IITB SUMMER INTERNSHIP 2017

Xcos-on-web

FOSSEE, IIT Bombay

May-July 2017

Under the guidance of Prof. Madhu Belur

Contributors

Jayaprakash Akula

Dattatreya Sarma G.V.S

Ritveeka Vashistha

Ruhi Taj

Shashank Bhalotia

Shivendra Pratap Singh

Mentors

Dipti Ghosalkar

Inderpreet Arora

Acknowledgement

By doing this project, we would like to thank all those who have been instrumental in helping us in achieving the desired outcome. We would like to thank **Prof. Madhu Belur** and IITB for providing us with this opportunity and having faith in our abilities. We would like to thank our mentors Ms.Dipti Ghosalkar and Ms.Inderpreet Arora for being a constant support and giving us a broader picture of the whole scenario. Their constant endeavor and guidance has helped us to complete the project within the stipulated deadline. We would like to thank Mr.Bikas Chhatri and Ms.Komal Solanki for providing us the administrative help and making our working environment comfortable. Last but not the least, we would like to thank our fellow interns for helping us out with all the problems that we faced and making our internship experience an enjoyable one.

Summer Internship 2017

Project Approval Certificate

FOSSEE

Indian Institute of Technology, Bombay

The project entitled "Xcos-On-Web" submitted by Dattatreya Sarma, Jayaprakash Akula, Ritveeka Vashistha, Ruhi Taj, Shashank Bhalotia and Shivendra Pratap Singh is approved for Summer Internship 2017, at Department of Computer Science and Engineering, IIT Bombay.

Prof. Madhu Belur
Co-Principal Investigator
Dept. Of EE, IITB

Miss. Dipti Ghosalkar
Project Mentor

Place: IIT Bombay
Date: July 4, 2017

Contents

1 Introduction **1**
- 1.1 Aim . . . 1
- 1.2 Motivation . . . 1

2 Using Xcos on web **2**
- 2.1 Building Scilab from Source on Linux . . . 2
- 2.2 Installing Other Requirements . . . 3

3 Limitations with earlier version **4**

4 Two way communication for TKSCALE block **5**
- 4.1 Introduction . . . 5
- 4.2 Methodology . . . 5
  - 4.2.1 Creating the sliders for TKSCALE block . . . 5
  - 4.2.2 Logging the values of TKSCALE to file . . . 7
  - 4.2.3 Generating graph from values of TKSCALE . . . 9
- 4.3 Results . . . 10
  - 4.3.1 Experimental setup having a single TKSCALE block . . . 11
  - 4.3.2 Experimental setup having two TKSCALE blocks . . . 11

5 Killing of scilab processes **12**
- 5.1 Introduction . . . 12
- 5.2 Methodology . . . 12
- 5.3 Results . . . 12

References **12**

Chapter 1

Introduction

Numerical simulation is nowadays essential in system design process. Simulation of complex phenomena (physical, mechanical, electronics, etc.) allows the study of their behavior and results without having to conduct costly real experiments. Scilab is an open source software dedicated for this purpose. It is widely used in the world of industry and the future generation of engineers and scientists are trained since secondary school to the concepts of modeling and simulation.

Xcos is a Scilab tool dedicated to the modeling and simulation of hybrid dynamic systems including both continuous and discrete models. It also allows simulating systems governed by explicit equations (causal simulation) and implicit equations (a causal simulation). Xcos includes a graphical editor which allows to easily represent models as block diagrams by connecting the blocks to each other. Each block represents a predefined basic function or a user-defined one. This blocks can be selected from a window called palette browser [1].

1.1 Aim

Xcos on web is a project to port the core functionalities of Xcos to a browser-only version that can be used without installing additional plug-ins or software on the cloud.

1.2 Motivation

Scilab is free and open source software for numerical computation providing a powerful computing environment for engineering and scientific applications. The current Scilab is a desktop version, which requires several installations. The current project is to build a web version, which carries all the functionalities of the original version and requires no installations. This would allow the user to use the scilab with more ease.

Chapter 2

Using Xcos on web

2.1 Building Scilab from Source on Linux

- Download Scilab source folder.
  - Goto: https://github.com/shivendra3/scilab_for_xcos.git
  - Clone the repository.
- If permission issue occurs, use chmod to grant permission. For Eg. If name of the extracted folder is scilab-master_5.5.2, use :
  - `$ chmod ugo+wrx scilab-master_5.5.2/ -R`
- Go to the scilab folder through terminal and update using :
  - `$ sudo apt-get update`
- Installing dependencies :
  - Scilab can be compiled with
    - For the C language: GCC (tested from gcc-4.0 => 4.5)
    - For the Fortran: Gfortran (GCC suite)
    - Java version (>=1.8)
  - It is required to install all the build dependencies under Ubuntu and Debian. Starting with Ubuntu (>= 9.4) and Debian (>= Squeeze), the following command will install all the build dependencies of Scilab.
    - `$ sudo apt-get build-dep scilab`
- Run configure script :
  - This script checks for many dependencies on your system and creates a “Makefile”
  - Type the following in the terminal : $ ./configure –disable-static-system-lib
- Make using : `$ sudo make -j4`
- Now run scilab using : `$ ./bin/scilab`

Note :
If at any step, “access denied” or some other related error occurs, again grant permission to the scilab folder as before and then redo the step.

2.2 Installing Other Requirements

- First install python and other necessary software.
- Open terminal and type :
  `$ sudo apt-get install libevent-dev python-dev python-setuptools`
- Then type : `$ pip install gevent flask`
- Once installation is completed, download Xcos-on-Web project from github : https://github.com/shankyb9/xcos_on_web.git
- Extract Xcos-on-Web, navigate through terminal inside that folder
- And type command : `$ python SendLog.py`
- Then open browser and type 127.0.0.1:8001
- This will open xcos on browser.

Make sure you change the path of scilab call ie. SCI variable in SendLog.py to your build scilab path before running any simulation.

Chapter 3

Limitations with earlier version

- Two way communication for TK Scale block was not present
- Graphs were not generated dynamically. Only images of chart were being displayed
- Parameters of blocks could not be modified at runtime
- Some blocks could not be dragged to the working area

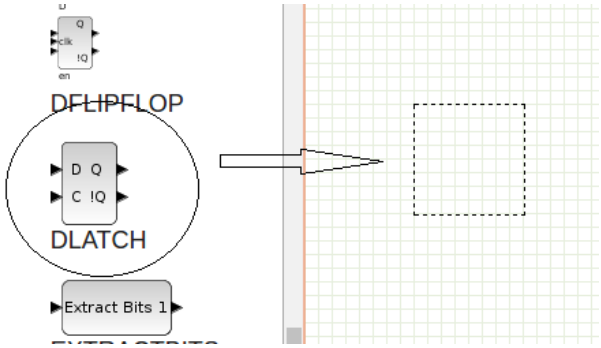


Figure 3.1: Block could not be dragged.

- The images for blocks were disappearing on setting any parameter
- 3-D graphs for some 3-D blocks was not possible
- No option to download the graphs and the .xcos file of the simulated diagram
- No method for identification of blocks

Chapter 4

Two way communication for TKSCALE block

4.1 Introduction

TK Scale is a 'Slider Widget' used to choose a numeric value through sliders. We can say simply say that TK Scale is an input slider Block. User can change the input during simulation/runtime. TK Scale is a 'Slider Widget' used to choose a numeric value through sliders. We can say simply say that TK Scale is a input slider Block. User can change the input during simulation/runtime.

Figure 4.1: TKSCALE block diagram

TK Scale has 3 parameters: minimum, maximum and normalization. Minimum and maximum represents the input range and the output of the block is determined as the slider's present value divided by the normalization factor. User can change the input range using these parameters.

Slider is used to change the input value by the user. User can change the input range and output of the slider by changing the parameters of TK Scale and all the obtained data should update in periodically with the specified 'PERIOD' parameter of CLOCK_c.

Out of the blocks present in Xcos application, TKSCALE is the only block which has a real time interaction with the user for input. The earlier version of Xcos on web has no module which enables the server to interact with user.

4.2 Methodology

4.2.1 Creating the sliders for TKSCALE block

Slider is used to change the input value by the user. User can change the input range and output of the slider by changing the parameters of TK Scale and all the obtained data should update in periodically with the specified 'PERIOD' parameter of CLOCK_c. So to create the slider and update the data the below steps are followed.

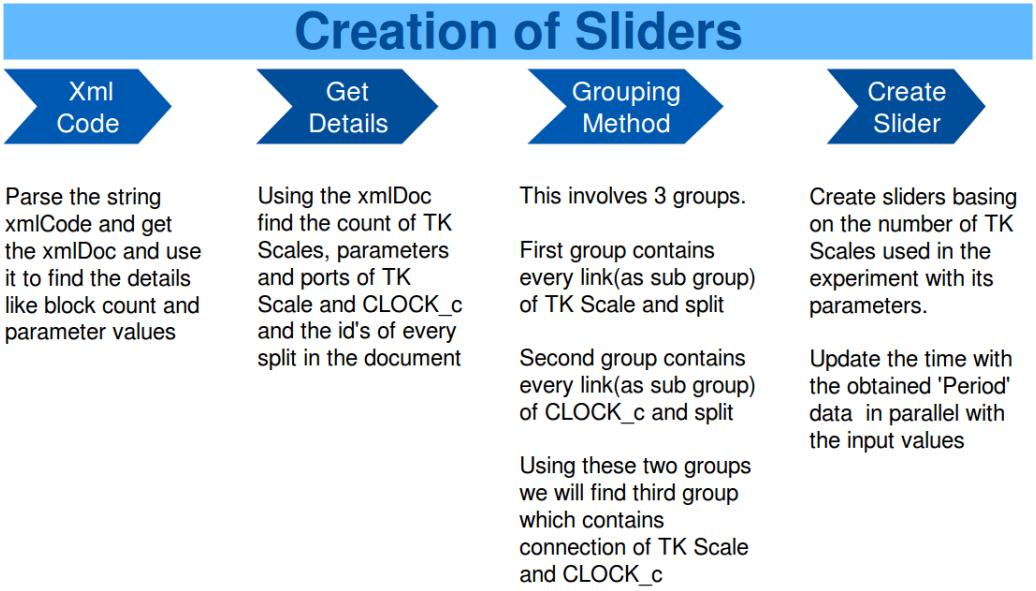


Figure 4.2: Flow Chart depicting the creation of sliders

Get details of TK Scale, CLOCK_c, Splits: Now, a loop is run on xmlcode to get the TK Scale (Basic Block) Details. We have obtained the count of TK Scales used in experiment with their parameters and stored them in an array. Also, we have to update the data (input) periodically, so we should also find 'which TK Scale is connected to which CLOCK' and get the clock parameters. To find the TK Scale and Clock relation the following steps were used:

- Find all the clocks with its parameters (clock details)
- Find the ports of TK Scale and CLOCK_c to find their links. The details of the ports are under mxcell tags in xmlcode.Also find all split id's.
  We used 2-D array, 1st column represents split id value, 2nd column indicates to which group split belongs to.
  -1 indicates split does not grouped with any other group.
- Grouping Method:
  Technique which iterates all the links in the experiment and grouping all the TK Scale and CLOCK_c blocks to find their relation so that we can update the input data periodically with the specified 'PERIOD' parameter.
  Iterating all the links:
  - Find the source and target of the present link
  - Find whether source or targets belong to any of the splits
  - If both source and target belong to any of the splits:
    * If both are not grouped to any other split then make grouping to both
    * If source is not grouped and target is grouped to another split then add source to the group where target belongs to

    - If target is not grouped and source is grouped to another split then add target to the group where source belongs to
    - If source, target both are grouped with some other splits then merge one group into another with minimum index number
  - If source does not belong to splits then it belongs to one of the CLOCK_c id's then find the source and add it a new group
  - If target does not belong to splits then it belongs to one of the TK Scale id's then find the target and add it to a new group

We got two connections by the above steps :

- which TK Scale connected to which split
- which CLOCK_c connected to which split

Using these two connections we will find the RELATION between TK Scale and CLOCK_c From the obtained connection between CLOCK_c and splits find the connection between CLOCK_C and TK Scale. Finally we found the required connection and added the 'Period' parameter to an array.

Create Sliders with Obtained Data: Now, we have created the sliders with obtained TK Scale details using slider.html and the url is passed with the required parameters. We have allowed the maximum number of Sliders (TK Scales) in the experiment to be 10. This condition is checked after the xcos file is uploaded.

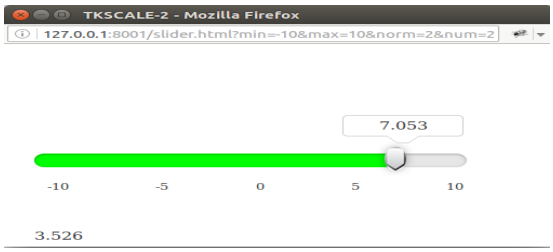


Figure 4.3: Example of slider generated

4.2.2 Logging the values of TKSCALE to file

Whenever user slides the pointer in any one of the slider we update values of each slider and store it based on the slider number. The ten id's tk1 to tk10 obtain the updated value from the slider. Whenever any one of the id value (innerhtml) is changed, 'getcurval' function is called. Together we store them in an array and an 'Ajax' request is made with that data.

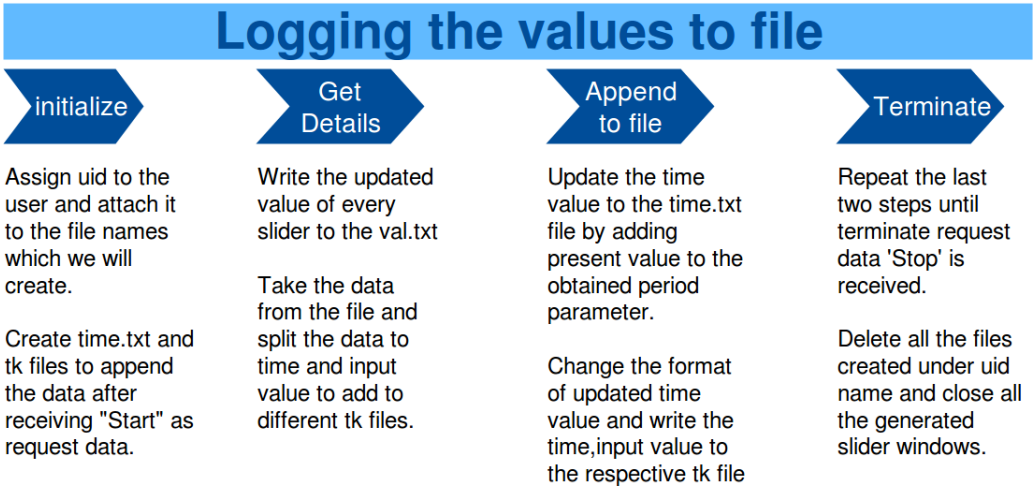


Figure 4.4: Flow Chart depicting logging of values to file

Note that when user changes the slider value, updated data will be written to the file else previous value will be written to the file continuously until new value appears. As we are writing the updated values (input values) to a file. These may overlap when there are multiple users. To overcome this situation we created files with uid (user session id) generated to every user. Using session ID we will not get any duplicate uid's. If there are more than 2^{14} requests in less than ten nanoseconds then, a few cases might appear to get duplicates which is impossible in our case due to less time span (10 ns).

Initialize the Process:
Updating the data to the file:

- Get the uid
- Check the initialization of writing data

`Val.txt` file contains the slider's present value and whenever any data updates, the file contents will be overwritten and use the same file to append the data repeatedly to get the graph. This function takes the updated data and store it in the `val.txt` file in values folder. Findfile function checks the `val.txt` file content. If it contains "Start" it initializes thread and creates required files, if it contains "Stop" it stops the thread, else data updation will be done. By the return value initialization or updation or stopping is done.

Thread Initialization: We will update the tk files every 0.1 second thereby getting the correct plot for the graph using g̈etdetailsthread()". Get details functions takes the `val.txt` file and makes the data to be appended at the file end.

Get details from Val.txt: We have to update the data periodically with the time. `Val.txt` contains only 'PERIOD' parameter so in order to update the time as well as data (input value) we pass the data in val.txt to another function "appendtotkfile" which appends data and time to the file.

We created a file `time.txt` which updates time with the 'PERIOD' parameter initialized to 0's in findfile function. For every iteration we add the time in `time.txt` file with the 'PERIOD' parameter and append it to the file with the data in the `val.txt`. Change Format will change the time floating value to scientific format and the input value is received in scientific format.
This process continues until user stops the simulation by closing the simulation window.

Terminate: For closing sliders and to stop the thread we use the event closing of simulation window. Stoptk function will close all slider windows as well as the thread by sending "Stop" data in ajax request in `index.html`. By receiving the request in `SendLog.py` file we will delete all the related files created. We will also call the stoptk function even if user closes the main window without closing the simulation window.

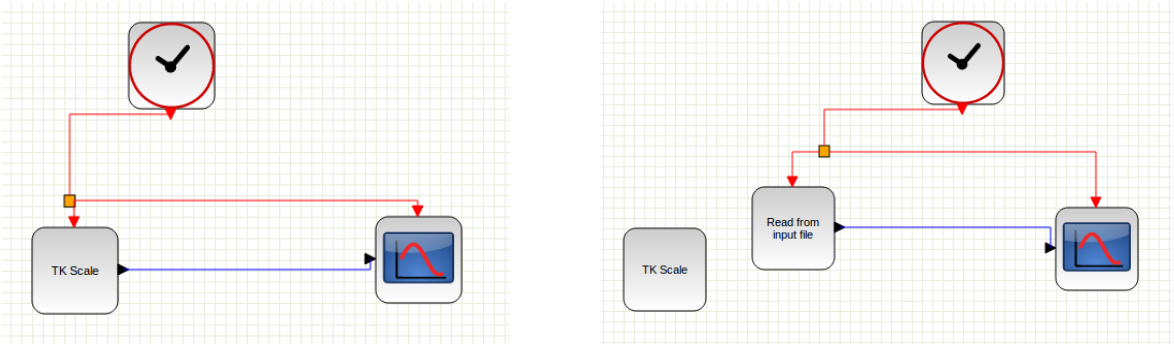
4.2.3 Generating graph from values of TKSCALE

The TKSCALE block, now has to read the values from the log file and process it to generate appropriate graphs. The exact method by which TKSCALE block transmits its value to the corresponding sink block is unknown. Only the method by which sink block obtains the resulting value is known. So, the first approach for this problem was to modify the latter method. But, this method too has its own limitations. For this approach to work correctly the source code of around 25-30 blocks had to be modified and it is not practically feasible and not an efficient method.

So, a different approach was adopted. This approach would replace TKSCALE block by a RFILE_f block. RFILE_f block allows user to read data in a file with the name defined with the input file name parameter, in text formatted mode or in binary mode. The file is a sequence of records. RFILE_f block updates its value from the file periodically based on the time period set for the CLOCK block it is connected to. This block might encounter an end of file argument during the reading process. In such cases the block repeats the last value it read and hence obtained output might not tally with the expected output.

The problem now is how to change the diagram at the backend without the user actually realizing it. The actual procedure of changing the diagram with TKSCALE to RFILE_f block is done only when the user runs the simulation. At that time the current xcos diagram (basically an xml file) is passed as an argument (.xcos file) to upload function. The entire manipulation of block is done here at the backend without reflecting these changes to the front end user.

The question that arises now is Ḧow do we create a new block in the xml file?" Code has been written to add the entire BASIC BLOCK xml part of RFILE_f in the .xcos file that



(a) Before Altering (b) After altering

Figure 4.5: A sample example of the block diagram and its corresponding diagram at the backend after altering it.

gets uploaded to server. For every TKSCALE block a new RFILE_f basic block xml has been added. The actual basic block xml for RFILE_f block can be obtained by exporting the xml for diagram containing only that block. As the xml part of the block is very big. It has been copied to a file and later read from it and written to the .xcos upload file.

The real task is to replace all the connections that a TKSCALE block had with the new RFILE_f block. A module has been written in python to modify the xcos that retains the connections of the TKSCALE and make RFILE_f to read the file containing the slider values. The connections is set in the desired way by just changing ID of RFILE_f with that of a TKSCALE block. Later the ID of TKSCALE block is altered with a negative number and henceforth virtually removing it. The parameters of RFILE_f block are changed to the read the required file by providing it with its location on the disk. Also the exact format in which the numbers are present in the file have been provided to the RFILE_f block [2]. Furthermore, the real time scaling value for experiments containing TKSCALE has been hard coded to 1.0 to give user a live experience.

Usually for the other blocks scilab process communication (continues till the end of simulation time) and graph generation go hand in hand. But for diagrams with TKSCALE the same procedure is not applicable as writing values and reading values need to happen parallel. For this purpose simulation diagrams are checked for non-empty canvas and then scilab communication is terminated. As scilab communication would restrict any parallel reading of file necessary for generation of graph. The code is made to sleep for a small duration of time during the log file reading procedure as some time is required for updating the log file with new value of TKSCALE.

4.3 Results

The following images will reflect the changes made and show the results of a few experimental setups containing TKSCALE block.

4.3.1 Experimental setup having a single TKSCALE block

The above experimental results are in the case where a single TKSCALE block is connected to CSCOPE block. The windows next to the setup are the slider window and simulation graph window. The simulation graph clearly reveals the value of slider at the instant of time.

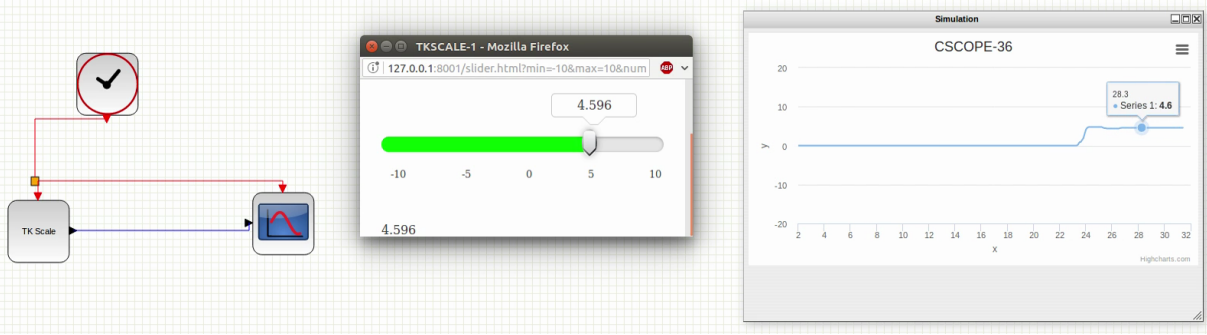


Figure 4.6: Experimental setup with its corresponding output

4.3.2 Experimental setup having two TKSCALE blocks

The above experimental results are in the case where two TKSCALE blocks are connected to CMSCOPE block with 2 input ports. The windows next to the setup are the slider window and simulation graph window. The simulation graph clearly reveals the value of slider at the instant of time. The first TKSCALE block has been set a normalization parameter as 2 and for the second one as 3.

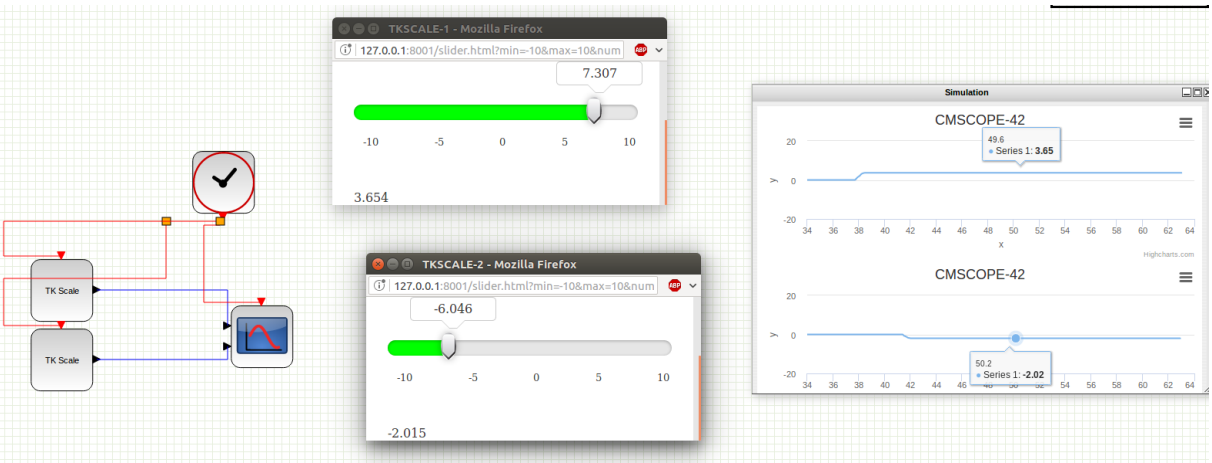


Figure 4.7: Experimental setup with its corresponding output

Chapter 5

Killing of scilab processes

5.1 Introduction

Xcos on web application is loaded on a server. So, it might have some limitations in terms of memory. So, one needs to kill the scilab instances and also delete the unnecessary log files after the end of simulation.

5.2 Methodology

Various files are created during a simulation. They include log files used in graph generation, workspace files, block identification files, files created in case of experiments containing TKSCALE block and .xcos file being uploaded to the server. These files may vary in size from few KBs to several MBs. They need to be deleted as they are irrelevant at the end of simulation and take up a lot of space.

At the end of simulation ie , when user closes the simulation window or when the browser is closed by the user without stopping simulation the a function call is made to kill the scilab processes. In this function the scilab process is killed with the help of its process ID.

However, the log files are not deleted completely as sometimes the user may perform more than simulation in a single session. So, only the log file is emptied and also the points from chart array (used for generating chart) are removed. The log file and other files are deleted at the end of the current session. Also the count of line (based on which log file is being read) is reset to zero in order to maintain proper file reading in case of multiple simulations in a single session.

5.3 Results

Several testings have been performed to test the same. Testings have also been done with multiple users. All the testings were successful and the deletion of files occurred properly.

References

[1] "Using xcos." https://www.scilab.org/content/download/390/2810/file/scilabtec_Xcos.pdf. 1

[2] "Fortran number format." https://www.cs.mtu.edu/~shene/COURSES/cs201/NOTES/chap05/format.htm. 10