



Summer Fellowship Report

on

SCILAB OPTIMISATION TOOLBOX

Submitted by

Naveen Krishna Yellanti

Sastra Deemed University, Thanjavur

Under the guidance of

Prof.Kannan M. Moudgalya
Department of Chemical Engineering,
IIT Bombay

Prof.Ashutosh Mahajan
Department of Computer Science Engineering,
IIT Bombay

Mentor

Mr.Rupak Rokade

May 20,2020

Acknowledgement

After visualising the seriousness of covid-19. I have almost decided that I am not going to do any internship this Summer. I almost lost hope on the fellowship this year. But shattering all my misbeliefs ” FOSSEE Team,IIT Bombay “ stood up strong during this tough period by conducting Remote Summer Fellowship. My Heartfelt thanks to “FOSSEE TEAM” for conducting Remote fellowship and giving me precious opportunity to learn many things and also giving me a chance to contact many wonderful people with immense knowledge and passion towards their work.

I am deeply grateful to Prof.Kannan M. Moudgalya, head of FOSSEE team, IIT Bombay, for giving this wonderful opportunity to enhance my skills at the same time spending time with family.

I received generous support from Prof.Ashutosh Mahajan, professor in the Department of Computer Science, IIT Bombay. I Thank him for his great support and suggestions.I am in debt for him for his necessary advices and guidance in both professional and personal aspects. I choose this moment to acknowledge his selfless and cool attitude.

I am particularly grateful for the assistance given by Mr.Rupak Rokade. I am thankful for his commitment towards his work and at same time guiding me in every aspect of the work without losing patience. I personally feel blessed for having him as my mentor for this fellowship.

I am glad for getting this auspicious opportunity that helps in career development. I will remember the lessons learnt during fellowship throughout my life and will try my best to enhance skills and to become an expert in these fields. Hope these skills will be helpful in my future.

Contents

1	Introduction	1
1.1	About FOSSEE Optimisation Toolbox	1
1.2	COIN-OR	2
2	Structure of Optimisation Toolbox:	3
2.1	Files	4
2.1.1	builder.sce	4
2.1.2	loader.sce	4
2.1.3	demos	4
2.1.4	Macros	4
2.1.5	etc	5
2.1.5.1	FOSSEE_Optimization_Toolbox.start	5
2.1.5.2	FOSSEE_Optimization_Toolbox.quit	5
2.1.6	jar	6
2.1.7	cleaner.sce	6
2.1.8	unloader.sce	6
2.1.9	Help	6
2.2	Guidelines to use the toolbox	6
3	Contributions	9
3.1	Task of the fellowship	9
3.1.1	Problem Statement	9

3.1.2	Solutions	10
3.1.2.1	For CNES report:	10
3.1.2.2	For 0.4 version	13
3.2	Problems Faced	14
3.3	Problem that is yet to be solved	15
3.4	Summary of my contributions	16
4	Useful Links	17
5	References	18

1. Introduction

Scilab is a open-source numerical computational package licensed under GPLv2 license which has broad applications in educational and engineering domains. It is a competitive alternative to Matlab and Octave. Scilab was initially maintained and developed by INRIA³(French Institute for Research in Computer Science and Automation). Scilab consortium was formed in June 2010 which now handles Scilab development. FOSSEE Optimization toolbox uses a dozen of open-source optimization solvers to solve the optimization problems. Many of these solvers are part of the COIN-OR initiative which promotes the development and use of open-source softwares for operations research community. Scilab provides API functions to call libraries from C, C++ and FORTRAN. Most of the solvers used by the toolbox is programmed in C++.

1.1 About FOSSEE Optimisation Toolbox

FOSSEE Optimization toolbox is a toolbox in Scilab maintained and developed by FOSSEE(Free and Open Source Software in Education), IIT Bombay.

It can solve the following optimization problems :

1. Linear programming(LP)
2. Quadratic programming(QP)
3. Nonlinear programming(NLP)
4. Integer programming(IP)
5. Second order Conic Programming(SOCP)

It also solves specific optimization problems like least squares, minimax and goal attainment problem.

FOSSEE Optimization toolbox mainly uses six mathematical optimization libraries, namely :

1. CLP7 (Coin-or Linear Programming)
2. Ipopt8 (Interior Point OPTimizer)
3. Symphony9
4. Bonmin10 (Basic Open-source Nonlinear Mixed INteger programming)
5. CBC11(Coin-or branch and cut)
6. ECOS12

These libraries, in turn are dependent other libraries such as:

1. CGL13 (Cut Generation Library)
2. LAPACK14(Linear Algebra PACKage)
3. BLAS15 (Basic Linear Algebra Subprograms)
4. MUMPS16 (MUltifrontal Massively Parallel Sparse direct Solver)
5. OSI17 (Open Solver Interface)

1.2 COIN-OR

Computational Infrastructure for Operations Research (COIN-OR), is a project that aims to "create for mathematical software what the open literature is for mathematical theory." The open literature (e.g., a research journal) provides the operations research (OR) community with a peer-review process and an archive. Mathematical libraries and Source codes used in FOT are extracted from COIN-OR

2. Structure of Optimisation Toolbox:

Followings are the list of visible files and folders in the main directory of the FOSSEE-Optimisation-toolbox of Scilab:

1. *builder.sce*
2. *loader.sce*
3. *demos*
4. *etc*
5. *jar*
6. *cleaner.sce*
7. *unloader.sce*
8. *help*

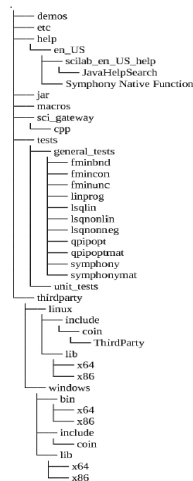


Figure 2.1: FOT Folder structure tree

2.1 Files

2.1.1 builder.sce

This file builds the macros, help and the loader.sce files. Type the command `exec builder.sce` in the scilab console to execute this file.

Both these have to be executed every time, to load the toolbox for usage.

2.1.2 loader.sce

It is basically a file that calls the function `FOSSEE_Optimization_Toolbox.start` present in the etc folder. This has to be executed first on opening the toolbox, using the command `exec loader.sce` in the scilab console.

2.1.3 demos

Contains demo files for the toolbox

2.1.4 Macros

Macros folder contains scilab function files(*.sci). Files with extensions other than sci will not be compiled when the builder is run. Scilab macros can be:

1. A Scilab function file which returns the result after computation.
2. A Scilab function which calls a C, C++ or FORTRAN code.
3. A Scilab function which calls a binary library.

The general outline of most of macros files in FOT is as follows:

1. Help page comments
2. Input retrieval
3. Error checks
4. Input modifications
5. Call to the C++ library
6. Output retrieval,checks and modifications

2.1.5 etc

etc directory contains the initialization and finalization script of the toolbox which are run at the beginning and termination of the toolbox. They are executed while executing the loader and unloader files.

2.1.5.1 FOSSEE_Optimization_Toolbox.start

The name of the initialization script for a toolbox is the name of the toolbox followed by ".start".

In our case, the name of the file is "FOSSEE_Optimization_Toolbox.start".

The FOSSEE_Optimization_Toolbox.start file is executed when we run the loader. It's purpose includes :

1. Load function libraries from macros directory.
2. Load gateway and shared libraries form sci_gateway and thirdparty directory.
3. Load help from help directory.
4. Load demos from demos directory.

2.1.5.2 FOSSEE_Optimization_Toolbox.quit

The name of the finalization script for a toolbox is the name of the toolbox followed by ".quit". In our case, the name of the file is

"FOSSEE_Optimization_Toolbox.quit". The FOSSEE_Optimization_Toolbox.quit file is executed when we run the loader.

It's purpose includes :

1. Unlink the toolbox libraries.
2. Remove any preferences that were set by the toolbox.

2.1.6 jar

This folder has a `scilab_en_US.jar` file. This file is basically a Java Archive package file format typically used to aggregate many Java class files and associated metadata and resources (text, images etc.) into a file for distribution.

2.1.7 cleaner.sce

This file is generated by `builder.sce`. On executing this file, we actually delete the `loader.sce` and the `unloader.sce` files. One caution to the users is that do not edit this file.

2.1.8 unloader.sce

It generally unloads the toolbox.

2.1.9 Help

The FOSSEE Optimization Toolbox has an extensive help section that covers all of the functions that the toolbox currently consists of.

2.2 Guidelines to use the toolbox

The toolbox will be available at <https://github.com/FOSSEE/FOSSEE-Optimization-toolbox.git>. The users are requested to go through the `README.md` file prior using the toolbox.

Else, the users are requested to follow the following steps:

1. Clone this repository as it is and put it in the same folder as the dependency folder.

2. Download coin-or libraries from http://www.ieor.iitb.ac.in/files/faculty/amahajan/tmp/fot_mingw.zip

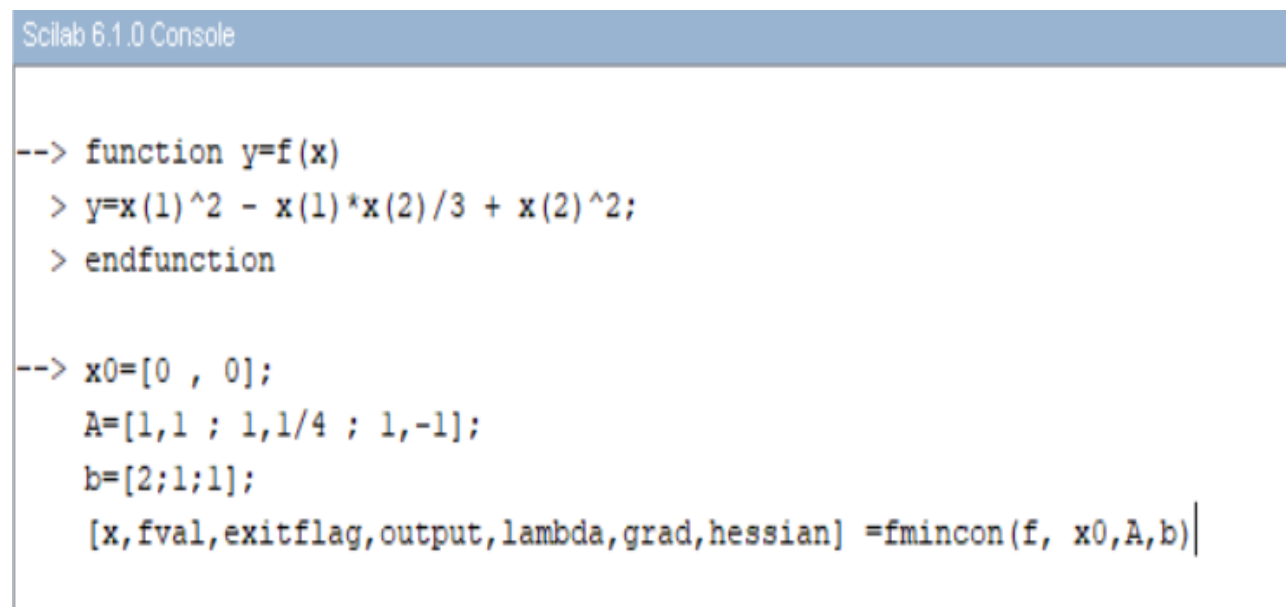
After extracting the .zip file copy windows folder into FOSSE-Optimization-toolbox/ThirdParty directory.

3. Go to the FOSSEE_Optimization_toolbox folder, execute the builder.sce using `exec builder.sce` in Scilab-console.

4. Inside `scigateway/cpp` directory edit `loader.sce`. The path and order of the files inside `link()` has to be similar to that of files in `ThirdParty/windows/bin`.

3. Then execute `loader.sce` using `exec loader.sce` and start using the functions in the toolbox. This step should be repeated every time you restart the Scilab to load the toolbox again.

Once the toolbox is built and loaded by following the steps mentioned above, we can verify the functioning of the toolbox by executing any file inside `tests/unit_tests`. The list of functions can be seen at `FOSSE-Optimization-toolbox/macros`. We can execute any of the desired functions in the toolbox. An example (for `fmincon`) is shown below:



```
Scilab 6.1.0 Console

--> function y=f(x)
  > y=x(1)^2 - x(1)*x(2)/3 + x(2)^2;
  > endfunction

--> x0=[0 , 0];
  A=[1,1 ; 1,1/4 ; 1,-1];
  b=[2;1;1];
  [x,fval,exitflag,output,lambda,grad,hessian] =fmincon(f, x0,A,b)
```

Figure 2.2: Testing the whether toolbox was loaded properly by using `fmincon`

It should give the result as shown:

```
Scilab 6.1.0 Console
--> [x,fval,exitflag,output,lambda,grad,hessian] =fmincon(f, x0,A,b)
Optimal Solution Found.
x =
    0.0000001    0.0000003
fval =
    7.623D-14
exitflag =
    0.
output =
Iterations = 15
Cpu_Time = 2.109
Objective_Evaluation = 16
Dual_Infeasibility = 0.0000005
Message = "Optimal Solution Found"
lambda =
lower = [0,0]
upper = [0,0]
ineqlin = [4.545D-08,9.091D-08,9.091D-08]
eqlin = []
ineqnonlin = []
eqnonlin = []
grad =
    0.0000002    0.0000005
hessian =
    2.          -0.3333333
   -0.3333333  -0.3333333
```

Figure 2.3: Result

3. Contributions

3.1 Task of the fellowship

3.1.1 Problem Statement

1. CNES, French Space Agency is the biggest user of FOSSEE Optimisation toolbox. Report given by CNES for some improvements in some functions is given as a problem statement by my mentor to me for this fellowship.

There are totally 6 divisions in the report:

I have been assigned with division 3,4 and 5. Every division will have some sub-divisions.

These tasks were:

Division 3: Code (fmincon.sci)

Division 4: Calling sequence

Division 5: Toolbox test

2. Some important changes are required for updating FOT to version 0.4.

Some of the changes were given as problem statement. They are:

1. Symphony include files, functions, macros etc need to be removed.
2. Check for other unimplemented/obsolete functions or macros.
3. `fotversion()` and `fot_version()` functions don't work. It should report:
 - a. git # of the commit which was used to build the toolkit
 - b. version numbers of each library used (lapack, mumps, ipopt, cbc, clp, bonmin)
4. All functions that we provide to Scilab must start with `fot_` .
e.g. `fot_intfmincon()` instead of `intfmincon`.

3.1.2 Solutions

3.1.2.1 For CNES report:

Division 3.1: CALLS TO EXECSTR:

This is a simple task.

In the code of `fmincon`(inside `macros` directory), there are calls to `execstr` like this:

```
if(execstr('y=lHess(x,obj,lambda)','errcatch')==32 | execstr('y=lHess(x,obj,lambda)',  
, 'errcatch')==27)
```

We can observe that the same function with the same arguments is called twice. It is not a simple function. It requires a high amount of time to execute. Simple solution for this problem is to use an extra variable that calls the function once and stores the result. Later we can use that variable any number of times. After changes it looks like:

```
execstr1 = execstr('y=lHess(x,obj,lambda)','errcatch');  
if (err == 32 | err == 27)
```

Similar changes are done at 5 different places in complete code.

Division 3.3: INTERNAL FUNCTION (NAMING CONVENTION):

Some internal functions are defined, such as `Checktype`.

The name may conflict with functions defined by the user who does not even know that this function is defined.

So a better name would be something like: `fot_checktype`.

Totally five functions were renamed.

They are:

- 1.fot _ Checktype
- 2.fot _ Checklhs
- 3.fot _ Checkrhs
- 4.fot _ Checkvector
- 5.fot _ Checkdims

Division 4.1. "OPTION" ARGUMENT LIMITED USAGE:

Previously calling sequences of `fmincon` were able to call options only after all remaining 9 arguments were given. Task is to change code as such options can be called with any number of arguments but should be at last.

Calling Sequence

```
xopt = fmincon(f,x0,A,b)
xopt = fmincon(f,x0,A,b,Aeq,beq)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub,nlc)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub,nlc,options)
[xopt,fopt] = fmincon(.....)
[xopt,fopt,exitflag]= fmincon(.....)
[xopt,fopt,exitflag,output]= fmincon(.....)
[xopt,fopt,exitflag,output,lambda]=fmincon(.....)
[xopt,fopt,exitflag,output,lambda,gradient]=fmincon(.....)
[xopt,fopt,exitflag,output,lambda,gradient,hessian]=fmincon(.....)
```

Figure 3.1: Initial Calling sequence

Calling Sequence

```
xopt = fmincon(f,x0,A,b)
xopt = fmincon(f,x0,A,b,options)
xopt = fmincon(f,x0,A,b,Aeq,beq)
xopt = fmincon(f,x0,A,b,Aeq,beq,options)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub,options)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub,nlc)
xopt = fmincon(f,x0,A,b,Aeq,beq,lb,ub,nlc,options)
[xopt,fopt] = fmincon(.....)
[xopt,fopt,exitflag]= fmincon(.....)
[xopt,fopt,exitflag,output]= fmincon(.....)
[xopt,fopt,exitflag,output,lambda]=fmincon(.....)
[xopt,fopt,exitflag,output,lambda,gradient]=fmincon(.....)
[xopt,fopt,exitflag,output,lambda,gradient,hessian]=fmincon(.....)
```

Figure 3.2: Calling sequence after updating

To change the calling sequence changes has to be while checking the number and type of user inputs. Necessary if and else conditions have to be used while storing the user input parameters.

Division 4.2. "OPTION" ARGUMENT TYPE:

Task is to change the type of options from list to struct for convenience of usage. It was done by using `fieldnames()` function. `fieldnames()` function with argument of struct datatype gives the array of keys in that struct. This function is used while checking the inputs given by the user and simply we can use `type(param.Maxiter)` function (where `param` is the struct variable and `Maxiter` is one of the keys inside `param`) for checking the user input datatype and values.

Division 4.3. DIMENSION OF OUTPUT

The size of `xopt` seems to always be `Nx1`.

It would probably be better to set the size of `xopt` so that it is identical to the size of `x0`, which is probably what the user expects.

It is also a simple task. I have used a flag variable to get the initial size. And finally used the same flag to give required dimensions as output.

Division 5:

The script “`fmincon.tst`” fails:

After making necessary changes in `assertclose` line() it is working fine.

3.1.2.2 For 0.4 version

1. Symphony related things were removed from FOT. "git rm" command is used for this operation.

Things that are removed from toolbox:

Macros: Symphonymat, Symphony, Symphony_call.

Scigateway/cpp: Sci_sym_get_iteration_count.cpp, sci_sym_loadproblem.cpp, sci_sym_loadproblem_adv.cpp, sci_sym_openclose.cpp, sci_sym_set_variables.cpp, sci_sym_solution.cpp, sci_sym_solve.cpp, sci_sym_vartype.cpp.

tests/unit_tests: symphonybase.tst, symphonybase.dia.ref, symphonymat_base.tst, symphonymat_base.dia.ref.

tests/general_test: symphony and symphonymat.

2. ecos is the unplemented function in FOT. So I have removed it from macros and also other files related to ecos in scigateway,tests and demos directories.

3. Earlier fotversion and fot_version functions were not working. I have debugged the problem. Problem is inside sci_fotversion.cpp file inside scigateway/cpp directory. It is checking for the wrong number of inputs. Number of inputs has to be zero but it is checking for 1. After changing it to zero these functions were working fine.

4. There are 18 functions that FOT provides to scilab. They are:

fgoalattain	intlinprog
fminbnd	intquadprog
fmincon	linprog
fminimax	lsqlin
fminunc	lsqnonlin
intfminbnd	lsqnonneg
intfmincon	quadprog
intfminimax	quadprogCLP
intfminunc	quaprogramat

All these function names are renamed with `fot_` in the front.

Changing the function effects all the contents of other directories so they have to be changed:

I have changed all the names and contents of `demos`, `help`, `macros` and `tests`.

Earlier `.sce`, `.dem`, `tst` and `.sce` files used to call actual function names I have changed everything with `fot_` in front of them.

3.2 Problems Faced

Main problem is that utmost care has to be taken while changing any line of the code. Since `fmincon` is very long code (nearly 1000 lines). If any mistake is done it will affect the whole result and even debugging of the problem will be a hectic task.

While changing the argument type of options. It took many hours to overcome the task of the Error Checking part. Initially it was of list type and error check sequence is easier but with struct type it is difficult to get the dimension of input and to know what all the parameters that user has assigned. After knowing about `fieldnames()` function in `scilab` my task became simpler. After that retrieving the key values became simpler.

For changing the function call (naming Convention) names for five functions. Since there are 31 functions totally and every function calls other functions internally. The five functions (`Checktype`, `Checkdims`, `Checkvector`, `Checklhs`, `Checkrhs`) are called by most of the 31 functions. So after changing the function I used to go to every function and check for those function calls and changed them.

I have faced a problem with pushing the repo to Github page. Binary files should not be pushed into repo. So whenever I make any change in a cloned repo I have made it carefully. Since if we push once, reverting the changes is a hectic job. And the safe way that I discovered is to make the copy of the cloned directory and make experiments with the copy. And whenever we get the favourable results

making changes in the main repo and pushing it.

3.3 Problem that is yet to be solved

Division 3.2: NUMDERIVATIVE

It is one of the changes that CNES mentioned in the report. Whenever I tried to make changes as mentioned in the report the scilab console is crashing for one of the example problems. Even there is no much gain in the computation time.

For 0.4 version

There are number of test functions but they are failing while using test_run funtion. It is because test_run works by comparing auto generated .dia file with .dia.ref. If a single dis-similarity is encountered the test fails. While comparing results it compares CPU_time too. Where it time taken by CPU varies everytime we run the function test_run is failing.

3.4 Summary of my contributions

Since the main Aim of the report given by CNES is to reduce the computation time. And making functions to run smoother and softer.

The following table gives the enhancement in the computation speed of function fmincon with some unique examples:

Example num:	6.1.0 initially		6.1.0 After Limited changes	
	TT	CPU t	TT	CPU t
1	2.252	2.079	1.344	1.238
2	0.767	0.603	0.473	0.352
3	0.77	0.604	0.468	0.361
4	1.355	1.188	0.83	0.715
5	0.762	0.629	0.31	0.21
6	1.936	1.763	1.182	1.069
7	1.526	1.348	0.903	0.795

TT= time measured using tic toc around fmincon function call

CPU t = time given along with results

6.1.0 initially = Readings are from FOT without any changes as mentioned in CNES

6.1.0 After limited Changes= Readings are taken from FOT after making some changes as mentioned in CNES report

* Answers in all the three cases are similar.

4. Useful Links

FOSSEE-OPTIMISATION toolbox:

<https://github.com/FOSSEE/FOSSEE-Optimization-toolbox.git>

My work related to fellowship can be found in:

Github Repository:

<https://github.com/Naveen2307/FOSSEE-Optimization-toolbox.git>

Work status Spreadsheet:

<https://docs.google.com/spreadsheets/d/1GIFjkMS>

5. References

<https://scilab.in/fossee-scilab-toolbox/optimization-toolbox/functions/fmincon>

<https://www.scilab.org/build-toolbox>

<https://projects.coin-or.org/Clp/attachment/ticket/16/Ipopt.sln>

<http://www.mingw.org/wiki/MSYS>