Semester Long Internship Report

On

FOSSEE Optimization Toolbox

Submitted by

Pranshu Malhotra
Punjabi University, Patiala

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering
IIT Bombay

Prof.Ashutosh Mahajan
Computer Engineering
IIT Bombay

Rupak Rokade
Project Manager at FOSSEE
IIT Bombay

September 4, 2021

Acknowledgment

My heartfelt gratitude to FOSSEE TEAM for conducting Semester Long Internship remotely and giving me amazing opportunity to learn and contribute to the open source projects and also giving me a chance to work with 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 technical skills while contributing to an amazing free and open source software for education.

I received generous support from Prof.Ashutosh Mahajan, professor in the Department of Computer Science, IIT Bombay. I thank him for his great support. I want to thank him for his necessary advices and guidance.

I am particularly grateful for the support given by Mr.Rupak Rokade. I am thankful for his support and his management skills, conducting this remote internship smoothely.

I am glad for getting this opportunity that helps in skill development and also giving me an opportunity to contribute to the community by contributing to this wonderful free and open source educational toolbox. I will remember the lessons learnt during fellowship throughout my life and will try my best to enhance skills and to keep contributing in these open source projects.

Contents

Chapter 1

Introduction

1.1 FOSSEE Optimization Toolbox

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++

1.1.1 Functionality provided by the 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.

1.1.2 Mathematical Optimization Libraries

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

1.1.3 Other libraries and dependencies

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)

Chapter 2

Structure of Optimization toolbox

2.1 Files

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

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.

2.1.6 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.

Chapter 3

Contributions

3.1 Tasks assigned

The tasks assigned were as follows:

1. Exploring, implementing and testing the ”limited-memory” option of IPOPT for reducing the computation time (Issue raised in the CNES report)
2. Translation of CUTest test cases from AMPL to Scilab.
3. ”Integer Constraints Not Working” (Issue raised on GitHub)
4. Writing/editing code files and verification of technical details in Spoken Tutorials.

Besides these, other improvements have been done to the toolbox.

3.2 Task analysis and Solutions

3.2.1 Improving performance for non-linear optimization problems

IPOPT Library and Hessian Approximation

Ipopt (Interior Point Optimizer, pronounced “Eye-Pea-Opt”) is an open source software package for large-scale nonlinear optimization.

Ipopt implements an interior point line search method that aims to find a local solution of (NLP). Ipopt has an option to approximate the Hessian of the Lagrangian by a limited-memory quasiNewton method (L-BFGS). You can use this feature by setting the option hessian approximation to the value “limited-memory”. In this case, it is not necessary to implement the Hessian computation method Ipopt::TNLP::eval h. If you are using the C or Fortran interface, you still need to implement these functions, but they should return false or IERR=1, respectively, and don’t need to do anything else.

In general, when second derivatives can be computed with reasonable computational effort, it is usually a good idea to use them, since then Ipopt normally converges in fewer iterations and is more robust. An exception might be in cases, where your optimization problem has a dense Hessian, i.e., a large percentage of non-zero entries in the Hessian. In such a case, using the quasi-Newton approximation might be better, even if it increases the number of iterations, since with exact second derivatives the computation time per iteration might be significantly higher due to the very large number of non-zero elements in the linear systems that Ipopt solve in order to compute the search direction.

Since the Hessian of the Lagrangian is zero for all variables that appear only linearly in the objective and constraint functions, the Hessian approximation should only take place in the space of all nonlinear variables. By default, it is assumed that all variables are nonlinear, but you can tell Ipopt explicitly which variables are nonlinear, using the Ipopt::TNLP::get number of nonlinear variables and Ipopt::TNLP::get list of nonlinear variables methods, see Additional methods in TNLP. (Those methods have been implemented for the AMPL interface, so you would automatically only approximate 5 the Hessian in the space of the nonlinear variables, if you are using the quasi-Newton option for AMPL models.) Currently, those two methods are not available through the C or Fortran interface.

Solution

The following changes have been made in the codebase to achieve the use of "limited memory" option of Ipopt, and to test it successively:

•The option for choosing the Hessian Approximation has been added in the options struct in the file sci gateway/cpp/sci ipoptfmincon.cpp. The option is aptly named as HessianApproximation, and takes a scalar, either 0 or 1 as an input, as follows:

options = struct("MaxIter", [3000], "CpuTime", [600],"HessianApproximation", [1])

Giving the value 0 passes the value of hessian approximation as "exact" in Ipopt, whereas a value of 1 passes the value of hessian approximation as "limited-memory".
The line where this occurs (line 205) is as follows:

app->Options()- >SetStringValue("hessian approximation", ha);
where ha takes the value "exact" or "limited-memory" accordingly.

•The `fot_fmincon.sci` file has been updated to include the description for the newly added HessianApproximation option in the options section of the fot fmincon help page, and to show that it takes the default value as 0.

• The error message has been added in the fot fmincon.sci file, so that if the user enters any other value for HessianApproximation except 0 and 1, the following message shows up on the Scilab console:

```
fot fmincon:  Value for Hessian Approximation should be either 0 or 1
```

3.2.2 Translation of non-linear problems from APML to Scilab

CUTEst testing environment

CUTEst (Constrained and Unconstrained Testing Environment with safe threads) is the latest evolution of CUTE, the constrained and unconstrained testing environment for numerical optimization. It is a versatile testing environment for optimization and linear algebra solvers. The test problems provided are written in so-called Standard Input Format (SIF). A decoder to convert from this format into well-defined Fortran 77 and data files is available as a separate package. Once translated, these files may be manipulated to provide tools suitable for testing optimization packages. Ready-to-use interfaces to existing packages, such as MINOS, SNOPT, filterSQP, Knitro, and more, are available. CUTEst is available on a variety of UNIX platforms, including Linux and is designed to be accessible and easily manageable on heterogeneous networks.

Selection criteria for non-Linear programming problems

A selection of problems have been made from the CUTE set of PrincetonLib. PrincetonLib is a collection of nonlinear programming (NLP) models. The purpose of the collection is to provide algorithm developers of nonlinear optimization codes with a large and varied set of both theoretical and practical test models. It also aids in the software quality assurance process by providing a set of tools to facilitate benchmarking and performance analysis. The original models are in AMPL format and collected by Robert Vanderbei and colleagues at Princeton University. The problems have been translated from AMPL to Scilab and such problems have been selected which have a large number of variables and equations.

Testing

In order to test the problems conveniently, the value of variables have been decreased so as to keep it within the processing capacity of the testing systems. In order to verify the value of the optimal solution, and the optimal value of the objective function at these reduced values, NEOS Ipopt/AMPL Solver has been used.

The problems that have been selected are as follows:

1. broydn3d
2. cosine
3. curly10
4. dtoc1l
5. hadamard
6. hager1
7. hager4
8. liswet10
9. mancino

10. msqrtb
11. penalty1
12. power
13. reading2
14. sipow3
15. tfi2
16. ubh1

Execution of CUTEst testcases

The test cases can be found at the location

`FOSSEE-Optimization-Toolbox/ tests/general tests/CUTEst`

in the source repository. In order to execute the CUTEst test cases, perform the following steps:

1. Open Scilab and load the FOSSEE Optimization Toolbox by running exec loader.sce on the Scilab console. If the toolbox has not been built, run exec builder.sce before the previous command.

2. Navigate to the CUTEst tests folder by executing the command:
   `$ cd FOSSEE-Optimization-Toolbox/ tests/general tests/CUTEst`

3. Now you can execute the individual test cases by running the corresponding files. A problem can be executed by executing the command:
   `$ exec <problem name>.sce`
   where <problem name>can be replaced by any of the problem names mentioned above. For instance, to execute the problem broydn3d, execute the command – `$ exec broydn3d.sce`.

The expected test results obtained from NEOS Ipopt/AMPL Solver are written at the top in each test file , and one can check the results obtained on the console with them in order to verify whether they are correct.

3.2.3 Testing the fix for the function fot_intlinprog and other integer constraint functions

- Issue 43 on github :
  https://github.com/FOSSEE/FOSSEE-Optimization-toolbox/issues/43

- In macros/cbcmatrixintlinprog.sci, line 229 has been added, and line 262 has been altered to pass numintcons to the gateway file.

- In sci gateway/cpp/sci intlinprog matrixcpp.cpp line 99 has been added to initialize numintcons, and line 49 has been altered to check whether nin is equal to the new number of input arguments, i.e. 12

- All the integer constraint function:

  - fot_intlinprog()
  - fot_intfminbnd()
  - fot_intfmincon()
  - fot_intfminmax()
  - fot_intquadprog()
  - fot_intfminunc()

  All the above mention functions where tested for the similiar issue.

3.2.4 Technical contribution to Spoken Tutorials

- Technical inspection and contribution to the files of the chapter unconstrained optimization for technical errors
  - Verifying slides
  - Verifying scripts and fixing technical errors
  - Contribution to code files
  - Contribution to assignment solutions

3.3 Problems faced

- A critical bug was found(in the function **fot_fmincon()**)while working on implementation of limited-memory option to enable hessian approximation in fot_fmincon(), the implementation was not working, then I have to debug the **fmincon.sce(macro file)** and the **associated C++ file**. where after a lot of debugging, the bug was found in fmincon.sce file where left-hand-side arguments where not handled correctly.

- During the translation of CUTEst nonlinear problems with large number of variables, from AMPL to Scilab, several minutes was taken by testcases when testing. This was very time consuming and CPU intensive process.

Chapter 4

Useful links

- FOSSEE-OPTIMISATION toolbox:
  https://github.com/FOSSEE/FOSSEE-Optimization-toolbox.git
- My work related can be found in: Github Repository:
  https://github.com/osmium8/FOSSEE-Optimization-toolbox
- CUTEst non linear optimization problems AMPL files:
  https://vanderbei.princeton.edu/ampl/nlmodels/cute/
- AMPL solvers:
  https://ampl.com/products/solvers/
- Hessian Approximation in theory reference :
  http://www.princeton.edu/ yc5/ele522_optimization/lectures/quasi_Newton.pdf

Chapter 5

Reference

- https://scilab.in/fossee-scilab-toolbox/optimization-toolbox/functions/fmincon
- https://scilab.in/fossee-scilab-toolbox/optimization-toolbox
- https://scilab.in/fossee-scilab-toolbox/optimization-toolbox/functions
- https://coin-or.github.io/Ipopt/
- https://ampl.com/
- https://github.com/aiobofh/cutest
- https://cute-test.com/guides/cute-framework-guide/
- https://www.scilab.org/
- https://www.scilab.org/build-toolbox