FOSSEE Fellowship 2019 Report

on

FOSSEE Optimization Toolbox

submitted by

Yash I. Kataria

B. Tech (Hons. in ICT with minors in Computational Science)

DAIICT, Gandhinagar

under guidance of

Prof. Kannan Moudgalya

5th July, 2019

Acknowledgement

The fellowship opportunity I had with FOSSEE Team was a great chance for me to learn and experience professional software development. Therefore, I consider myself lucky to have been provided with such a wonderful opportunity. I am also grateful for having a chance to meet so many skilled and talented professionals who led me through this internship. Bearing in mind, I'd like to use this opportunity to express my deepest gratitude and special thanks to Mr. Siddharth Agarwal who in spite of being extraordinarily busy with her/his duties, took time out to hear, guide and keep me on the correct path and allowing me to carryout my assigned tasks at their esteemed organization during the training. I express my deepest thanks to Prof. Kannan M. Moudgalya and Prof. Ashutosh Mahajan for taking part in useful decisions and giving necessary advises and guidance and for arranging all facilities to make my life easier. I choose this moment to acknowledge his contribution gratefully.I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. I also hope to continue cooperation with all of you in the future.

Contents

Chapter 1

Introduction

1.1 FOSSEE Optimization Toolbox

FOSSEE Optimization Toolbox (FOT) for Scilab offers several optimization routines including, but not limited to, linear optimization, integer linear optimization, unconstrained optimization, bounded optimization and constrained optimization. The function calls and outputs are similar to those available in Matlab. These routines call optimization libraries in the backend, most of which are COIN-OR libraries. CLP is used for linear programming, CBC and SYMPHONY for integer linear programming, IPOPT (with MUMPS) for nonlinear optimization and Bonmin for integer nonlinear optimization. There are also routines for specific optimization problems like linear and nonlinear least squares, minimax, and goal programming using these solvers.

fminsearch
(Already present in scilab) Find minimum of unconstrained multi-variable function.

fsolve
(Already present in scilab) Solve system of nonlinear equations.

fgoalattain
: Solves a multiobjective goal attainment problem.

fminbnd
: Solves a nonlinear optimization problem on bounded variables.

fmincon
: Solves a general nonlinear optimization problem.

fminimax
: Solves a minimax optimization problem.

fminunc
: Solves an unconstrained optimization problem.

fot_version
: Displays current versions of various libraries and latest git reference id.

intfminbnd
: Solves a mixed-integer nonlinear optimization problem on bounded variables.

intfmincon
: Solves a constrained mixed-integer nonlinear optimization problem.

intfminimax
: Solves a mixed-integer minimax optimization problem.

intfminunc
: Solves an unconstrained mixed-integer nonlinear optimization problem.

intlinprog
Solves a mixed-integer linear optimization problem in intlinprog format with CBC.

intquadprog
Solves an integer quadratic optimization problem.

lsqlin
Solves a linear least squares optimization problem.

lsqnonlin
Solves a nonlinear least squares optimization problem.

lsqnonneg
Solves a nonnegative linear least squares optimization problem.

quadprog
Solves a quadratic optimization problem.

quadprogmat
Solves a quadratic optimization problem (with input in Matlab format).

symphony
Solves a mixed-integer linear optimization problem.

symphonymat
Solves a mixed-integer linear optimization problem (with input in Matlab format).

1.2 Tools and Technologies

1.2.1 Scilab

Scilab is a free and open-source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulations, numerical optimization, and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations. Scilab is one of the two major open-source alternatives to MATLAB, the other one being GNU Octave.

1.2.2 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. Papers in operations research journals on mathematical theory often contain supporting numerical results from computational studies. The software implementations, models, and data used to produce the numerical results are typically not published. The status quote impeded researchers needing to reproduce computational results, make fair comparisons, and extend the state of the art.
The COIN-OR website was launched as an experiment in 2000, in conjunction with 17th International Symposium on Math Programming in Atlanta, Georgia. In 2007, COIN-OR had 25 application projects,[1] including tools for linear programming (e.g., COIN-OR CLP), nonlinear programming (e.g., IPOPT), integer programming (e.g., CBC, Bcp and COIN-OR SYMPHONY), algebraic modeling languages (e.g., Coopr) and more. By 2011, this had grown to 48 projects.[2] COIN-OR is hosted by the Institute for Operations Research and the Management Sciences, INFORMS, and run by the educational, non-profit COIN-OR Foundation.

1.2.3 Tango

TANGO

Trustable Algorithms for
Nonlinear General Optimization

TANGO (Trustable Algorithms for Nonlinear General Optimization) is a set of Fortran routines for Optimization developed at the Department of Applied Mathematics of the State University of Campinas and at the Department of Computer Science of the University of São Paulo, under the coordination of Professor J. M. Martínez. Only well-established methods are included. The codes are easy to use and require minimum previous knowledge. On-line support is provided.
TANGO is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation. Non-free versions of TANGO are available under terms different from those of the General Public License. Professors J. M. Martínez (martinez@ime.unicamp.br, martinezimecc@gmail.com) or E. G. Birgin (egbirgin@ime.usp.br, egbirgin@gmail.com) should be contacted for more information related to such a license, future developments and/or technical support.

Chapter 2

Mac OS Binaries

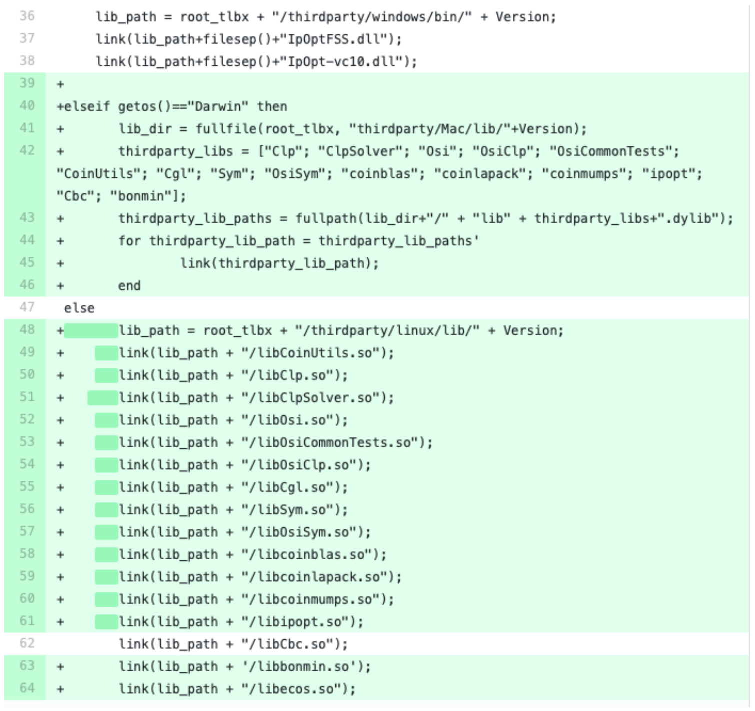
2.1 Compatibility with Mac OS

Previously, the Fossee Optimization toolbox was compatible with Linux and Windows operating system only. For it's compatibility with Mac OS, builder gateway, FOSSEEOptimizationToolbox.start needs to be updated and certain commands had to be changed as the compiler was clang instead of gcc. In the third party folder, a dedicated Mac Folder had to be created where all the dependencies of the functions were to be installed.

2.2 Changes in source code

2.2.1 FOSSEE_Optimization_Toolbox.start

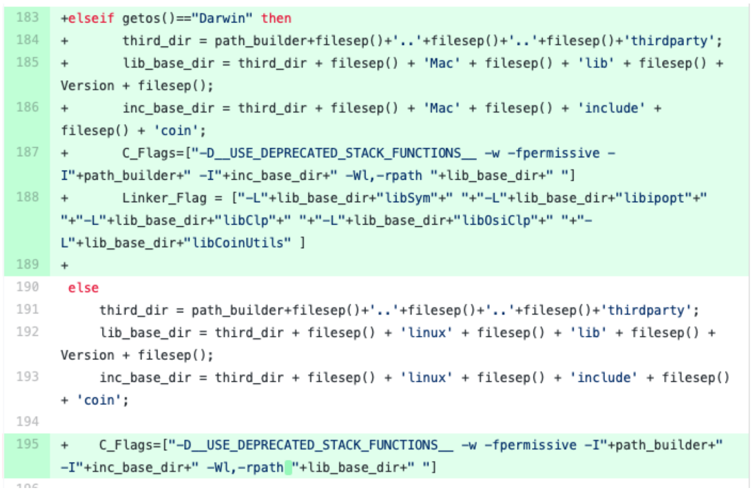
FOSSEE_Optimization_Toolbox.start runs when loader.sce is executed. This script links all the dynamic libraries required by the functions. When it detects Operating System as Darwin i.e. Mac OS, it links the .dylib files mentioned in the code. Here is the code snippet for the following:



```
     lib_path = root_tlbx + "/thirdparty/windows/bin/" + Version;
     link(lib_path+filesep()+"IpOptFSS.dll");
     link(lib_path+filesep()+"IpOpt-vc10.dll");
+
+elseif getos()=="Darwin" then
+       lib_dir = fullfile(root_tlbx, "thirdparty/Mac/lib/"+Version);
+       thirdparty_libs = ["Clp"; "ClpSolver"; "Osi"; "OsiClp"; "OsiCommonTests";
"CoinUtils"; "Cgl"; "Sym"; "OsiSym"; "coinblas"; "coinlapack"; "coinmumps"; "ipopt";
"Cbc"; "bonmin"];
+       thirdparty_lib_paths = fullpath(lib_dir+"/" + "lib" + thirdparty_libs+".dylib");
+       for thirdparty_lib_path = thirdparty_lib_paths'
+               link(thirdparty_lib_path);
+       end
 else
+       lib_path = root_tlbx + "/thirdparty/linux/lib/" + Version;
+       link(lib_path + "/libCoinUtils.so");
+       link(lib_path + "/libClp.so");
+       link(lib_path + "/libClpSolver.so");
+       link(lib_path + "/libOsi.so");
+       link(lib_path + "/libOsiCommonTests.so");
+       link(lib_path + "/libOsiClp.so");
+       link(lib_path + "/libCgl.so");
+       link(lib_path + "/libSym.so");
+       link(lib_path + "/libOsiSym.so");
+       link(lib_path + "/libcoinblas.so");
+       link(lib_path + "/libcoinlapack.so");
+       link(lib_path + "/libcoinmumps.so");
+       link(lib_path + "/libipopt.so");
        link(lib_path + "/libCbc.so");
+       link(lib_path + '/libbonmin.so');
+       link(lib_path + "/libecos.so");
```

2.2.2 Builder_gateway_cpp.sce

Builder_gateway_cpp.sce compiles all the cpp codes and includes and links header files and library files respectively. When it detects OS as Darwin, it compiles the cpp codes and includes all the header files and links all the libraries. Following is the code snippet used to achieve it.



```
+elseif getos()=="Darwin" then
+       third_dir = path_builder+filesep()+'..'+filesep()+'..'+filesep()+'thirdparty';
+       lib_base_dir = third_dir + filesep() + 'Mac' + filesep() + 'lib' + filesep() +
Version + filesep();
+       inc_base_dir = third_dir + filesep() + 'Mac' + filesep() + 'include' +
filesep() + 'coin';
+       C_Flags=["-D__USE_DEPRECATED_STACK_FUNCTIONS__ -w -fpermissive -
I"+path_builder+" -I"+inc_base_dir+" -Wl,-rpath "+lib_base_dir+" "]
+       Linker_Flag = ["-L"+lib_base_dir+"libSym"+" "+"-L"+lib_base_dir+"libipopt"+"
"+"-L"+lib_base_dir+"libClp"+" "+"-L"+lib_base_dir+"libOsiClp"+" "+"-
L"+lib_base_dir+"libCoinUtils" ]
+
 else
     third_dir = path_builder+filesep()+'..'+filesep()+'..'+filesep()+'thirdparty';
     lib_base_dir = third_dir + filesep() + 'linux' + filesep() + 'lib' + filesep() +
Version + filesep();
     inc_base_dir = third_dir + filesep() + 'linux' + filesep() + 'include' + filesep()
+ 'coin';

+    C_Flags=["-D__USE_DEPRECATED_STACK_FUNCTIONS__ -w -fpermissive -I"+path_builder+"
-I"+inc_base_dir+" -Wl,-rpath "+lib_base_dir+" "]
```

2.2.3 Fixed errors in some files

There were certain cpp files which were throwing errors while compiled with clang. Fixed those minor errors. Following is the example of it:

2.3 Dependencies

2.3.1 GCC

In order to use Fossee Optimization Toolbox, certain dependencies are required. One of them is GCC, Fossee_Optimization_Toolbox requires gfortran in order to load. Hence, GCC is required so libgfortran can be linked with the toolbox. In Mac OS, users can install it using Homebrew. After installing Homebrew, users can run this command in terminal:

```
brew install gcc
```

2.3.2 GMP

The toolbox has GMP as a dependency. The full form of GMP is GNU Multiple Precision Arithmetic Library. The toolbox requires libgmp.10.dylib from the gmp package in order to load and execute the functions. libgmp.10.dylib is provided by Scilab, but the version required by the other libraries is incompatible with that provided by Scilab. Hence, libgmp.10.dylib comes pre-installed with our toolbox.

2.3.3 COIN-OR Libraries

FOSSEE Optimization Toolbox uses shared libraries. For different OS, the extension for shared library it is .so for linux, .dylib for Mac OS, .dll for windows. Hence, for different OS, the source code of library needs to be compiled, so as to generate header files and shared libraries for OS that it has been compiled on. The source code was installed by using coinbrew and the library and header files were generated by following the instructions given in coinbrew.

2.4 Features

2.4.1 Support for more Optimization libraries

The shared libraries and header files included in third party folder for MacOS have support for not just the required files but it also has additional shared library and header files for libraries like Couenne, Bcp, CppAD etc. In future, if any additional library is required, we can directly include and use it instead of compiling and installing it.

2.4.2 Shared libraries take less space

The shared libraries in the Mac Folder takes very less space compared to shared libraries in Windows and Linux folder inspite of having support for more shared libraries. In Mac, the size is 21.1 MB whereas for windows and linux it is 31 and 36 MB respectively. Hence, shared libraries in Mac are more compact than linux and windows.

2.5 Problems and their solution

2.5.1 Hardcoded paths in shared libraries

While installing shared libraries through coinbrew, the dependencies in these shared library had hardcoded paths so when using on machine other than the local one, it was not able to locate the dependencies. Also, Coinbrew had no option for giving relative path to the dependencies in .dylib files. Hence, all the path to the dependencies were to be changed manually to relative path. For setting relative path, loader_path was used which searches in the directory of the dependency calling library. Relative path was set using install_name_tool command. After this, any user could install the toolbox and could start using it after installing all the dependencies.

2.5.2 GMP dependency

The toolbox used libgmp.10.dylib present in the gmp package, it came pre-installed in Scilab but the version was 13.0 and all the library had libgmp.10.dylib(version 14.0) as a dependency. Hence, the libgmp.10.dylib needed to be changed in the Scilab folder but Scilab might have files which depended on the previous version of libgmp.10.dylib. Hence, it was included in the thirdparty folder and all the libraries having it as a dependency, their path were changed to relative as discussed in the above section.

2.5.3 GCC dependency

The toolbox uses libgfortran.5.dylib present in the gcc package. As the libgfortran.5.dyib has certain dependencies and also they could not be changed by using install_name_tool command. Hence, it had to be installed by the user. A limitation to it is that users will have to install it using brew. Any other tool like Macports will not work. Hence, users will need to install Homebrew, and then install gcc by using brew command.

2.5.4 fmincon crashes

When using fmincon function in Mac OS, fmincon crashes giving error as Illegal instruction 4 which occurs when a pointer is uninitialized or gets corrupted. After initializing all the pointers, the error still occurred. Hence, the error occurred due to corruption of a pointer. After debugging, the error was found in the code for evaluation of hessian, still the root cause is not found out. But an alternative solution to it is using the inbuilt hessian approximation provided by Ipopt but it gives a large deviation when the input is very large.

Chapter 3

Quadratic Constraint Quadratic Programming Solver

3.1 Quadratic Programming

In mathematical optimization, a quadratic-ally constrained quadratic program (QCQP) is an optimization problem in which both the objective function and the constraints are quadratic functions. It has the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^{\mathrm{T}}P_0x + q_0^{\mathrm{T}}x \\ \text{subject to} & \frac{1}{2}x^{\mathrm{T}}P_ix + q_i^{\mathrm{T}}x + r_i \leq 0 \quad \text{for } i = 1, \ldots, m, \text{ where } P_0, \ldots \\ & Ax = b, \end{array}$$

,P_m are n-by-n matrices and x Rn is the optimization variable. If P_0, …,P_m are all positive semi-definite, then the problem is convex. If these matrices are neither positive nor negative semi-definite, the problem is non-convex. If P_1, …, P_m are all zero, then the constraints are in fact linear and the problem is a quadratic program.

3.2 Build

3.2.1 Function

The function of qcqp macro is to solve quadratic-ally constrained quadratic problem. This was achieved by interfacing Tango's ALGENCAN library to FOT.
ALGENCAN: Fortran code for general nonlinear programming that does not use matrix manipulations at all and, so, is able to solve extremely large problems with moderate computer time. The general algorithm is of Augmented Lagrangian type and the sub problems are solved using GENCAN. GENCAN (included in ALGENCAN) is a Fortran code for minimizing a smooth function with a potentially large number of variables and box-constraints. ALGENCAN has interfaces with AMPL, C/C++, CUTEr, Matlab, Python, Octave and R (statistical computing).

3.2.2 Installation

1. Go to folder $ALGENCAN and type
   `make`
   It will generate the Algencan library file named 'libalgencan.a' within folder $ALGENCAN/lib/.

2. Go to the folder where your main file and problem subroutines are. If you did not code them yet, you may copy the Fortran 90 file toyprob.f90, located at $ALGENCAN/examples/f90/ into your folder.

3. Compile (the example file) typing:
   ```
   gfortran -O3 toyprob.f90 -L$ALGENCAN/lib -lalgencan
   -o algencan
   ```

4. Run typing and see the output in the screen.
   `./algencan`

3.2.3 Dependencies

The following shared libraries are required by Algencan to perform its function.

- `libgfortran`
- `libalgencan`

The libgfortran library depends on multiple GCC libraries. Hence, GCC must be installed in the system. The above libraries are added newly in the `Thirdparty` folder of FOT.

3.2.4 Generating Shared Library

The installation of ALGENCAN builds static archive algencan library by default.By making following changes in the `Makefile` of ALGENCAN, and performing the installation again, we get shared library `libalgencan.so` in the lib folder.

- Inside $ALGENCAN/MAKEFILE
  1. `AR := gcc`
  2. `FFLAGS := -O3 -fPIC`
  3. `CFLAGS := -O3 -fPIC -pedantic -Wall -Wextra -march=native -lgfortran`
- Inside $ALGENCAN/sources/algencan/MAKEFILE
  1. `lib: $(ALGENCAN)`
     `$(AR) $(CFLAGS) $(ALGENCAN) $(HSL) -o libalgencan.so -shared`

2. clean:
```
rm -f *.o
rm -f *.mod
rm -f $(LIB)/libalgencan.so
```

3.2.5 Gateway

The gateway is sci_qcqp.cpp which implements quadratic-ally constrained quadratic problem using ALGENCAN. The interface of ALGENCAN is `c_algencan` that takes references to problem's and result's structure as parameters.

```
void c_algencan(void *myevalf, void *myevalg, void *mye-
valh, void *myevalc, void *myevaljac, void *myevalhc,
void *myevalfc, void *myevalgjac, void *myevalgjacp,
void *myevalhl, void *myevalhlp, int jcnnzmax, int
hnnzmax,double *epsfeas, double *epsopt, double *ef-
stin, double *eostin, double *efacc, double *eoacc,
char *outputfnm, char *specfnm, int nvparam,char **vparam,
int n, double *x, double *l, double *u, int m, dou-
ble *lambda, _Bool *equatn, _Bool *linear, _Bool *coded,_Bool
checkder, double *f, double *cnorm, double *snorm,
double *nlpsupn,int *inform);
```

It takes 11 function pointers to describe problem. A new structure is created to describe the problem in the gateway:

```
struct prob
{
//initial point : x
double * x;
//Objective function : x'Hx+f'x
double ** H;
double * f;
//No of variables
int n;

//Linear inequality constraint Ax ≤ b
double ** A; // m x n
double * b; // m x 1
//No of Linear inequality Constraints
int m;

//Linear equality constraint Aeqx = b
double ** Aeq;
double * beq;
//No of Linear equality constraint
int p;

//Quadratic inequality constraint x'Qx + c'x ≤ r
double *** Q;
double ** c;
double *r;
//No of Quadratic Constraint
int q;

//Lower and Upper bounds
double * lb;
double * ub;

};
```

The following functions are implemented using the above structure and given to `c_alagencan` function.

- `myevalf()` : calculates value of objective function for given $\mathbf{x}$.
- `myevalg()` : calculates value of objective function's gradient for given $\mathbf{x}$.

- `myevalh()` : calculates value of objective function's hessian for given **x**.
- `myevalc()` : calculates value of constraint function gradient for given **x**.
- `myevaljac()` : calculates value of constraint function's gradient for given **x**.
- `myevalhc()` : calculates value of constraint function's hessian for given **x**.

3.3 Documentation

3.3.1 Syntax

```
xopt = qcqp(x,H,f)
xopt = qcqp(x,H,f,Q,c,r,A,b)
xopt = qcqp(x,H,f,Q,c,r,A,b,Aeq,beq)
xopt = qcqp(x,H,f,Q,c,r,A,b,Aeq,beq,lb, ub)
[xopt,fopt,lambda,exitflag] = qcqp( ...  )
```

3.3.2 Inputs

x : a matrix of double, represents initial point.

H : a symmetric matrix of double, represents coefficients of quadratic in the quadratic problem.

f : a vector of double, represents coefficients of linear in the quadratic problem

Q : a n x n.q matrix of double, represents coefficients of quadratic terms in the quadratic constraints. x'.Q.x + c'.x $\leq$ r c : a q x n matrix of double, represents coefficients of linear terms in the quadratic problem. x'.Q.x + c'.x $\leq$ r

r : a vector of double, represents the linear constants in the inequality constraints x'.Q.x + c'.x $\leq$ r.

A : a matrix of double, represents the linear coefficients in the inequality constraints Ax$\leq$ b.

b : a vector of double, represents the linear terms in the inequality constraints Ax$\leq$ b.

Aeq : a matrix of double, represents the linear coefficients in the equality constraints Aeqx = beq.

beq : a vector of double, represents the linear terms in the equality constraints Aeqx = beq.

lb : a vector of double, contains lower bounds of the variables. The default value is 0.

ub : a vector of double, contains upper bounds of the variables. The default value is inf.

3.3.3 Outputs

xopt : a vector of double, the computed solution of the optimization problem.

fopt : a double, the value of the function at x.

lambda : a vector of double, final estimation of the Lagrange multipliers

exitflag : a double, this output parameter tells what happened in this subroutine, according to the following conventions.

The exitflag allows to know the status of the optimization which is given back by Algencan.

exitflag=0 : Optimal Solution Found

exitflag=1 : Maximum Number of Output Iterations Exceeded. Output may not be optimal

exitflag=2 : Maximum Number of Total number of Inner Iterations Exceeded. Output may not be optimal

exitflag=3 : Maximum Number of Total number of Functional Evaluations Exceeded. Output may not be optimal

exitflag=4 : The algorithm stopped by "lack of feasibility progress", i.e., the current point is infeasible

3.4 Examples

//Reference : http://www.minlplib.org/nvs10.html

$$\text{Minimize :} \quad 7 * \mathbf{i_1}^2 + 6 * \mathbf{i_2}^2 - 35 * \mathbf{i_1} - 80.4 * \mathbf{i_2}$$

subject to

$$\begin{aligned} -9 * \mathbf{i_1}^2 - 10 * \mathbf{i_1} * \mathbf{i_2} - 8 * \mathbf{i_2}^2 &\geq -583 \\ -6 * \mathbf{i_1}^2 - 8 * \mathbf{i_1} * \mathbf{i_2} - 6 * \mathbf{i_2}^2 &\geq -441 \\ 0 \leq \mathbf{i_1} &\leq 200 \\ 0 \leq \mathbf{i_2} &\leq 200 \end{aligned}$$

Solution

H = $[70; 06]$;
f = $[-35; -80.4]$;
Q = $[[95; 58]; [34; 43]]$;
c = $[00; 00]$;
r = $[583; 441]$;
x = $[1; 1]$;

```
lb = [0;0];
ub = [200;200];
[xopt,fopt] = qcqp(x,H,f,Q,c,r,[],[],[],[],lb,ub);
```

Chapter 4

Toolbox Review

The documentation of functions were not consistent with the code and it's description. Hence, documentation of many functions were reviewed and the required changes were made to it's documentation. The functions which were reviewed are given as follows:

fgoalattain
: Solves a multiobjective goal attainment problem.

fot_version
: Displays current versions of various libraries and latest git reference id.

linprog
: Solves a linear programming problem.

intlinprog
: Solves a mixed-integer linear optimization problem in intlinprog format with CBC.

lsqlin
: Solves a linear least squares optimization problem.

lsqnonlin
Solves a nonlinear least squares optimization problem.

lsqnonneg
Solves a nonnegative linear least squares optimization problem.

symphony
Solves a mixed-integer linear optimization problem.

symphonymat
Solves a mixed-integer linear optimization problem (with input in Matlab format).

Chapter 5

Conclusion

During the fellowship, I was exposed to open source software and culture. I had a very steep learning curve during the whole tenure of our fellowship. Right from zero experience to understanding various concepts and making something productive and deploy able from it, I progressed a lot and understood what it means to be an IT professional and software engineer. I learnt about APIs, building interface through them, and also developed more proficiency in C++. Also, I learnt about the difficulties when building a cross-platform software, how your code gets impacted when your OS changes?, how to modify it?, how to handle errors and eventually solve them. Apart from technical things, I learnt about time management, Critical and Analytical thinking and Goal Management. Problems faced during our fellowship will help me face challenges in a working environment. At the end, I would like to thank and appreciate everyone who made my fellowship a superb learning and memorable experience.