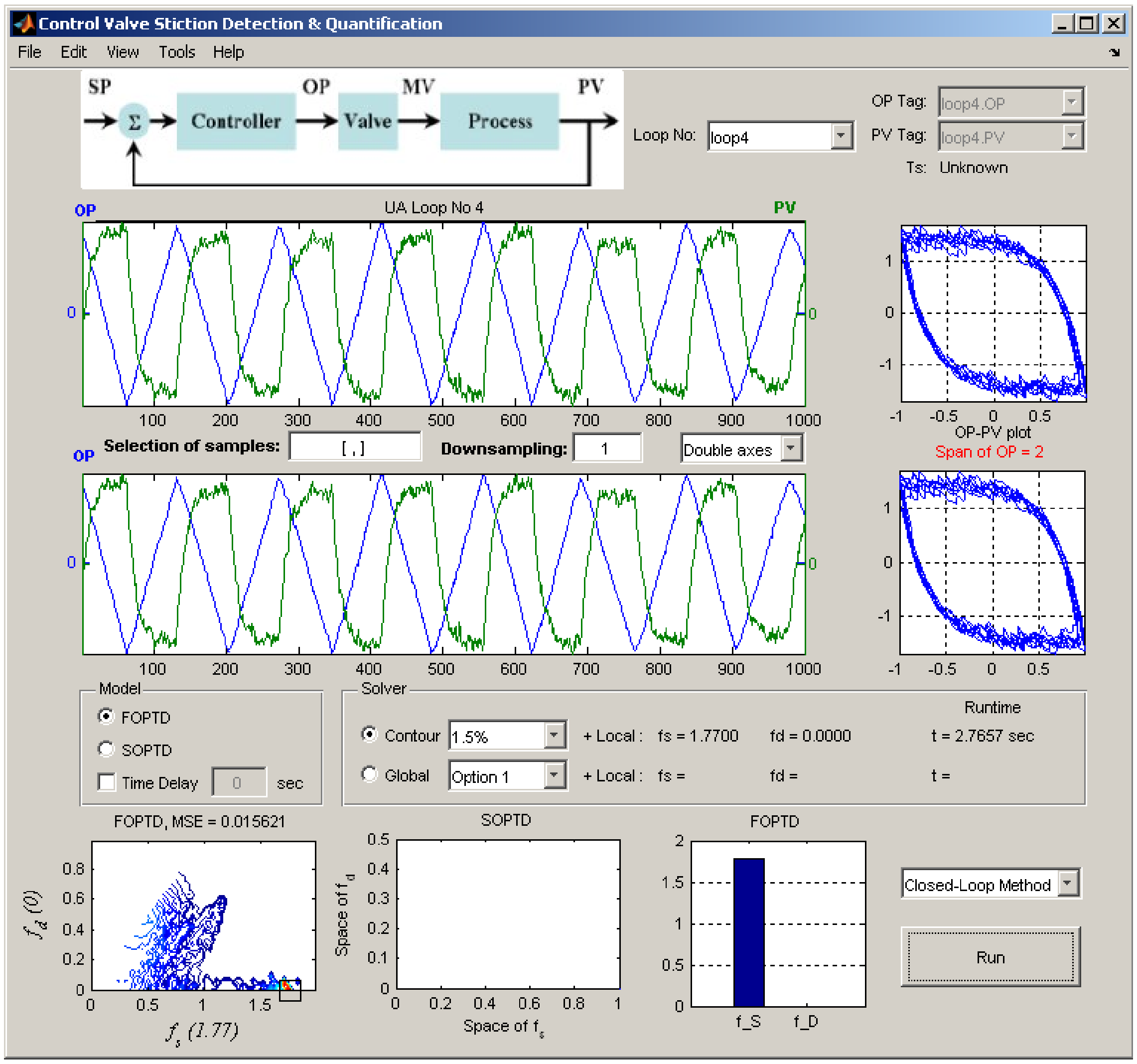


Loop No: loop2
OP Tag: loop2.OP
PV Tag: loop2.PV
Ts: Unknown
Span of OP = 2
Selection of samples: [,]
Downsampling: 1
(●) FOPTD
( ) SOPTD
[ ] Time Delay 0 sec
(●) Contour 1.5% + Local : fs = fd = t =
( ) Global Option 1 + Local : fs = fd = t =

Figure 2: Screen shot after loading the data

6) The defaults chosen by the programme will be used for the analysis. Thus, press the "Run" button located in region 6 of Figure 1.

7) Be patient as the programme determines the answer. This can take some time in order for the programme to obtain the results. After a few seconds, the screen should become similar to that shown in Figure 3.



Loop No: loop4
OP Tag: loop4.OP
PV Tag: loop4.PV
Ts: Unknown
Selection of samples: [,]
Downsampling: 1
Double axes
Model: [✓ FOPTD] SOPTD
Time Delay 0 sec
[✓ Contour] 1.5% + Local : fs = 1.7700 fd = 0.0000 t = 2.7657 sec
Global Option 1 + Local : fs = fd = t =
FOPTD, MSE = 0.015621
Closed-Loop Method

Figure 3: Screen shot after clicking "Run"

8) Figure 3 shows that there is stick and slip behaviour in this particular control loop.

9) To exit the toolbox, click the "Exit" Menu.

Detailed Instructions

Theory

Valve stiction can be defined as the presence of nonlinear behaviour in a valve (Shoukat Choudhury, Thornhill, & Shah, 2005). According to Shoukat Choudhury et al. (2005), this nonlinear behaviour can be attributed to static friction in the valve, which impedes the motion of the valve until a sufficiently large force can overcome the static friction. The typical behaviour of a valve suffering from stiction can be seen in Figure 5. It can be noted that the regions denoted by A and B are together referred to by the symbol f_S. It should be noted that slip jump behaviour can also occur in the moving phase, if the force applied to the valve is less than that required due to static friction. This minimal value is commonly given in terms of percentage of valve travel area, that is, the amount that the valve must be changed before any noticeable change occurs.

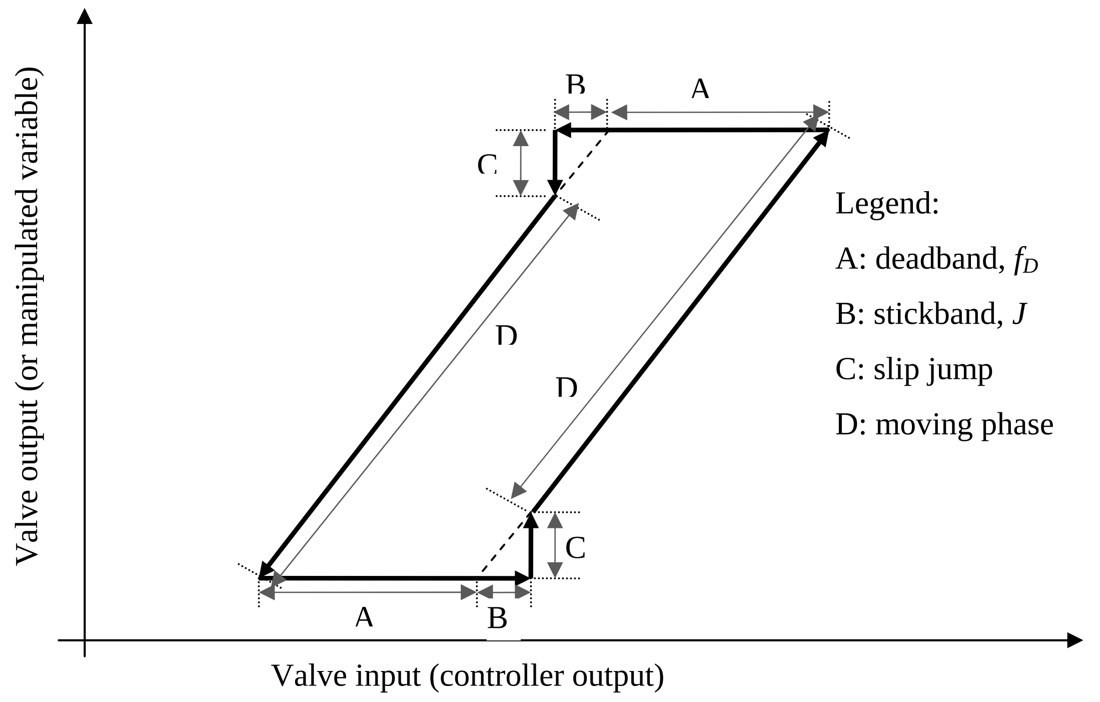


Figure 4: Phase plot for the typical behaviour of valve with stiction (after (Shoukat Choudhury, Thornhill, & Shah, 2005))

The diagram shown in Figure 5 can be constructed by plotting the valve position or manipulated variable as a function of the controller output. Furthermore, the actual curve will display a wide range of different behaviours depending on the magnitude, type, and location of stiction in the given valve, as well as the variables used as proxies for the manipulated variables and controller outputs.

The stiction model proposed by (He, Wang, Pottmann, & Qin, 2005) is used in this toolbox, in order to estimate the “best” possible values for the stiction parameters (f_S and f_D).

The algorithm used in this toolbox is based on nonlinear-systems identification as presented in (Lee, Ren, & Huang, 2008). In this algorithm, the primary graph used to analyze the data is a plot of f_D as a function of f_S. A typical plot with the crucial regions labelled is shown in Figure 6. Since the following relationship holds (Lee, Ren, & Huang, 2008),

$$f_D + f_S \leq S_0 \tag{1}$$

where S_0 is approximately given by the span of the controller output data, the bounds for the optimisation constraints can be determined using this relationship. Often, in order to determine the appropriate type of stiction for a given valve, a contour map of the mean square error (MSE) as a function of both f_D and f_S is plotted. The region with the smallest MSE represents the most probable values of f_D and f_S for the valve. Thus, this region determines what type of valve nonlinearity is most probably causing the observed oscillations.

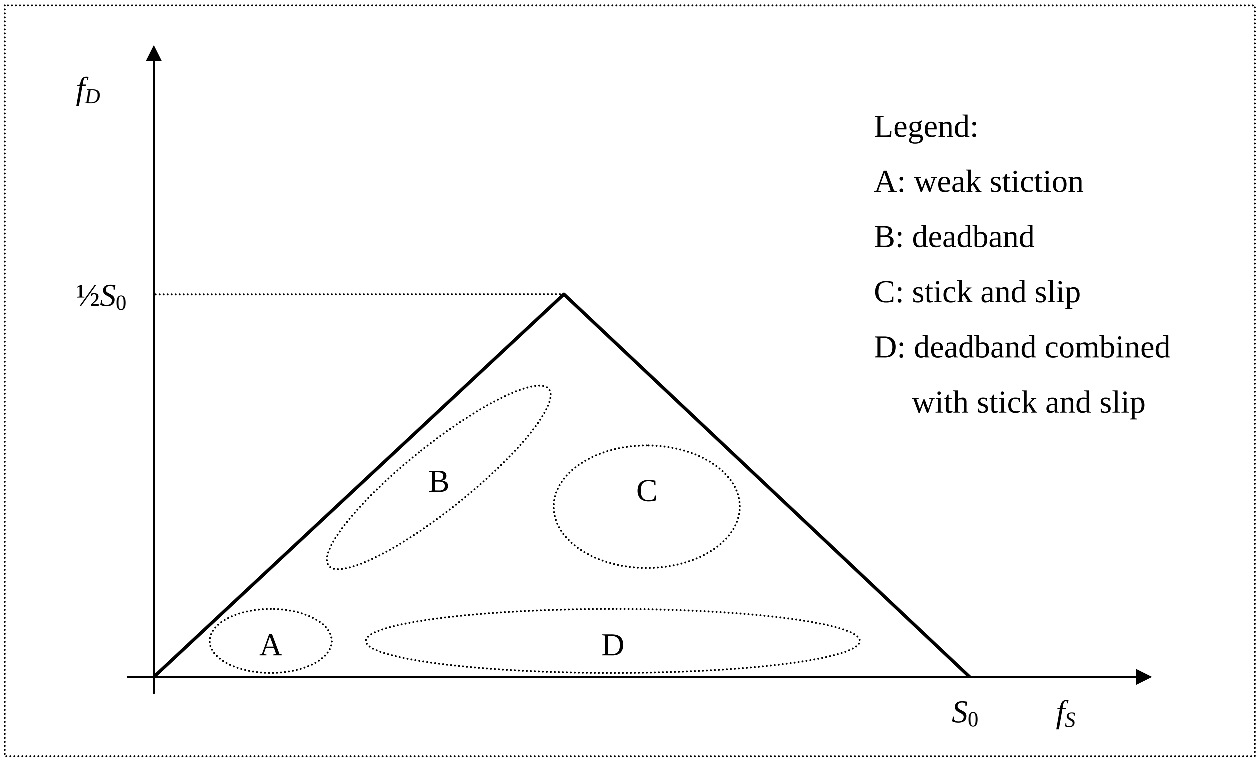


Figure 5: f_D as a function of f_S. The crucial regions are also highlighted (after (Lee, Ren, & Huang, 2008)).

A generalised model of stiction fitting can be summarised in the following 6 steps (Lee, Ren, & Huang, 2008):

1) **Data Selection**: In this step, the user makes sure that the data has *at least* 50 data points per period.
2) **Data Processing**: In this step, the data is detrended, that is, the mean of the data samples is subtracted out from the data set, in order to set the mean of the data to zero.
3) **Stiction Model Structure**: In this step, the appropriate model for a sticky valve, or valve with stiction, is selected. For the purposes of this toolbox, the model proposed by (He, Wang, Pottmann, & Qin, 2005) is used.

4) **Search the Stiction Model Space**: In this step, the appropriate bounds for the optimisation search are determined based on the controller output data.
5) **Process Model Identification**: For this step, the user is presented with two choices: either a first- or a second-order plus time delay model. The time delay can either be known or it can be determined using the method presented in (Lee, Ren, & Huang, 2008). The decision process used to pick the model is shown in Figure 6 .
6) **Quantification of Valve Stiction**: Using a specified optimisation technique, the optimal values for the parameters is determined. A coarse search is first performed. This is followed by a fine local search to determine the actual optimal points.

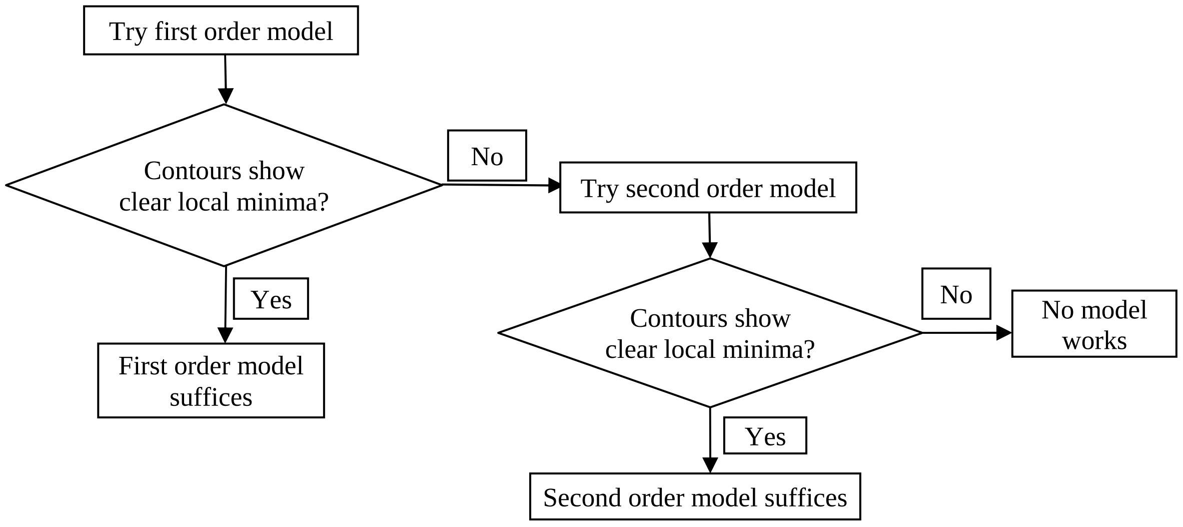

Try first order model
Contours show clear local minima?
No
Try second order model
Yes
First order model suffices
Contours show clear local minima?
No
No model works
Yes
Second order model suffices

Figure 6: Decision chart for determining the appropriate model (after (Lee, Ren, & Huang, 2008))

Data Storage

The data required for this program is stored in a single file. For a quick generation of the compact data storage objects, the file “gen_cmpct_Data_stiction.p” can be used.

The data for this process is stored as follows:

1) **int_LoopNumber**: This is an integer that stores the number of loops present in the file.
2) **str_Compact**: This is an object that stores all the data for a given loop. The data for the *i*th loop is stored in the field called “loop#”, where # represents the number of the *i*th loop. Inside each of the fields, “loop#”, the following fields are found:

a. **dbl_Compact_Data**: This is an *N*-by-2 double matrix that contains in the first column the controller output (OP) data, while the second column contains the controlled variable (PV) data. There are a total of *N* sample data points.
b. **dbl_SamplingTime**: This is a double value that contains the sampling time for the process.
c. **cell_char_TagList**: This is a 2-by-1 cell array that contains the tag list for the controller output and controlled variable.
d. **cell_char_TimeStamp**: This is an *N*-by-1 cell array that contains the time stamp for each of the *N* sample data points.
e. **cell_Comment**: A cell array that contains any user specified comments about the given loop.

Data Generation

The data storage files for multivariate analysis can be generated using the *p*-file "gen_cmpct_Data_stiction.p". The following steps should be followed to create a compact data set for a process with 3 sample points and 2 control loops with a sampling time of 3 s.

1) Start MATLAB and point the directory to the location of the above binary file.
2) At the command prompt create the following 2 data matrices for the first loop, which contains the 3 sample values of the process for the controller output and controlled variable "**`>> OP1=[1;3;5]`**" and ""**`>> PV1=[2;4;6]`**". For the second loop, create "**`>> OP2=[3;7;9]`**" and ""**`>> PV2=[2;6;8]`**".
3) Type at the command prompt: ""**`>> gen_cmpct_Data_stiction`**". The following should appear

```
Consider a control-loop with control valve
1. Input data (OP): N x 1 matrix
2. Output data (PV): N x 1 matrix
3. Sampling time (sec)
4. Comment on the control loop
How many control loops?
```

4) Type "2", since there are 2 control loops. Press "Enter"
5) Now, the programme will prompt for the data for the first loop. Thus, type "OP1" and press enter. This is the sampled controller output matrix. Next, type "PV1" and press

enter. This is the sampled controlled variable data. Finally, type “3” and press enter. This is the time delay. Press enter again, since we do not wish to enter any comment for the loop.

6) Repeat Step 5 for the second control loop, replacing OP1 with OP2 and PV1 with PV2.
7) Enter a name for the 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**.

In order to view the results, type “`>>load test_model.mat`” followed by “`>>int_LoopNumber`” to view the number of loops, which should be 2. In order to view the results for the first loop, type “`>>str_Compact.loop1`”, which will display all the fields for the given loop. In order to view the results in the individual loops, type the above followed by the desired string, for example, “`>>str_Compact.loop1.dbl_Compact_Data`”. The results should be identical to that displayed in Table 1 for the first loop and in Table 2 for the second loop.

Table 1: Summary of the parameters generated using the "gen_cmpct_Data_Stiction" command for the first loop

Entry	Value
dbl_Compact_Data	[1 2 3 4 5 6]
dbl_SamplingTime	3
cell_char_TagList	{'loop1.OP' 'loop1.PV'}
cell_char_TimeStamp	It should give the current time incremented by 1, 2, and 3
char_Comment	‘’

Table 2: Summary of the parameters generated using the "gen_cmpct_Data_Stiction" command for the second loop

Entry	Value
dbl_Compact_Data	[3 2 7 6 9 8]
dbl_SamplingTime	3
cell_char_TagList	{'loop2.OP' 'loop2.PV'}
cell_char_TimeStamp	It should give the current time incremented by 1, 2, and 3
char_Comment	‘’

Optimisation Subroutine

The optimisation routines presented in this toolbox are self-contained that is they can be run without the need for the installation of any additional MATLAB programmes. However, this toolbox provides users with the ability to provide their own optimisation functions. In order to do this, it is necessary to modify the file “`conopt.m`”.

The file that comes with the toolbox has been modified to run with the optimisation software provided by TOMLAB (http://tomopt.com/tomlab/), which is called “lgo”. It should be noted that this software is not free and a license must be obtained in order to use it. A free, 21-day trial for the software is available

The general format for the file consists of the header file and a global declaration of the variables that are required for optimisation. For example, the following would be the start of the file:

```
function [fs fd runtime] = conopt(option)
 % optimization for valve stiction quantification
 global model d N u y u_v y_h fs_max a b V
% The required code to run the programme would then be entered.

% end conopt
```

For example, the conopt.m file for the TOMLAB optimisation software would be given as (comments have been added to explain what is required):

```
function [fs fd runtime] = conopt(option)
 % optimization for valve stiction quantification
 global model d N u y u_v y_h fs_max a b V
% TOMLAB lgo (http://www.tomopt.com
%Initialisation of the required variables for the programme
%x_L is the lower bound for the 2 parameters that are being
optimised.
%x_U is the upper bound.
%All the variables initialled as ‘[]’ are empty.
x_L = [0 0];
x_U = [fs_max fs_max/2];
name = [];
A = [-1 1;1 1];
b_L = [-fs_max 0];
```

```
b_U = [0 fs_max];
c = [];
c_L = [];
c_U = [];
x_0 = [];
% Note that 'objfun' includes the penalty functions for constraints
% so it can also be handled as an unconstrained problem
%The next few steps create the handles to allow the optimisation to
be run.
Prob = glcAssign('objfun',x_L,x_U,name,A,b_L,b_U,c,c_L,c_U,x_0);
% Set the maximum number of iterations
Prob.LGO.options.G_maxfct = 10000;
% Select the appropriate optimisation mode based on the user's
selections.
if option == 1     % Multi-start adaptive random search
    Prob.LGO.options.opmode = 3;
elseif option == 2 % Adaptive random search
    Prob.LGO.options.opmode = 2;
elseif option == 3 % Branch and bound search
    Prob.LGO.options.opmode = 1;
end
% Run the optimisation programme to obtain the results.
Result = tomRun('lgo',Prob,1);
% Postprocessing of the results, so that the Stiction toolbox can
understand what has been done.
%Thus, it is required to store final values for fs, fd, and runtime,
which is the amount of time required to complete the optimisation.
x_opt = Result.x_k;
fs = x_opt(1);
fd = x_opt(2);
runtime = Result.REALtime;
% END of conopt.m
```

Using the Toolbox

Installation

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

Starting the Toolbox

The toolbox can be accessed from MATLAB using the following sequence of commands. First MATLAB itself should be started and the directory pointed to the folder containing the files for this toolbox. Then, at the command prompt type "`>>UA_stiction_online`". The GUI shown in Figure 8 should appear. In Figure 8, the main parts of the GUI are highlighted and will be discussed separately.

In-depth Discussion of the Toolbox

Section 1: Main Menu

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

1) **Open**: Allows the user to load the desired data into the toolbox.
2) **Exit**: Causes the toolbox to close.
3) **Undo**: Cancels the last actions done by user.
4) **Full Screen/ Window View**: Switches between window and full screen views.
5) **Help**: Contains links to the manual, updates and general information about the software.

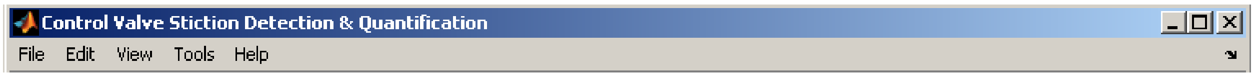


Figure 7: Close-up of the main menu

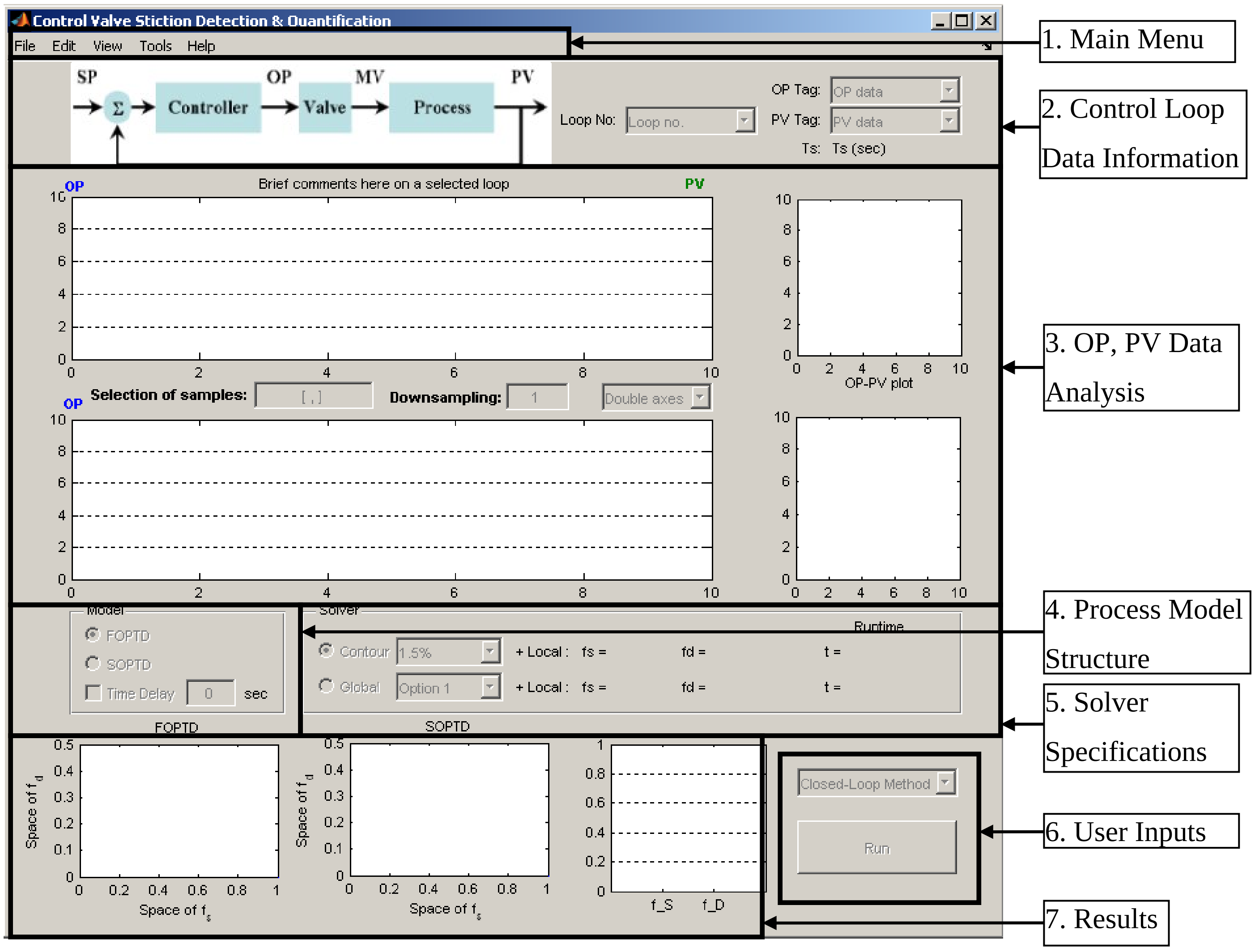


Figure 8: The main GUI for this toolbox, with the main regions highlighted

Section 2: Control Loop Data Information

In this section, information about the process is presented. Going from left to right, the following features can be seen:

1) **Process Diagram**: This is a diagram that explains the meaning of the symbols used. The following abbreviations are used:
   a. **SP**: The setpoint
   b. **OP**: The controller output
   c. **MV**: The manipulated variable
   d. **PV**: The controlled variable, or the output.

2) **Loop Selection Dropdown box**: This dropdown box allows the user to select which loop, if more than one loop is present in the data file, to be used in the analysis of the process.
3) **Process Information**: The last section, which consists of three drop-down boxes, displays the names assigned to the variables, as well as the sampling time (Ts).

Section 3: OP, PV Data Analysis

In this area, the graphs display the data that has been loaded into the toolbox. There are two separate plots of the data. The first plot is a simple time-series plot of the data, while the second plot is similar to the phase plots given in Figure 5. The first data shows the results for the unadulterated data, while the second set of graphs shows the results after changing the downsampling value.

In between the two graphs, there are three fields that can be used to change the values. The first area, called “Selection of samples,” allows the user to select what time sections are used in the analysis. The format for this entry is “[start sample value, end sample value]”. A comma must separate the two values and the end value must be greater than the start value. As well, there must exist a sufficient number of data samples in order for the computer to estimate a model. Thus, for example, to display only the data between sample times 20 and 400, the entry would be “[20,400]”. Multiple time sections are not supported. Once a value has been entered, the “enter” button should be pressed to register the changes.

The second area, called “Downsampling”, allows the user to reduce the number of data points used in the analysis. It should be noted that downsampling can result in a frequency that leads to aliasing. The value can be any real number greater than 1, but less than the number of data samples present in the process. Once a value has been entered, the “enter” button should be pressed to register the changes.

The third area is a dropdown box that allows the user to select whether the data is displayed on a single or double (separate) axis. The double axis should be used if the process data used contains two widely different values and ranges.

Section 4: Process Model Selection

In this section, the user can pick between a first-order plus dead time model (FOPDT) or a second-order plus dead time model (SOPDT). If the sampling time for the process has been specified, then it is possible to manually select a time delay for the process. Otherwise, the time delay is calculated as described in (Lee, Ren, & Huang, 2008).

Section 5: Solver Specifications

In this section, the user can select which global solver is used for coarse process optimisation, which can be selected by clicking the radio button beside the desired option. There are 2 main choices:

1) **Contour**: This option allows the user to select what contour fineness is used in the coarse search to determine the location for a finer, local search. The percentages represent what fraction of the **range of the controlled variable** is used. The smaller the percentage the more contours that are calculated and the longer the programme will take to determine the answer.
2) **Global**: This option allows the user the option to select any optimisation method. The method used to implement this is described in the section "Optimisation Subroutine."

After the programme has been run, the values for f_S, f_D, and the runtime are displayed in the same section.

Section 6: User Inputs

In this section, the user can select the method used to solve the problem from the dropdown box, or click the "Run" button to run the programme and obtain the desired results.

Section 7: Results

In this section, the three graphs show the results of the optimisation routine. The first two graphs show the contour plots as a function of f_S and f_D. The colours used are explained in Figure 9.

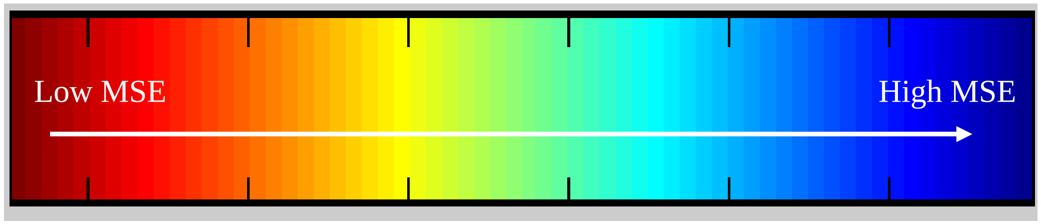


Figure 9: Colour scheme used in the MSE plot

The suggested optimal region is shown by a rectangular box. The smallest mean square error (MSE) is shown, while the corresponding values of f_S and f_D are given in brackets in the axes labelling. It should be noted that the first graph is only used for the first-order plus time delay model, while the second graph is used for the second-order plus time delay model. The third graph is a histogram that shows the value of the optimal values for f_S and f_D.

Examples

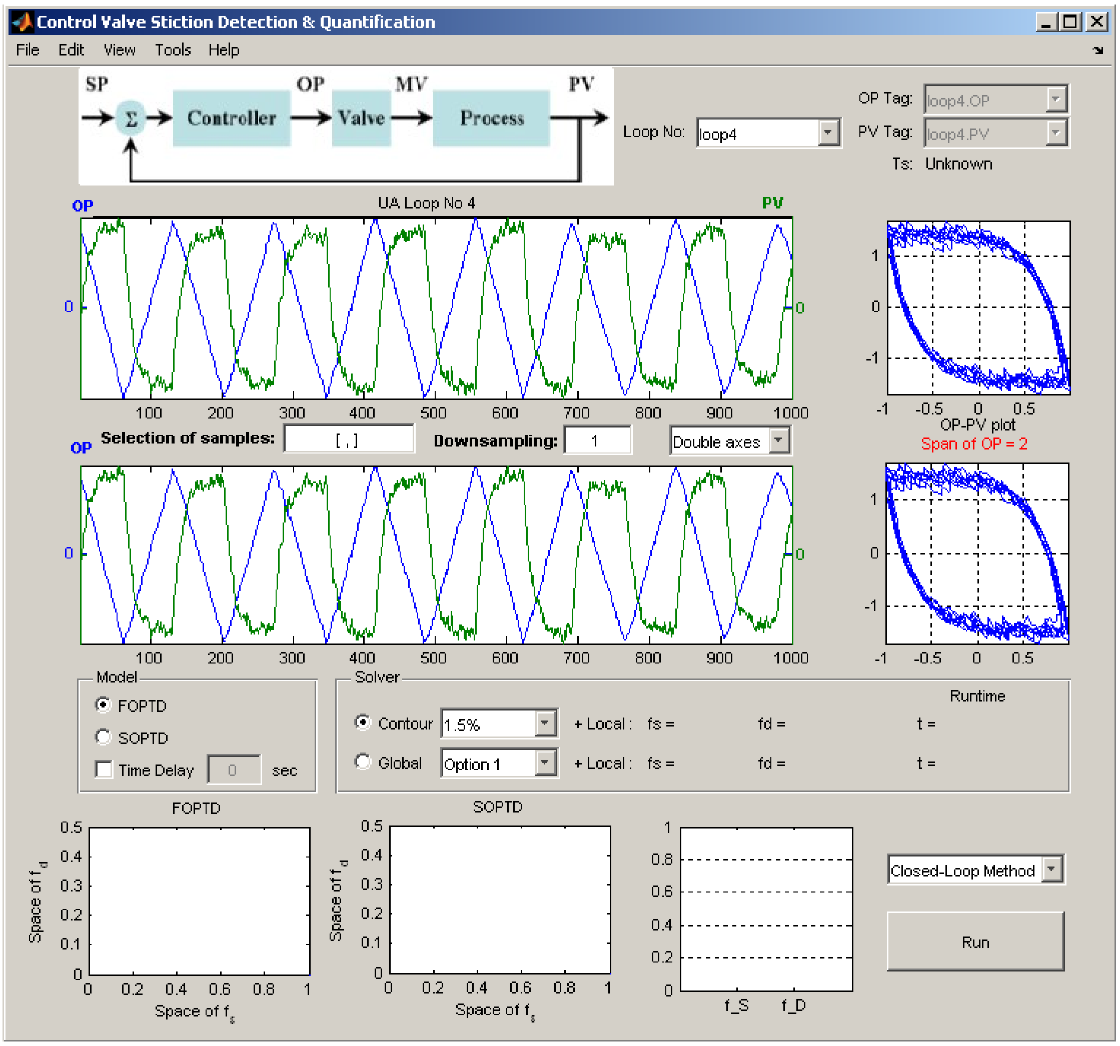
Part I: Using the Programme to Obtain Results

The first step is to load the programme, "**`>>UA_stiction_online`**".

Next, go to File→Open and select the file "UA_cmpct_Data_stiction.mat". Finally, click on "Open" to load the file. Next, go the section 2 and select the fourth loop (loop4). The GUI should look like that shown in Figure 10. All other defaults will be used.

Once it has been verified that Figure 10 matches the screen that was obtained, click on the "Run" Button and patiently wait until the results are displayed. The results take about 30 seconds to display. The screen should then resemble Figure 11.

From Figure 11, it can be seen that using a first order model, there seems to be strong deadband with stick and slip behaviour (see Figure 6). Adopting a second-order model, where the results are shown in Figure 12, does not change the results. Basically, the same parameters are obtained.



Loop No: loop4

OP Tag: loop4.OP

PV Tag: loop4.PV

Ts: Unknown

Selection of samples: [,]

Downsampling: 1

Double axes

[✓ FOPTD]

SOPTD

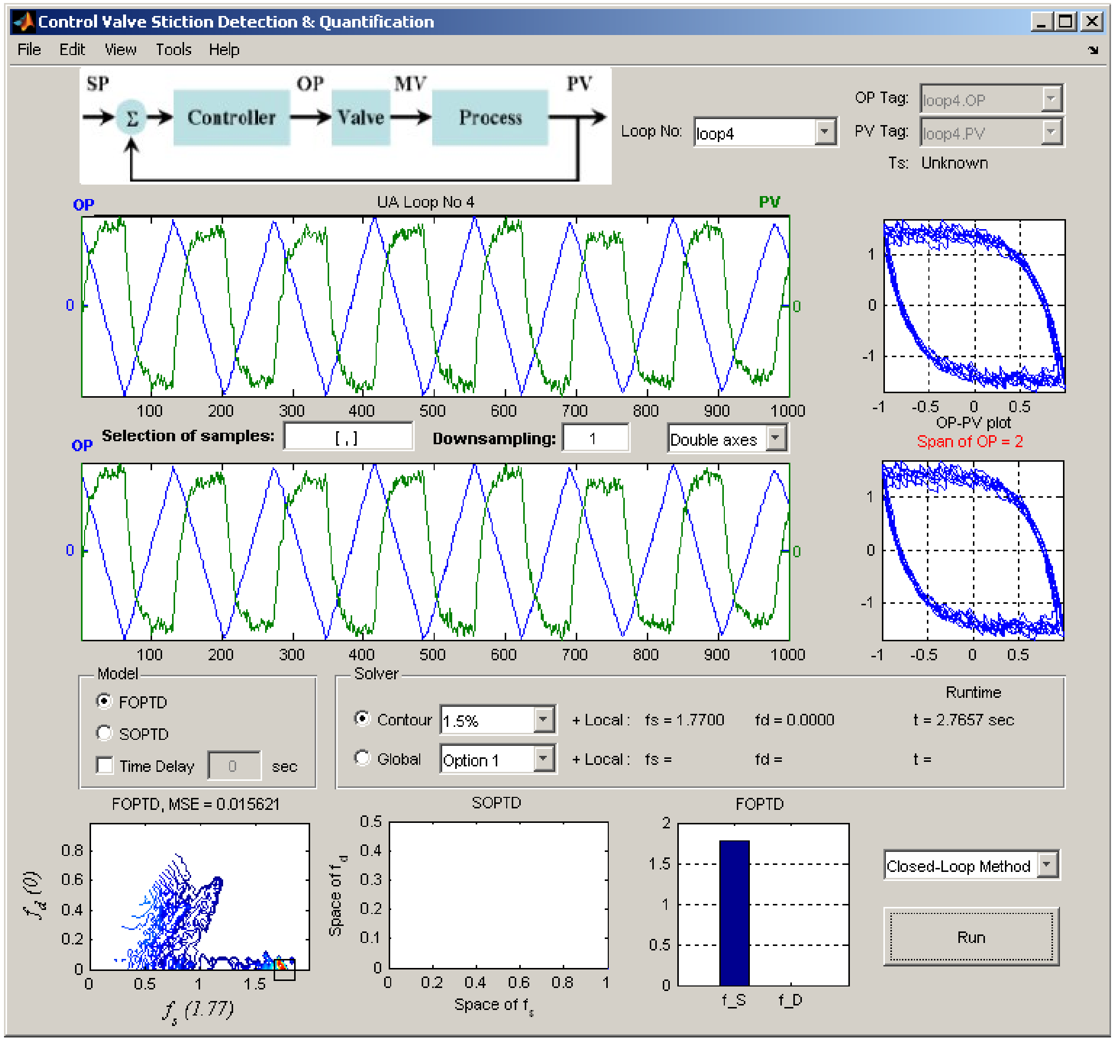
Time Delay 0 sec

[✓ Contour] 1.5% + Local : fs = fd = t =

Global Option 1 + Local : fs = fd = t =

Closed-Loop Method

Figure 10: Screen shot 1 for the example



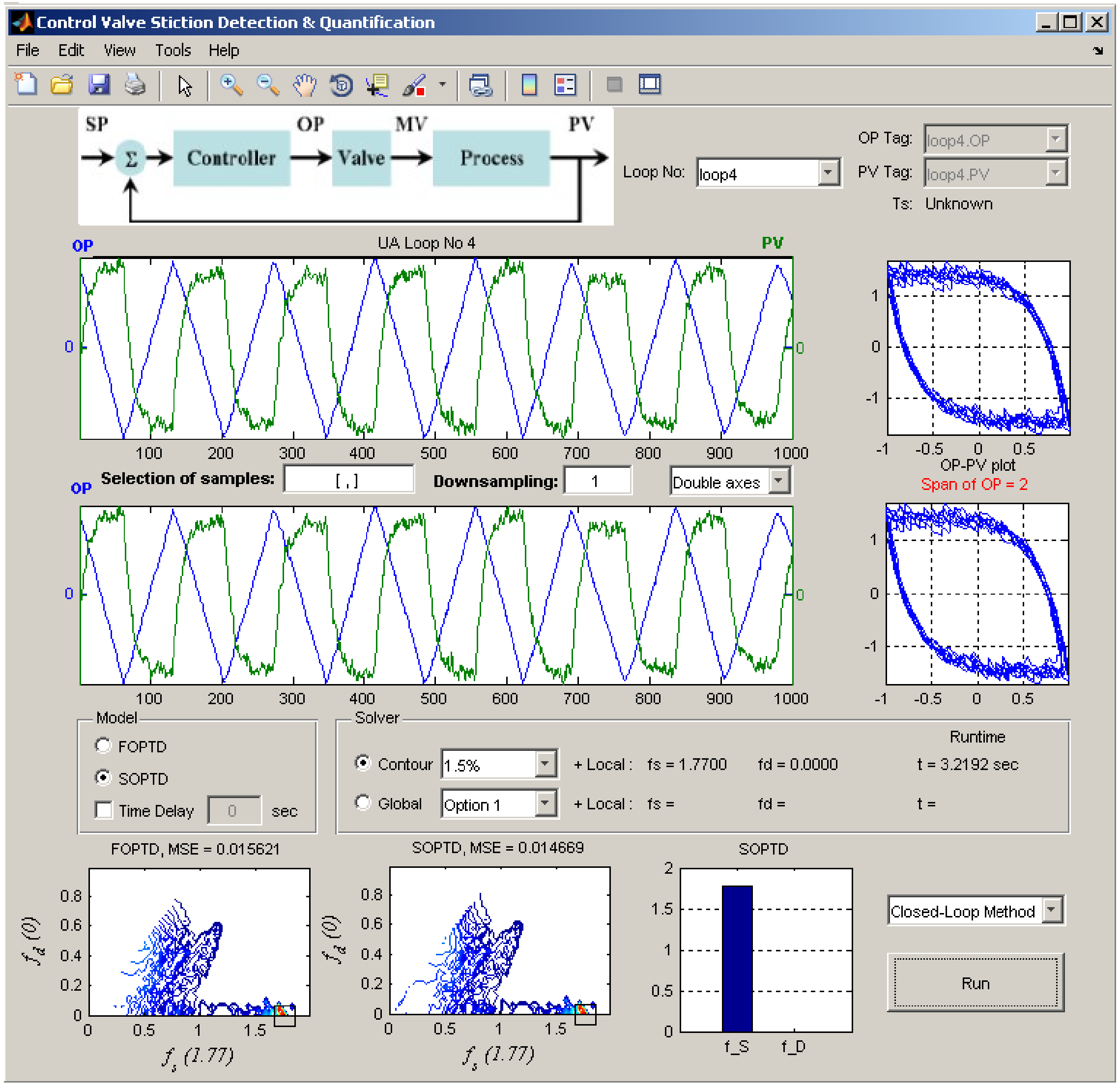
Loop No: loop4
OP Tag: loop4.OP
PV Tag: loop4.PV
Ts: Unknown
Span of OP = 2
Selection of samples: [,]
Downsampling: 1
Double axes

Model
[✓ FOPTD]
SOPTD
Time Delay 0 sec

Solver
[✓ Contour] 1.5% + Local : fs = 1.7700 fd = 0.0000 Runtime t = 2.7657 sec
Global Option 1 + Local : fs = fd = t =

FOPTD, MSE = 0.015621
$f_d\ (0)$
$f_s\ (1.77)$
Closed-Loop Method

Figure 11: Screen shot 2 for the example


Control Valve Stiction Detection & Quantification
File Edit View Tools Help
SP OP MV PV
Σ Controller Valve Process
UA Loop No 4
OP
PV
OP-PV plot
Span of OP = 2
f_d (0)
f_s (1.77)
SOPTD
f_S f_D

Loop No: loop4
OP Tag: loop4.OP
PV Tag: loop4.PV
Ts: Unknown

Selection of samples: [,]
Downsampling: 1
Double axes

Model
FOPTD
[✓ SOPTD]
Time Delay 0 sec

Solver
[✓ Contour] 1.5% + Local : fs = 1.7700 fd = 0.0000 Runtime t = 3.2192 sec
Global Option 1 + Local : fs = fd = t =

FOPTD, MSE = 0.015621
SOPTD, MSE = 0.014669

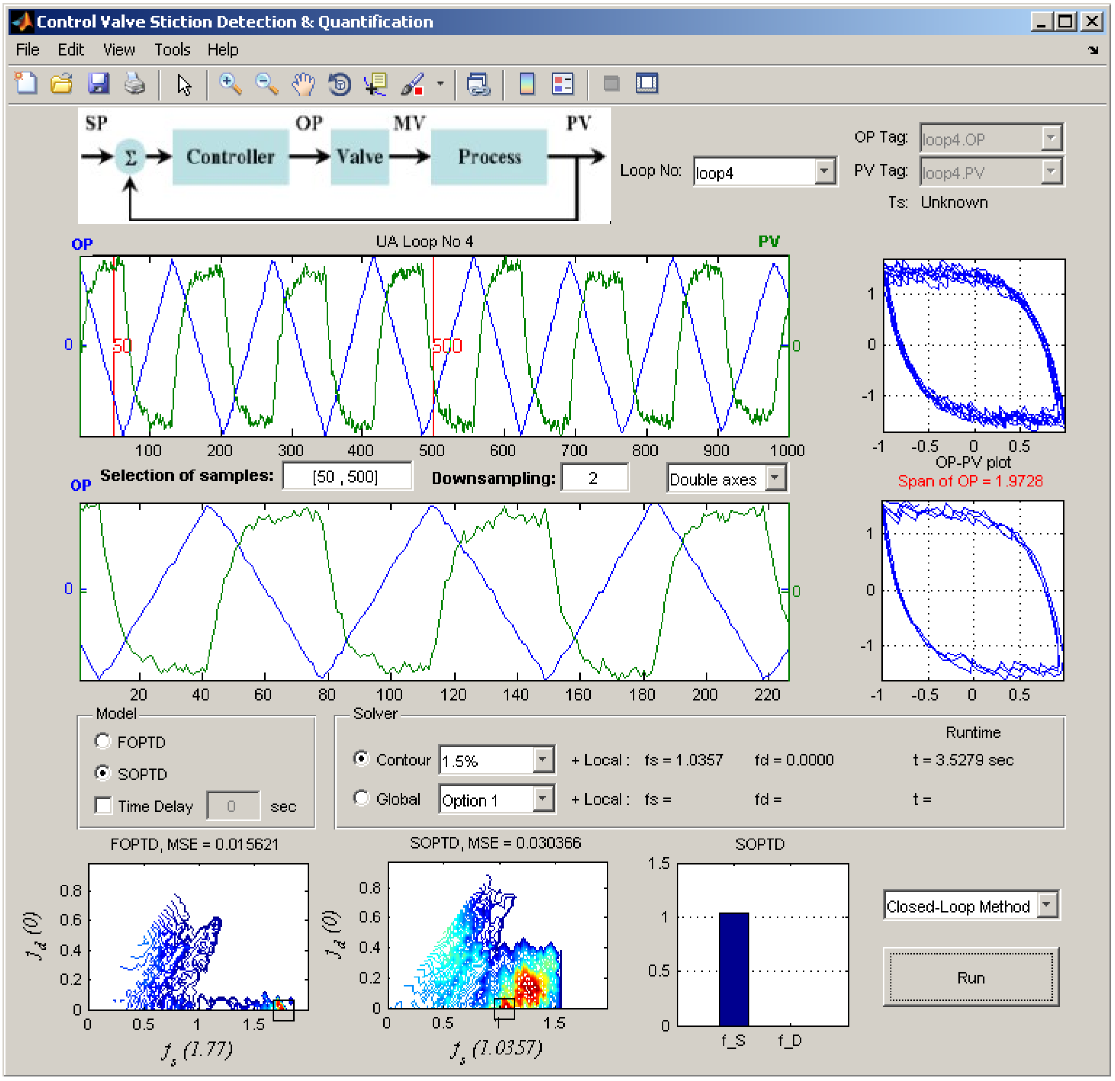
Closed-Loop Method
Run

Figure 12: Second-order model results

Part II: Some Other Issues Using the Toolbox: Time Sections and Downsampling

Assume that the same dataset is being used and a FOPTD model. Now, enter a time section as “[50, 500]”, which will allow the parameter estimates to be made for the 50th to the 500th samples. As well, a downsampling value of 2 will be entered. Remember to press enter after entering all of the values. The correct screen should look like Figure 13.

Figure 13 shows the results. On the whole, the parameter values are different, but they still allow the same conclusion: deadband with stick and slip behaviour.



Loop No: loop4

OP Tag: loop4.OP

PV Tag: loop4.PV

Ts: Unknown

Span of OP = 1.9728

Selection of samples: [50 , 500]

Downsampling: 2

Double axes

Model: FOPTD [✓ SOPTD] Time Delay 0 sec

Solver: [✓ Contour] 1.5% + Local : fs = 1.0357 fd = 0.0000 Runtime t = 3.5279 sec

Global Option 1 + Local : fs = fd = t =

FOPTD, MSE = 0.015621

SOPTD, MSE = 0.030366

SOPTD

Figure 13: Results of using a range of time sections

References

He, Q., Wang, J., Pottmann, M., & Qin, S. J. (2005). A curve fitting method for detecting valve siticiton in oscillating control loops. *Texas-Wisconsin Modeling and Control Consortium Spring Meeting.* Austin, Texas, United States of America.

Lee, K. H., Ren, Z., & Huang, B. (2008). Novel Closed-loop Stiction Detection and Quantification Method via System Identification. *Advanced Control of Industrial Processes (ADCONIP).* Jasper, Alberta, Canada.

Shoukat Choudhury, M. A., Thornhill, N. F., & Shah, S. L. (2005). Modelling valve stiction. *Control Engineering Practice* , 641-658.