FOSSEE Fellowship 2019 Report

on

FOSSEE Optimization Toolbox

submitted by

Adarsh T. Shah
B.E.(Computer Engineering)
LDCE, Ahmedabad

under guidance of

Prof. Kannan Moudgalya

5^{th} 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 multivariable 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.

qcqp
: 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) 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

QuadprogCLP

2.1 Linear Constrained Quadratic Problem

Linear Constrained Quadratic Programming is a technique for the optimization of a quadratic objective function, subject to linear equality and linear inequality constraints.
Quadratic programs with linear constraints are problems that can be expressed in canonical form as

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \mathbf{k}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \text{and} \quad & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

2.2 Build

2.2.1 Function

The function of quadprogCLP macro is to solve Quadratic Contrained Linear Problem. This was achieved by interfacing Coin-or's CLP library to FOT.

Clp (Coin-or linear programming) is an open-source linear programming solver written in C++. It is primarily meant to be used as a callable library, but a basic, stand-alone executable version is also available. It is designed to find solutions of mathematical optimization problems of the form

$$\begin{aligned} \text{Minimize :} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{such that :} \quad & \mathbf{row_{lower}} \leq \mathbf{Ax} \leq \mathbf{row_{upper}} \\ & \mathbf{col_{lower}} \leq \mathbf{x} \leq \mathbf{col_{upper}} \end{aligned}$$

2.2.2 Installation

The Clp source code is obtained either via subversion or in form of nightly generated tarballs. The recommended method is to use subversion because it makes it easier to obtain updates. The following commands may be used to obtain and build Clp from the source code using subversion:

1. `svn co https:projects.coin-or.org/svn/Clp /stable/1.16 coin-Clp`
2. `cd coin-Clp`
3. `./configure -C`
4. `make`
5. `make test`
6. `make install`

2.2.3 Dependencies

The CLP library requires following packages in order to perform its function.They are already included in Fossee Optimization Toolbox.

1. BuildTools
2. Blas
3. Lapack
4. Metis
5. Mumps
6. Glpk
7. Sample
8. CoinUtils
9. Osi

2.2.4 Gateway

The gateway is sci_quadprogCLP.cpp that solves QCLP problem using CLP. There are three main classes used to implement the required function.

- CoinPackedMatrix
- CoinPackedVector
- CLPSimplex

CoinPackedMatrix stores **H** of the quadratic objective function and **A** of linear constraints. CoinPackedVector stores **k** of objective function, upper and lower bounds of **x** and constraints **b**. `COIN_DBL_MAX` is the maximum value that an element of Coin's data structure can hold and thus represents infinity in the gateway. ClpSimplex loads the QCLP problem using `loadProblem(constraints, lb, ub, C, row_lb, row_ub)`

and `loadQuadraticObjective(*matrix)`. It then solves problem using `initialSolve()`.
The optimum value of $\mathbf{x}$ is obtained by `getColSolution()`. The optimum value of objective function is obtained by `getObjvalue()`. Various status values are obtained by

- `isProvenOptimal()`
- `isProvenPrimalInfeasible()`
- `isProvenDualInfeasible()`
- `isIterationLimitReached()`
- `isAbandoned()`
- `isPrimalObjectiveReached()`
- `isDualObjectiveReached()`

The number of iterations is obtained using `getIterationsCount()`. The validations regarding correct dimensions of matrices are done in sci_quadprogCLP.cpp. The default value of lower bound is 0 and upper bound is infinity. `quadprogCLP.sci` is the macro that handles assigning default values to undefined terms in the problem.

2.3 Documentation

2.3.1 Syntax

```
xopt = quadprogCLP(H,f,A,b)
xopt = quadprogCLP(H,f,A,b,Aeq,beq)
xopt = quadprogCLP(H,f,A,b,Aeq,beq,lb,ub)
[xopt,fopt,exitflag,output,lamda] = quadprogCLP(...)
```

2.3.2 Inputs

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

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

b : a vector of double, represents the linear coefficients in the inequality constraints $\mathbf{Ax} \leq \mathbf{b}$.

Aeq : a matrix of double, represents the linear coefficients in the equality constraints $\mathbf{Aeqx} = \mathbf{beq}$.

beq : a vector of double, represents the linear coefficients in the equality constraints $\mathbf{Aeqx} = \mathbf{beq}$.

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 infinity.

2.3.3 Outputs

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

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

exitflag : The exit status. See below for details.

iterations : Total number of iterations performed.

output : The structure consist of statistics about the optimization.

lambda : The structure consist of the Lagrange multipliers at the solution of problem.

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

exitflag=0 : Optimal Solution Found

exitflag=1 : Primal Infeasible

exitflag=2 : Dual Infeasible

exitflag=3 : Maximum Number of iterations exceeded

exitflag=4 : Solution Abandoned

exitflag=5 : Primal Objective Limit reached

exitflag=6 : Dual Objective Limit reached

2.4 Examples

Ref : example 14 :
Minimize : $-8 * \mathbf{x_1}^2 - 16 * \mathbf{x_2}^2 + \mathbf{x_1} + 4 * \mathbf{x_2}$
such that :

$$\begin{aligned} \mathbf{x_1} + \mathbf{x_2} &\leq 5 \\ \mathbf{x_1} &\leq 3 \\ \mathbf{x_1} &\geq 0 \\ \mathbf{x_2} &\geq 0 \end{aligned}$$

Solution
`H = [20;08];`
`f = [-8;-16];`

A = [11; 10];
b = [5; 3];
lb = [0; 0];
ub = [inf; inf];
[*xopt*, *fopt*, *exitflag*, *output*, *lambda*] = quadprogCLP(H, f, A, b, [], [], lb, ub)

Solving Linear Programming Problem

$$\min \quad -\mathbf{x_0} - \mathbf{x_1}$$

subject to

$$\mathbf{x_0} + 2 * \mathbf{x_1} \leq 3$$
$$2 * \mathbf{x_0} + \mathbf{x_1} \leq 3$$

Solution

f = [−1; −1];
A = [1, 2; 2, 1];
b = [3; 3];
[*xopt*, *fopt*] = quadprogCLP([], f, A, b);

Chapter 3

Qcqp

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

$$\begin{aligned} \text{minimize} \quad & x^{\mathrm{T}} P_0 x + q_0^{\mathrm{T}} x \\ \text{subject to} \quad & x^{\mathrm{T}} P_i x + q_i^{\mathrm{T}} x + r_i \leq 0 \quad \text{for } i = 1, \ldots, m, \\ & A_{eq} = b_{eq} \\ & Ax \leq b, \end{aligned}$$

where P_0, ... ,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 libal-
   gencan.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

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

subject to

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

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. Some of the functions didn't have documentation. Hence, documentation of many functions were reviewed, new documentations were created and the required changes were made to it's documentation. The functions which were reviewed are given as follows:

fminbnd
: Minor Corrections were done in the description and replaced faulty examples with new ones.

qcqp
: Created entire documentation from scratch with examples.

quadprogCLP
: Created entire documentation from scratch with examples.

quadprog
: Minor Corrections were done in the description and replaced faulty examples with new ones.

quadprogmat

Minor Corrections were done in the description and replaced faulty examples with new ones.

fmincon

No Corrections.

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 to study problem statement and develop efficient solution, how to interface multiple libraries, 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.