Semester-Long Internship Report

On

FOSSEE Optimization Toolbox

Submitted by

Yasa Ali Rizvi

IIIT Senapati, Manipur

Under the guidance of

Prof. Ashutosh Mahajan

Industrial Engineering and Operations Research

IIT Bombay

Mentor

Mr. Rupak Rokade

FOSSEE, IIT Bombay

July 10, 2021

Acknowledgement

The internship opportunity I had with the FOSSEE Team, IIT Bombay, was a great chance for learning and professional development. Therefore, I consider myself a fortunate individual as I was provided with an opportunity to be a part of it. Besides this, it was of special significance to me as I was studying 'Optimization Techniques' this semester, and this internship allowed me to practically apply the concepts that I studied in class. I am also grateful for having a chance to meet so many wonderful people and professionals across the country with whom I worked through this internship period.

I would like to use this opportunity to express my deepest gratitude and special thanks to Prof. Ashutosh Mahajan, professor in Industrial Engineering and Operations Research, for guiding the entire team through the various tasks and taking an active part in all discussions to clarify any doubts that we had regarding the mathematical concepts underlying the various functions provided in the toolbox. His guidance and expertise were of great significance as it assisted us to find the logical errors in otherwise syntactically correct functions, and allowed us to mathematically formulate the problems/test cases correctly.

It is my radiant sentiment to place on record my best regards and deepest sense of gratitude to our mentor, Mr. Rupak Rokade for his continuous support which was extremely valuable for my internship both theoretically and practically. It was his continued aid and encouragement which allowed me to complete all the assigned tasks to the best of my ability.

I perceive this opportunity as a big milestone in my career development. I will strive to use the gained skills and knowledge in the best possible way, and continue to work on their improvement, in order to attain my desired career objectives. Lastly, I wish to express my enthusiasm and hope to continue my work and cooperation with the FOSSEE team in the future.

Contents

1 Introduction **3**
- 1.1 Setting Up the Development Environment 4
  - 1.1.1 GNU/Linux 4
  - 1.1.2 Windows 4
  - 1.1.3 System Specifications 5
- 1.2 Tasks assigned 5

2 FOSSEE Optimization Toolbox **6**
- 2.1 Version Control 6
- 2.2 Structure of the Toolbox 7

3 Exploration of "limited memory" option of Ipopt **10**
- 3.1 Description of the Task 10
- 3.2 Contributions Made 12

4 Fixing of Integer Constraints in fot_intlinprog **15**
- 4.1 Description of the Task 15
- 4.2 Contributions Made 17

5 Debugging FOT Spoken Tutorials **24**
- 5.1 Description of the Task 24
- 5.2 Contributions Made 25

6 Conclusion **29**

Useful Links **30**

Chapter 1

Introduction

Mathematical optimization or mathematical programming is the selection of a best element, with regard to some criterion, from some set of available alternatives. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations constitutes a large area of applied mathematics.

More generally, optimization includes finding "best available" values of some objective function given a defined domain (or input), including a variety of different types of objective functions and different types of domains. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries[1].

Thus, optimization is a significant interdisciplinary field in terms of academic research and study, and being able to contribute to a software which deals with it and is used by several eminent organizations worldwide to solve various real-world problems, is a great opportunity in itself. The FOSSEE Optimization toolbox (FOT) in Scilab allowed me to do just that, and the contributions I made to it helped me to understand this fascinating subject better and instilled in me a sense of curiosity towards its various aspects, which I will indeed continue to pursue in future as part of my higher studies.

This report elaborates on every part of this internship opportunity right from the very beginning, i.e. from setting up our development environment, and continues to describe all the different kinds of contributions that were made by me to the toolbox or the project in general. Section 1.1 describes the process of setting up the development environment while section 1.2 throws light on the tasks that were assigned during this internship – their timelines and a brief overview of each of them.

Chapter 2 gives the reader a glimpse of what the toolbox is, what it comprises of and how its development takes place. Chapters 3, 4 and 5 describe the different tasks and the work done during each of them. Finally, chapter 6 talks about the significance of the work that was done during this internship and the goals that were achieved by being part of it.

[1]https://en.wikipedia.org/wiki/Mathematical_optimization; accessed: 10-July-2021

1.1 Setting Up the Development Environment

In order to work on the toolbox, the development environment needs to be set up. This requires a sequence of steps to be completed which are slightly different for GNU/Linux and Windows. These steps are described in the following sections.

1.1.1 GNU/Linux

1. Fork the source repository and clone it on your system.
2. The source code has the `thirdparty` folder missing. This folder contains the pre-built optimization libraries for windows and linux
3. Download the `thirdparty` folder for your OS from `https://scilab.in/fossee-scilab-toolbox/optimization-toolbox/download-pre-built-optimization-library` and paste it in the toolbox directory.
4. Then type `exec builder.sce` in the Scilab console to run the builder.
5. Now run `exec loader.sce` in the Scilab console. The toolbox will be ready to use.

1.1.2 Windows

1. Fork the source repository and clone it on your system.
2. The source code has the `thirdparty` folder missing. This folder contains the pre-built optimization libraries for windows and linux
3. Download the `thirdparty` folder for your OS from `https://scilab.in/fossee-scilab-toolbox/optimization-toolbox/download-pre-built-optimization-library` and paste it in the toolbox directory.
4. In windows you need MinGW installed along with its toolbox. See `https://atoms.scilab.org/toolboxes/mingw/8.3.0` and Step 0, 1, 2 of `https://github.com/FOSSEE/FOSSEE-Optimization-toolbox/blob/Scilab-6/doc/INSTALL.mingw`
5. Then type `exec builder.sce` in the Scilab console to run the builder.
6. After you build the toolbox successfully, follow instructions given in `https://github.com/FOSSEE/FOSSEE-Optimization-toolbox/blob/Scilab-6/doc/windows.edits`
7. Now run `exec loader.sce` in the Scilab console. The toolbox will be ready to use.

1.1.3 System Specifications

The specifications of the system that was used by me to perform the various development and testing tasks on this toolbox are as follows:

- **RAM :** 4 GB
- **Operating System :** Ubuntu 20.04 (Focal Fossa), Windows 8.1
- **OS Type :** 64-bit

1.2 Tasks assigned

The tasks that were assigned during this internship are as follows:

- **Exploration of "limited memory" option of Ipopt to reduce computation time :** This task involved the addition of the option, `HessianApproximation` to the function `fot_fmincon` provided by the toolbox. This option allows the user the facility of choosing whether he wants the exact Hessian for the Langrangian function to be calculated, which is preferable in case of small problems, or he wants it to be approximated using quasi-Newton approximation method provided by Ipopt via the "limited memory" option, as is generally required for very large/dense problems. The user can toggle the approximation of the Hessian by simply passing a 1 or 0 for the corresponding parameter in the `options` struct for the function. Chapter 3 gives a description of this task and the various contributions that were made to the toolbox as part of it.

- **Fixing of integer constraints in `fot_intlinprog` :** This task mainly involved fixing an issue raised on the FOT GitHub repository regarding the integer constraints not working in the function `fot_intlinprog`. The task fixed the issue thus causing the function to work correctly and return the expected results. Other subtasks that were undertaken during the same time interval were exhaustively testing the entire help section, so as to render it free of all errors, and ensure that each example works fine and returns results as expected by the corresponding problem statements. Chapter 4 gives a description of this task and the various contributions that were made to the toolbox as part of it.

- **Debugging of files and removal of technical discrepancies from FOT Spoken Tutorials :** This final task did not involve any direct contribution to the toolbox, rather it included tasks to check for technical errors in the Spoken Tutorials that are being created for this toolbox. This being said, it involved checking the narration script and slides of the Spoken Tutorials, debugging the associated code files, and adding the solutions to the assignments provided along with each tutorial. Chapter 5 gives a description of this task and the various contributions that were made to the toolbox as part of it.

Chapter 2

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. It uses CLP 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[1]. Section 2.1 throws light on the version control system used by this toolbox, while section 2.2 gives the reader a brief overview of the importance and utility of the different files and directories contained in the toolbox.

2.1 Version Control

The toolbox uses the well known version control system, Git, for source code management. Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows (thousands of parallel branches running on different systems).

To realise the full functionality of Git in software collaboration, the toolbox like so many other Free/Libre and Open Source Software (FLOSS), uses the popular online Git repository hosting service, GitHub. GitHub offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project. As described in section 1.1, the GitHub repository for FOT was forked and cloned on our systems, so that we can easily make any changes locally

[1]https://scilab.in/fossee-scilab-toolbox/optimization-toolbox; accessed: 10-July-2021

and push them upstream, which can then be merged in the main repository by simply raising a pull request.

2.2 Structure of the Toolbox

The toolbox has several folders and files each of which have their own roles in making it work. A summary of the different components of this toolbox is presented below.

demos

The use of the *demos* folder is quite apparent from its name. It contains a collection of files named in the format `<function-name>.dem.sce` which are demos for the functions after which the particular file is named. These files allow the user to demonstrate how the different functions work, and to test the use of the toolbox in general.

doc

The *doc* folder contains files describing the steps to build the toolbox in Windows. The file *INSTALL.mingw* lists the steps to compile the toolbox in Windows using MinGW compiler, while the other file *windows.edits* contains the content that is to added in the loader file within the *sci_gateway/cpp* folder in order to run it successfully, and thus load the toolbox without any errors.

etc

The *etc* folder 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. 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", which is executed when we run the loader. Its purpose is to load the function libraries from macros directory, the gateway and shared libraries from *sci_gateway* and *thirdparty* directory, the help from the *help* directory, and the demos from *demos* directory. 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 unloader. Its purpose is to unlink the toolbox libraries and remove any preferences that were set by the toolbox.

help

The *help* folder contains the all the XML help files for the different functions provided in the toolbox. These files are used to render the help section of the toolbox, and can be modified in order to improve them or correct any errors found in the examples provided by them.

jar

The *jar* folder contains a single file, *scilab_en_US_help.jar* which is used to generate files in the help section.

macros

The *macros* folder is the main folder of the toolbox. It contains Scilab function files (macros) which are compiled when the builder is run. The macro files in turn call C++ files containing gateway functions, which call the optimization solvers and return the solution to the user. The macro files generally contain help page comments, input retrieval, error checks, input modifications, calls to the C++ library, output retrieval, checks and modifications.

sci_gateway

The *sci_gateway* folder contains gateway files, which as the name suggests, act as a gateway between Scilab and C++. The inputs from Scilab are not compatible with other languages and hence Scilab provides an array of API functions to accomplish this. The *sci_gateway* files generally are used to get input form Scilab, pass it to the respective library, retrieve the results and pass the results back to Scilab.

tests

The *tests* folder contains two subfolders – *general_tests* and *unit_tests*. The *unit_tests* folder contains files used to test the various functions to see whether they return the expected results, whereas the folder *general_tests* contains the files to test the different parameters and parts of the functions, and see whether the functions are able to handle all such cases efficiently.

builder.sce

The *builder.sce* file is used to build the toolbox – it creates binaries of the toolbox that are loaded into the Scilab memory. This file executes other builder files which compile the required directories with respective commands. It also builds a *loader.sce* which is used to load the binary files into Scilab memory and a *cleaner.sce* file to remove them.

cleaner.sce

The *cleaner.sce* file is responsible for removing all the binary files, loader and unloader files created by builder.

loader.sce

The *loader.sce* file is responsible for loading the binaries created by builder into the Scilab memory.

unloader.sce

The *unloader.sce* file is responsible for unloading the binaries from the Scilab memory.

changelog.txt

The *changelog.txt* file lists all the changes made in the toolbox across different version releases, in the reverse chronological order.

Chapter 3

Exploration of "limited memory" option of Ipopt

The first task of this internship involved reading the comments given to the FOSSEE team by CNES (*Centre national d'études spatiales*), the French government space agency, regarding the improvements that could be made to the function, `fot_fmincon`, so as to enhance its usability and make it as versatile as the similar function provided by MATLAB. One of the many suggestions was the addition of `HessianApproximation` option to allow the user to choose the method of computing the Hessian. Since, the addition of this option was categorized as of "major" importance, this was the task that was taken up during this assignment. Section 3.1 enunciates the assigned task, and section 3.2 mentions the work that was done by me in the course of completing it.

3.1 Description of the Task

CNES mentioned in their comments that the quasi-Newton approximation method of calculating the Hessian is of particular importance as it would enable saving time. This method is available in the library Ipopt via the option "limited-memory". Thus, this task involved the exploration of this option of Ipopt so that it can be used to approximate the Hessian in `fot_fmincon` and save time for problems having dense coefficient matrices, as for such problems calculating the exact Hessian would be too time-consuming.

According to [1], Ipopt (**I**nterior **P**oint **Opt**imizer, pronounced "Eye-Pea-Opt") is an open source software package for large-scale nonlinear optimization. It can be used to solve general nonlinear programming problems of the form:

$$\begin{aligned} \min_{x\in\mathbb{R}} \quad & f(x) \\ \text{s.t.} \quad & g^L \leq g(x) \leq g^U \\ & x^L \leq x \leq x^U \end{aligned} \tag{NLP}$$

where $x \in \mathbb{R}^n$ are the optimization variables (possibly with lower or upper bounds, $x^L \in (\mathbb{R} \cup \{-\infty\})^n$ and $x^U \in (\mathbb{R} \cup \{+\infty\})^n$), $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are the general nonlinear constratints. The functions $f(x)$ and $g(x)$ can be linear or nonlinear and convex or non-convex (but should be twice continuously differentiable). The constraint functions, $g(x)$, have lower and upper bounds, $g^L \in (\mathbb{R} \cup \{-\infty\})^m$ and $g^U \in (\mathbb{R} \cup \{+\infty\})^m$. Note that equality constraints of the form $g_i(x) = \bar{g}_i$ can be specified by setting $g_i^L = g_i^U = \bar{g}_i$. Ipopt implements an interior point line search method that aims to find a local solution of (NLP).

As mentioned in [2], Ipopt has an option to approximate the Hessian of the Lagrangian by a limited-memory quasi-Newton method (L-BFGS). This feature can be used 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 the C or Fortran interface is being used, these functions still need to be implemented, 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 the 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 solves in order to compute the search direction.

In this task, an option called `HessianApproximation` was added to the `options` struct, and the user could toggle the quasi-Newton approximation "on" or "off" by simply passing 1 or 0 as an argument to the parameter. By default, it is off. i.e. the exact Hessian is calculated for any problem. The option can be used 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 in the file *sci_gateway/cpp/sci_ipoptfmincon.cpp* where this occurs (line 205) is shown below, where `ha` takes the value `"exact"` or `"limited-memory"` accordingly.

```
app->Options()->SetStringValue("hessian_approximation", ha);
```

Apart from this, a selection of problems were made from the CUTE set of PrincetonLib which were to be implemented in Scilab as test cases for this newly added option in `fot_fmincon`. According to the information present on [3], 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 his colleagues at Princeton

University. They were translated to GAMS by Andre Savitsky.

3.2 Contributions Made

The role that was assigned to me during this task was that of a **documenter**. As a documenter, I analysed all the different files where changes were made so that I could document them properly, and see what effects did they have on the output of the `fot_fmincon` function. The main files that were changed were the macro and gateway file associated with the function in order to add the option, and the help file to include this newly added feature in the documentation. The sources of the CUTEst test cases, and the procedure to execute them was also properly documented. The documentation for this task can be found **here**.

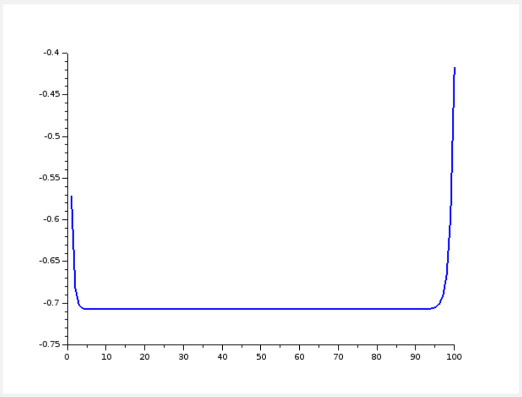
The next contribution that I made as part of this task was adding six CUTEst test cases to the *FOSSEE-Optimization-Toolbox/tests/general_tests/CUTEst* folder. The problems were translated from AMPL to Scilab and such problems were selected which have a large number of variables and equations. In order to test the problems conveniently, the number of variables had to be decreased (for instance, from 10000 to 1000) so as to keep them 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 optimal values of the decision variables computed were plotted alongwith those obtained from NEOS to verify that they are the same. This was done to ease the process of manually cross-checking several hundred values. The values obtained from NEOS have been mentioned on top of each test file for the feasibility of a future user/tester of the code. The problems that were translated by me are as follows:

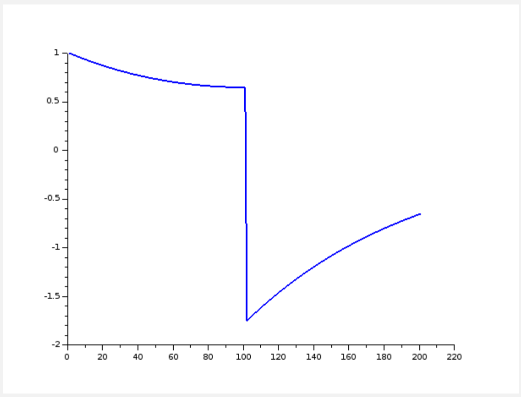
- broydn3d
- hager1
- mancino
- msqrtb
- reading2
- tfi2

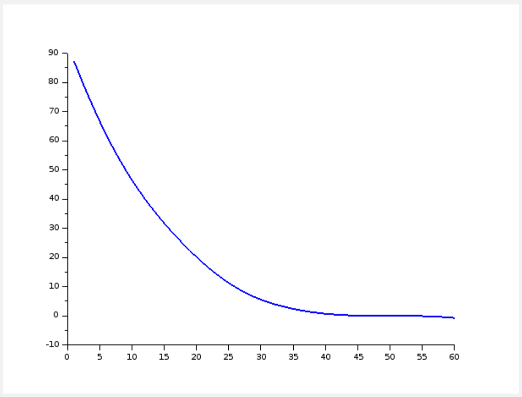
The plots for the optimal values obtained for these problems in Scilab are shown in figure 3.1. The test cases written can be executed in Scilab by the following these steps:

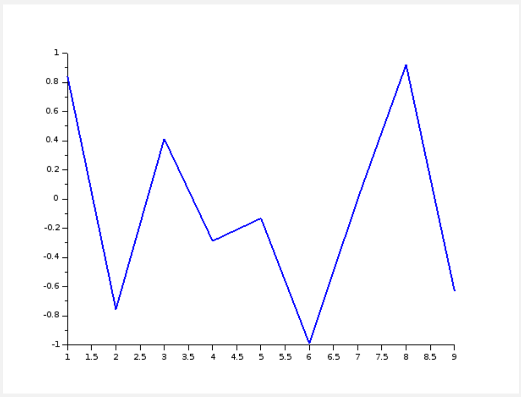
1. Open Scilab and load the FOSSEE Optimization Toolbox by running `exec loader.sce` on the Scilab console. If the toolbox has not been built, run `exec builder.sce` before the previous command.
2. Navigate to the CUTEst tests folder by executing the command:
`cd FOSSEE-Optimization-Toolbox/tests/general_tests/CUTEst`

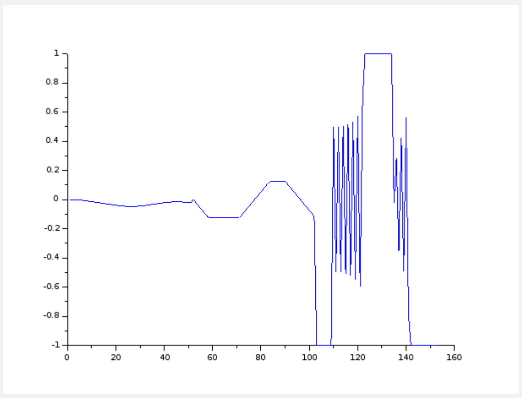
3. Now the individual test cases can be executed by running the corresponding files. A problem can be executed by executing the command – `exec <problem_name>.sce`, where `<problem_name>` can be replaced by any of the problem names mentioned above. For instance, to execute the problem *broydn3d*, execute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


(a) broydn3d


(b) hager1


(c) mancino


(d) msqrtb


(e) reading2

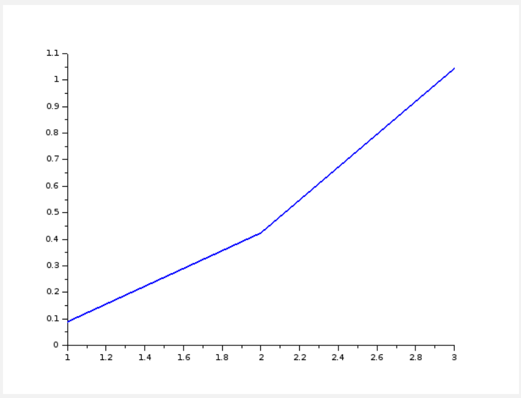

(f) tfi2

Figure 3.1: Plots for optimal values of CUTEst problems

Chapter 4

Fixing of Integer Constraints in fot_intlinprog

The second task of this internship involved fixing an issue raised on the toolbox's GitHub repository (issue #43) regarding the integer constraints in the function `fot_intlinprog`. Section 4.1 describes the assigned task while section 4.2 elaborates the contributions that were made by me to the toolbox during this assignment.

4.1 Description of the Task

The `fot_intlinprog` function available in FOT is used to solve mixed-integer linear optimization problem with CBC (**C**oin-OR **B**ranch-and-**C**ut) solver. According to the issue on GitHub, integer constraints were not working when a user was trying to solve a linear programming problem with integer constraints. He had constrained all the variables to have integral values, however one decision variable was not returning an integral value. The problem executed in Scilab gave the following result.

```
--> A = [-40 0 0 1 0 0 ; 0 -60 0 0 1 0 ; 0 0 -85 0 0 1 ; 0 0 0 -1 -1
   -1];

--> B = [0 0 0 -750]';

--> B = [0 0 0 -750]';

--> C = [200 275 325 1.5 1.8 1.9]';

--> lb = [1 1 1 1 1 1]';

--> ub = [8 5 3 840 560 3*85]';

--> [xopt,fopt,status,output] = fot_intlinprog(C,[1 2 3 4 5
   6]',A,B,[],[],lb,ub)

 xopt  =
   4.875
   5.
   3.
   195.
   300.
   255.
 fopt  =

   4642.
 status =

   0.
 output =

  relativegap = 0
  absolutegap = 0
  numnodes = 0
  numfeaspoints = 0
  numiterations = 4
  numberthreads = 0
  constrviolation = 0
  message = "Optimal Solution"
```

As can be seen in the above result, the first value of `xopt` is a floating point number, when it should have been an integer. This problem was fixed in this task, by correctly passing the integer constraints from the macro to the gateway file, and properly initializing the required variables. After making the necessary changes, the problem was executing correctly and returning the expected results. The result obtained after correcting the integer constraints is shown below.

```
--> A = [-40 0 0 1 0 0 ; 0 -60 0 0 1 0 ; 0 0 -85 0 0 1 ; 0 0 0 -1 -1
   -1];

--> B = [0 0 0 -750]';

--> C = [200 275 325 1.5 1.8 1.9]';

--> lb = [1 1 1 1 1 1]';

--> ub = [8 5 3 840 560 3*85]';

--> [xopt,fopt,status,output] = fot_intlinprog(C,[1 2 3 4 5
   6]',A,B,[],[],lb,ub)

 xopt  =

   5.
   5.
   3.
   200.
   300.
   250.
 fopt  =

   4665.
 status  =

   0.
 output  =

  relativegap = 0
  absolutegap = 0
  numnodes = 2
  numfeaspoints = 6
  numiterations = 7
  numberthreads = 0
  constrviolation = 0
  message = "Optimal Solution"
```

4.2 Contributions Made

The role that was assigned to me during this task was that of a **developer**. As the developer, I investigated all the files pertaining to the function, `fot_intlinprog`. It was found that the integer constraints were being received as a vector in the file */macros/cbcmatrixintlinprog.sci* but the number of integer constraints was not being stored in any variable, and hence not being passed to the gateway file, */sci_gateway/cpp/sci_intlinprog_matrixcpp.cpp*. Similarly, the gateway file had declared a variable `numintcons` with initial value 0 which was not being re-intialized to store the number of integer constraints. Therefore, the following code (lines 184-185 in *sci_intlinprog_matrixcpp.cpp*) to set integer constraints in the solver was not being executed, and consequently, no integer constraints were being passed to the solver causing it to neglect the integer constraints in the problem altogether.

```
for (int i = 0; i < numintcons; i++)                    ...line 184
      solver1.setInteger(intcon[i] - 1);                ...line 185
```

In order to solve this problem, the size of the integer constraints vector (or, the number of integer constraints) was stored as a variable `numintcons` in the macro file *cbcmatrixintlinprog.sci*. A line was added to the file (line 229) in order to implement this change as follows.

```
 numintcons = int32(size(intcon,1));
```

And this variable was passed as the last parameter to the gateway function `matintlinprog` in line 262 of the macro file as shown below.

```
[xopt,fopt,status,nodes,nfpoints,L,U,niter] =
matintlinprog(int32(nbVar),nbCon,c,intcon,conMatrix,conLB,conUB,
lb,ub,objSense,optval,numintcons);
```

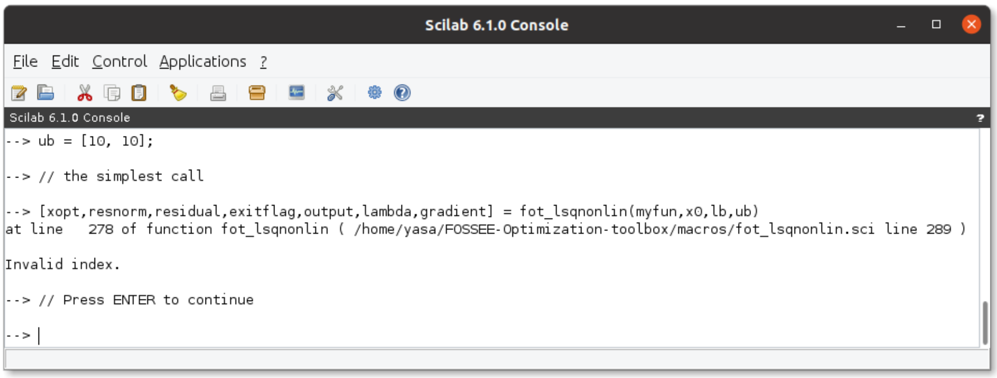
In the gateway file *sci_intlinprog_matrixcpp.cpp*, a line was added (line 99) to intialize the `numintcons` variable so that the integer constraints can be set for the solver, and it can return correct results. The line is shown below.

```
scilab_getInteger32(env, in[11], &numintcons);
```

After making these changes to the pertinent files, the issue of "integer constraints not working" was fixed and the main assignment was completed. Other integer programming functions were also inspected to ensure that they do not have a similar issue with the integer constraints, however no such problem was found. In addition to this, exhaustive testing of the help section was conducted so as to ensure all examples run successfully, and that the code written in each example matches with the problem statement given alongwith it. Several changes were made while testing the examples which are as follows:

- The examples given in the help for `fot_lsqnonlin` were giving an "Invalid Index" error, as shown in the figure 4.1. This is because incorrect indices were being used to reference `options`, which is a struct but was being referred to, as a list. On correcting the indices in lines 289 and 310 in the file */macros/fot_lsqnonlin.sci* from `options(6)` to `options("GradObj")`, the examples were executing correctly, and returning the expected results. The lines that were changed are shown below.

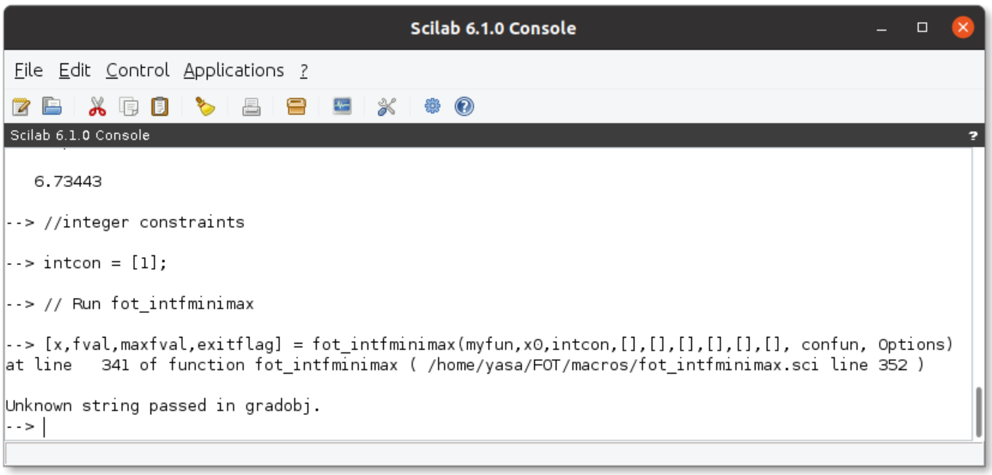
```
if (options("GradObj") == "on")          ...line 289

options("GradObj") = __fGrad;            ...line 310
```



```
--> ub = [10, 10];

--> // the simplest call

--> [xopt,resnorm,residual,exitflag,output,lambda,gradient] = fot_lsqnonlin(myfun,x0,lb,ub)
at line   278 of function fot_lsqnonlin ( /home/yasa/FOSSEE-Optimization-toolbox/macros/fot_lsqnonlin.sci line 289 )

Invalid index.

--> // Press ENTER to continue

--> |
```

Figure 4.1: Error on executing examples in `fot_lsqnonlin`

- In `fot_intfminimax`, example 6 was giving an error indicating "Unknown string passed in gradobj" as shown in figure 4.2, due to the example passing wrong parameters to the `options` struct. The `options` struct accepts only `"on"` or `"off"` values for the parameters `"GradObj"` and `"GradCon"` but the example passed the names of the functions – `"myfungrad"` and `"cgrad"` as the arguments for these parameters, thus producing the error. The example was rewritten so as to pass the correct arguments, and the gradients of the objective function and nonlinear constraints were included in the functions themselves. This change was implemented by modifying lines 415–440 in the file */help/fot_intfminimax.xml*. The corrected example executes successfully as shown in figure 4.3.



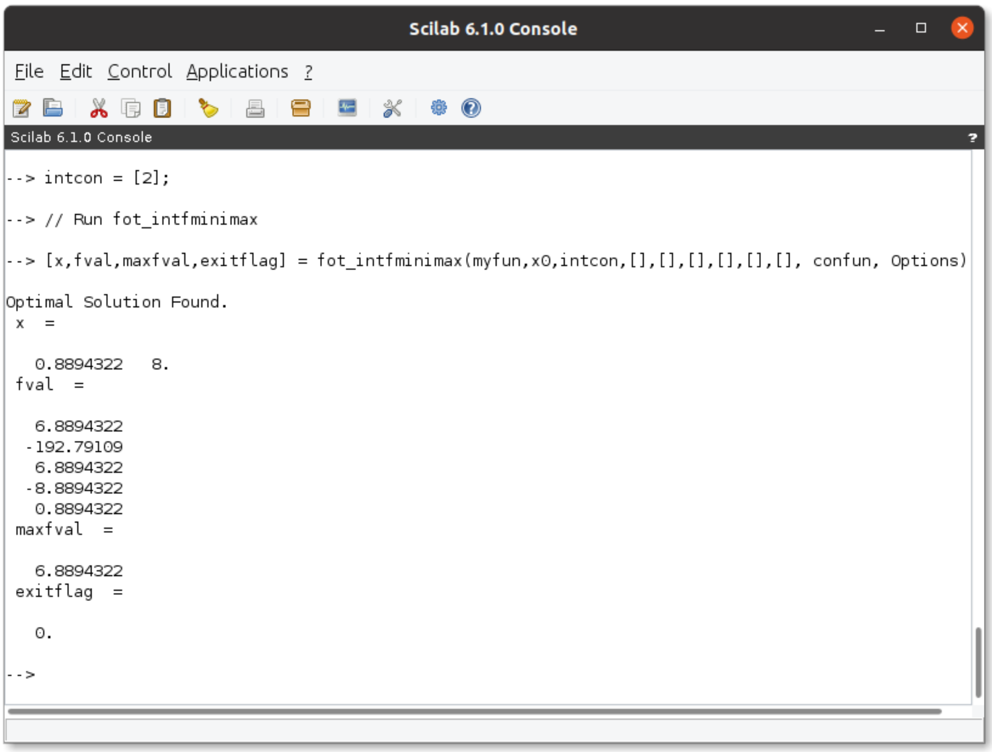
```
  6.73443

--> //integer constraints

--> intcon = [1];

--> // Run fot_intfminimax

--> [x,fval,maxfval,exitflag] = fot_intfminimax(myfun,x0,intcon,[],[],[],[],[],[], confun, Options)
at line   341 of function fot_intfminimax ( /home/yasa/FOT/macros/fot_intfminimax.sci line 352 )

Unknown string passed in gradobj.
--> |
```

Figure 4.2: Error on executing example 6 in `fot_intfminimax`

- Example 7 in `fot_intfminimax` was giving an error indicating wrong size for the input argument as shown in figure 4.4. This error was due to the input arguments being passed in the wrong order – `intcon` should be passed as the third argument after `x0`. On correcting this line in the file */help/fot_intfminimax.xml* (line 478), the example executed successfully. In addition to this, example 5 failed to pass nonlinear constraints `nlc` even though the problem statement specified the same. This was corrected by modifying line 391 in the file */help/fot_intfminimax.xml* to pass `nlc` as an argument to the function as shown below.

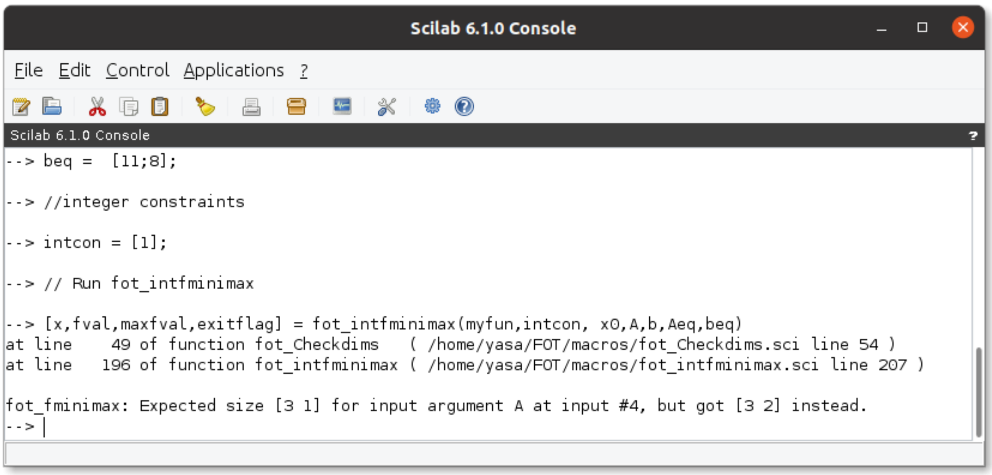
```
[x,fval,maxfval,exitflag] = fot_intfminimax(myfun,
x0,intcon,A,b,Aeq,beq,lb,ub,nlc)
```

- In the function `fot_intfmincon`, problem statements in examples 1 and 6 were corrected to correspond with the code given in the examples. Lines 159, 394, 395 and 401 in the file */help/fot_intfmincon.xml* were modified to implement these changes. Example 4 was executing correctly but an extra parameter – `options`, was being passed to the function. Line 334 in this file was modified



```
--> intcon = [2];

--> // Run fot_intfminimax

--> [x,fval,maxfval,exitflag] = fot_intfminimax(myfun,x0,intcon,[],[],[],[],[],[], confun, Options)

Optimal Solution Found.
 x  =

   0.8894322   8.
 fval  =

   6.8894322
  -192.79109
   6.8894322
  -8.8894322
   0.8894322
 maxfval  =

   6.8894322
 exitflag  =

   0.

-->
```

Figure 4.3: After modification, example 6 in `fot_intfminimax` runs correctly

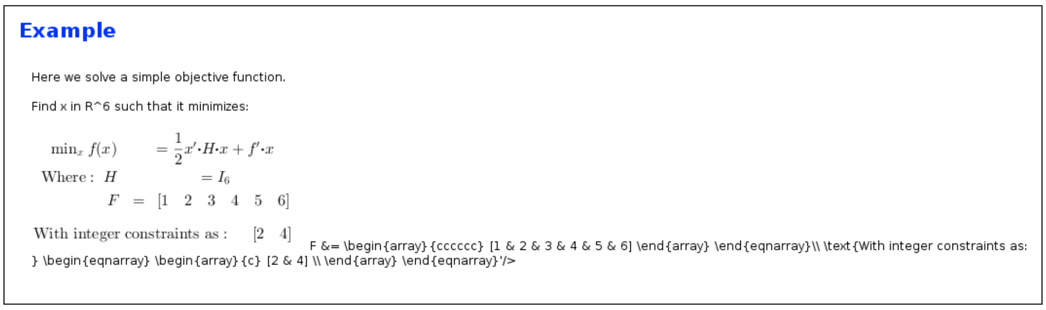


```
--> beq =  [11;8];

--> //integer constraints

--> intcon = [1];

--> // Run fot_intfminimax

--> [x,fval,maxfval,exitflag] = fot_intfminimax(myfun,intcon, x0,A,b,Aeq,beq)
at line    49 of function fot_Checkdims    ( /home/yasa/FOT/macros/fot_Checkdims.sci line 54 )
at line   196 of function fot_intfminimax  ( /home/yasa/FOT/macros/fot_intfminimax.sci line 207 )

fot_fminimax: Expected size [3 1] for input argument A at input #4, but got [3 2] instead.
--> |
```

Figure 4.4: Error on executing example 7 in `fot_intfminimax`

to remove this parameter and ensure that the code was written as specified in the problem statement.

- LaTeX rendering was incorrect for examples 1 and 7 in both `fot_intquadprog` and `fot_quadprogmat` as shown in figures 4.5 and 4.6 respectively. The incorrect rendering was due to the prime symbol used in `eqnarray` environment. The error was corrected by replacing the prime symbol ($\prime$) with superscript 'T' to denote the transpose of matrices. Besides this, other instances of incorrect rendering were resolved by correcting the LaTeX code in multiple places (see the GitHub file changes).



Example

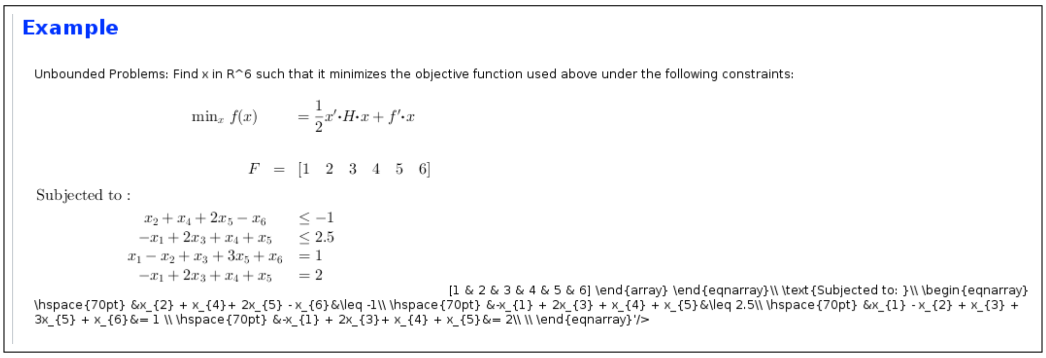
Here we solve a simple objective function.

Find x in R^6 such that it minimizes:

$$\begin{array}{lll} \min_x f(x) & = \frac{1}{2}x'\cdot H\cdot x + f'\cdot x \\ \text{Where}: H & = I_6 \\ F & = [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6] \end{array}$$

With integer constraints as : $[2 \quad 4]$ F &= \begin{array}{cccccc} [1 & 2 & 3 & 4 & 5 & 6] \end{array} \end{eqnarray}\\ \text{With integer constraints as: } \begin{eqnarray} \begin{array}{c} [2 & 4] \\ \end{array} \end{eqnarray}'/>

Figure 4.5: Incorrect LaTeX rendering in `fot_intquadprog`



Example

Unbounded Problems: Find x in R^6 such that it minimizes the objective function used above under the following constraints:

$$\min_x f(x) = \frac{1}{2}x'\cdot H\cdot x + f'\cdot x$$

$$F = [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6]$$

Subjected to :

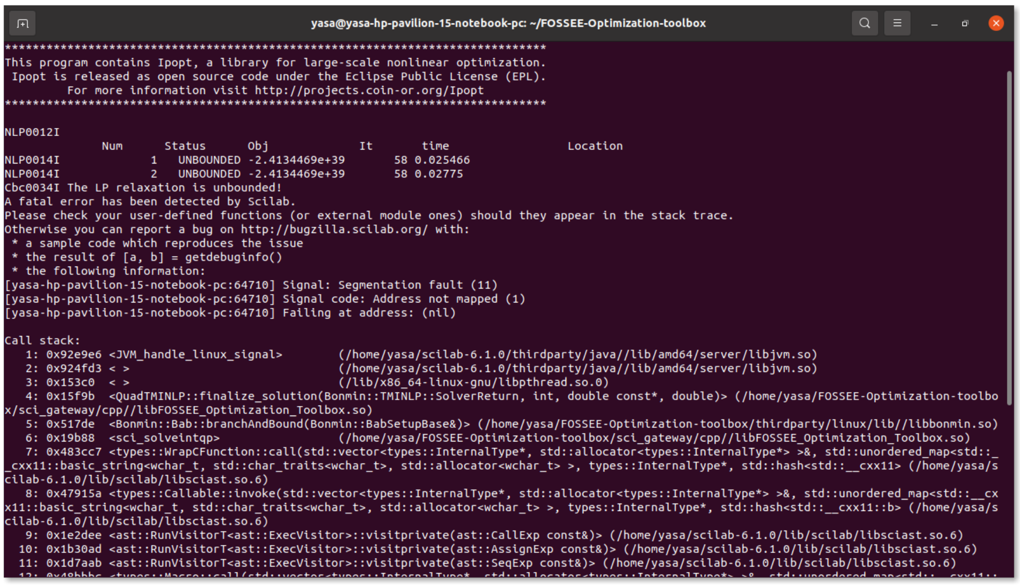
$$\begin{array}{rl} x_2 + x_4 + 2x_5 - x_6 & \leq -1 \\ -x_1 + 2x_3 + x_4 + x_5 & \leq 2.5 \\ x_1 - x_2 + x_3 + 3x_5 + x_6 & = 1 \\ -x_1 + 2x_3 + x_4 + x_5 & = 2 \end{array}$$

[1 & 2 & 3 & 4 & 5 & 6] \end{array} \end{eqnarray}\\ \text{Subjected to: }\\ \begin{eqnarray} \hspace{70pt} &x_{2} + x_{4}+ 2x_{5} - x_{6}&\leq -1\\ \hspace{70pt} &-x_{1} + 2x_{3} + x_{4} + x_{5}&\leq 2.5\\ \hspace{70pt} &x_{1} - x_{2} + x_{3} + 3x_{5} + x_{6}&= 1 \\ \hspace{70pt} &-x_{1} + 2x_{3}+ x_{4} + x_{5}&= 2\\ \\ \end{eqnarray}'/>

Figure 4.6: Incorrect LaTeX rendering in `fot_quadprogmat`

In addition to this, the coefficient matrix for linear equality constraints `Aeq` was corrected in examples 3–7 in both the files, `fot_intquadprog` and `fot_quadprogmat`. All the examples executed successfully after corrections were made to them except examples 6 and 7 in `fot_intquadprog`. An issue is already open on the toolbox's GitHub repository (issue #18) regarding `fot_intquadprog` crashing on infeasible problems, which explains the failure of example 6 to execute. As for example 7 which pertains to unbounded problems, it was giving an optimal solution before correcting the value of `Aeq` when it should have been returning an unbounded solution. On running the example after making the required corrections, it was observed that Scilab crashes due to a segmentation fault error in a somewhat similar fashion as example 6. Thus, it was concluded that `fot_intquadprog` crashes

Scilab on unbounded problems as well. The crash error message obtained can be seen in the figure 4.7.


yasa@yasa-hp-pavilion-15-notebook-pc: ~/FOSSEE-Optimization-toolbox

```
******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Eclipse Public License (EPL).
         For more information visit http://projects.coin-or.org/Ipopt
******************************************************************************

NLP0012I
              Num      Status      Obj             It       time                 Location
NLP0014I             1   UNBOUNDED -2.4134469e+39       58 0.025466
NLP0014I             2   UNBOUNDED -2.4134469e+39       58 0.02775
Cbc0034I The LP relaxation is unbounded!
A fatal error has been detected by Scilab.
Please check your user-defined functions (or external module ones) should they appear in the stack trace.
Otherwise you can report a bug on http://bugzilla.scilab.org/ with:
 * a sample code which reproduces the issue
 * the result of [a, b] = getdebuginfo()
 * the following information:
[yasa-hp-pavilion-15-notebook-pc:64710] Signal: Segmentation fault (11)
[yasa-hp-pavilion-15-notebook-pc:64710] Signal code: Address not mapped (1)
[yasa-hp-pavilion-15-notebook-pc:64710] Failing at address: (nil)

Call stack:
   1: 0x92e9e6 <JVM_handle_linux_signal>        (/home/yasa/scilab-6.1.0/thirdparty/java//lib/amd64/server/libjvm.so)
   2: 0x924fd3 < >                              (/home/yasa/scilab-6.1.0/thirdparty/java//lib/amd64/server/libjvm.so)
   3: 0x153c0  < >                              (/lib/x86_64-linux-gnu/libpthread.so.0)
   4: 0x15f9b  <QuadTMINLP::finalize_solution(Bonmin::TMINLP::SolverReturn, int, double const*, double)> (/home/yasa/FOSSEE-Optimization-toolbo
x/sci_gateway/cpp//libFOSSEE_Optimization_Toolbox.so)
   5: 0x517de  <Bonmin::Bab::branchAndBound(Bonmin::BabSetupBase&)> (/home/yasa/FOSSEE-Optimization-toolbox/thirdparty/linux/lib//libbonmin.so)
   6: 0x19b88  <sci_solveintqp>                 (/home/yasa/FOSSEE-Optimization-toolbox/sci_gateway/cpp//libFOSSEE_Optimization_Toolbox.so)
   7: 0x483cc7 <types::WrapCFunction::call(std::vector<types::InternalType*, std::allocator<types::InternalType*> >&, std::unordered_map<std::_
_cxx11::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >, types::InternalType*, std::hash<std::__cxx11> (/home/yasa/s
cilab-6.1.0/lib/scilab/libsciast.so.6)
   8: 0x47915a <types::Callable::invoke(std::vector<types::InternalType*, std::allocator<types::InternalType*> >&, std::unordered_map<std::__cx
x11::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >, types::InternalType*, std::hash<std::__cxx11::b> (/home/yasa/s
cilab-6.1.0/lib/scilab/libsciast.so.6)
   9: 0x1e2dee <ast::RunVisitorT<ast::ExecVisitor>::visitprivate(ast::CallExp const&)> (/home/yasa/scilab-6.1.0/lib/scilab/libsciast.so.6)
  10: 0x1b30ad <ast::RunVisitorT<ast::ExecVisitor>::visitprivate(ast::AssignExp const&)> (/home/yasa/scilab-6.1.0/lib/scilab/libsciast.so.6)
  11: 0x1d7aab <ast::RunVisitorT<ast::ExecVisitor>::visitprivate(ast::SeqExp const&)> (/home/yasa/scilab-6.1.0/lib/scilab/libsciast.so.6)
  12: 0x48bbbc <types::Macro::call(std::vector<types::InternalType*, std::allocator<types::InternalType*> >&, std::unordered_map<std::__cxx11::
```

Figure 4.7: Scilab crashes on executing unbounded problems in `fot_intquadprog`

- Example 5 in `fot_fminimax` was corrected in order to pass nonlinear constraints `nlc` to the function, as it was required by the problem statement. The change was implemented by updating line 441 in `fot_fminimax.xml` as shown below.

```
[x,fval,maxfval,exitflag,output,lambda] =
fot_fminimax(myfun, x0,A,b,Aeq,beq,lb,ub,nlc)
```

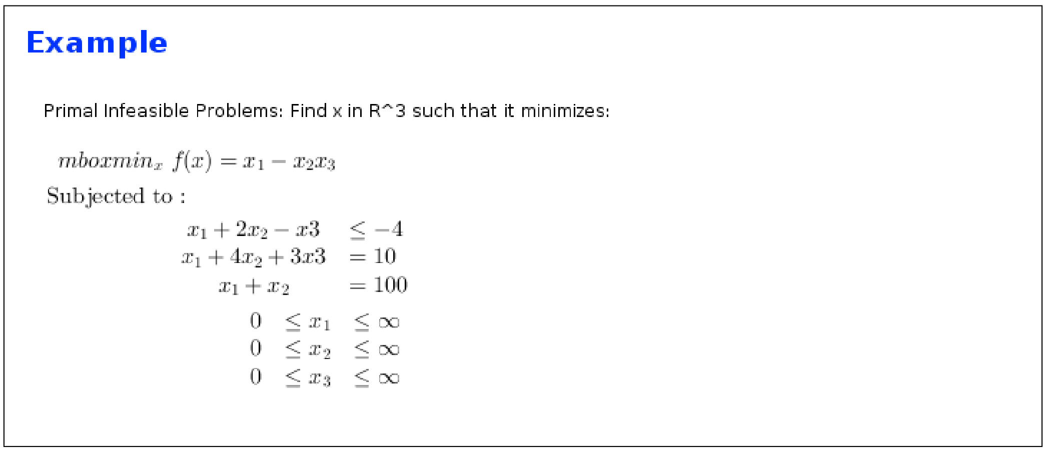
- Example 1 in `fot_quadprogCLP` and all examples in `fot_quadprog` were updated to pass correct values of `H` – the Hessian of the quadratic problem and `f` – the vector containing the coefficients of the linear terms in the quadratic problem. Besides this, the constraints in examples 1, 4 and 5 in `fot_quadprog` were corrected to correspond with the given code.

- The coefficient matrix for linear inequality constraints, `A` was corrected in example 7 of the function `fot_fmincon`, so as to correspond with the constraints specified in the problem statement and return the correct results. Line 461 was updated in the file `fot_fmincon.xml` to implement this change.

- In `fot_fminbnd.xml` lines 42 and 44 were updated to remove LaTeX rendering from the description section of the help, so that it is consistent with the rest of the help files.

- The function name was corrected in a comment in example 7 of `fot_fgoalattain` in order to make it absolutely error free. Line 502 was updated in the file `fot_fgoalattain.xml` in order to implement this change.

- In `fot_linprog`, LaTeX rendering was fixed in examples 4 and 5. The incorrect LaTeX rendering for this function is shown in figure 4.8. Apart from this, the value of `c` – the vector containing coefficients of the objective function was corrected for both the examples, and the value of `Aeq` – the matrix containing the coefficients for linear equality constraints was corrected for example 4.



Example

Primal Infeasible Problems: Find x in R^3 such that it minimizes:

$$mboxmin_x \ f(x) = x_1 - x_2x_3$$

Subjected to :

$$\begin{array}{ll} x_1 + 2x_2 - x3 & \leq -4 \\ x_1 + 4x_2 + 3x3 & = 10 \\ x_1 + x_2 & = 100 \end{array}$$

$$\begin{array}{lll} 0 & \leq x_1 & \leq \infty \\ 0 & \leq x_2 & \leq \infty \\ 0 & \leq x_3 & \leq \infty \end{array}$$

Figure 4.8: Incorrect LaTeX rendering in `fot_linprog`

Besides this, line 41 in the file */sci_gateway/cpp/sci_fotversion.cpp* was updated to change the version of the toolbox to "0.4.1". Lastly, the changelog was written and the Linux 64-bit binary for this toolbox was generated so that it could be published on the ATOMS page alongwith the new version release of this toolbox.

Chapter 5

Debugging FOT Spoken Tutorials

The final task of this internship did not entail any direct development/testing of the toolbox. Rather it involved working on removing all kinds of technical errors from the files on Spoken Tutorials being designed for this toolbox. The aforementioned files consist of the narration script, the slides for various tutorial topics, and the accompanying code files for each tutorial. Section 5.1 sheds light on the Spoken Tutorial project and the assigned task, while section 5.2 describes my contributions to these tutorials.

5.1 Description of the Task

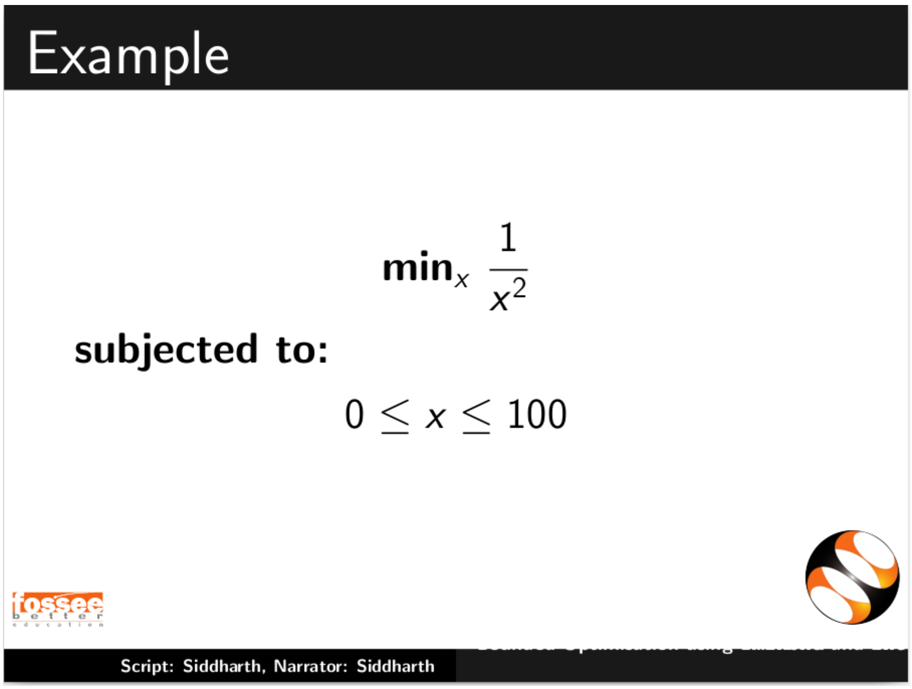
Spoken Tutorial is a multi-award winning educational content portal. According to [4], the platform is where one can learn various Free and Open Source Software all by oneself. Their self-paced, multi-lingual courses ensure that anybody with a computer and a desire for learning, can learn from any place, at any time and in a language of their choice. All the content published on the Spoken Tutorial website are shared under the CC BY SA license. End-of-Course online tests and certificates are available for those who wish to test their expertise in a particular software. These certificates give an edge to students during placement by increasing their employability potential. The Spoken Tutorial project is funded by the National Mission on Education through Information and Communication Technology (ICT), launched by the erstwhile Ministry of Human Resources and Development, Government of India.

The task assigned here involved scanning all the files associated with Spoken Tutorials being prepared for FOT. There were seven tutorials being prepared, viz. the installation of FOT, linear programming using `fot_linprog`, integer programming using `fot_intlinprog`, unconstrained optimization using `fot_fminunc` and `fot_intfminunc`, bounded optimization using `fot_fminbnd` and `fot_intfminbnd`, consrained optimization using `fot_fmincon` and `fot_intfmincon`, and quadratic optimization using `fot_quadprog` and `fot_intquadprog`. These tutorials were distributed among the team, and each member had to remove the technical discrepancies (if any) present in the narration script and slides of the tutorial, check if the associated code files execute without any errors on the Scilab console (and correct them if required), and add solutions to the assignment problems given in each tutorial, after verifying that they execute successfully in Scilab.

5.2 Contributions Made

The tutorials that were worked upon by me during this task were 'bounded optimization using `fot_fminbnd` and `fot_intfminbnd`', and 'constrained optimization using `fot_fmincon` and `fot_intfmincon`'. All the files in both the tutorials referred to the functions as `fminbnd`, `fmincon`, etc. that is without the prefix `fot_`, conforming to an older version of the toolbox. Thus, all occurrences of these names were corrected in the script, slides and code files. The problems specified in the slides were cross-checked with the course outline for the tutorials to ensure that they conform to the guidelines laid out for them. Investigating the slides for the tutorial on bounded optimization revealed several errors in the example problems given as well as the assignment. The changes that were made to this tutorial are as follows:

- The example given for the function `fot_fminbnd` was supposed to be example 1 from the help documentation of the same function, according to the course outline. However, the constraints of the example given in the slides did not match the example given in the help section as shown in figure 5.1. The upper bound for `x` should have been 1000, but the value specified was 100, and hence, it was corrected.



Example

$$\textbf{min}_x \ \frac{1}{x^2}$$

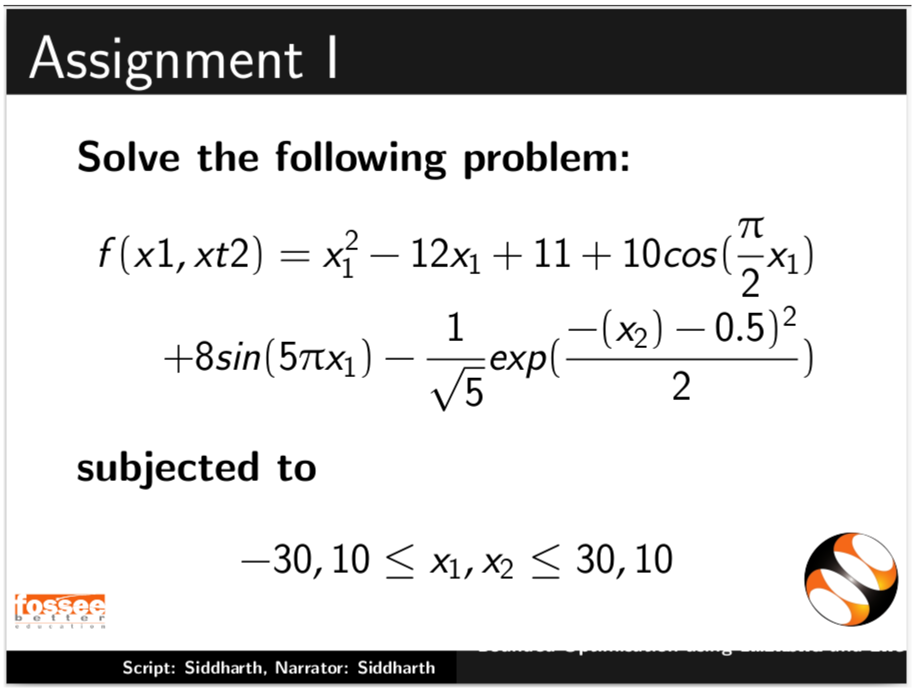
subjected to:

$$0 \leq x \leq 100$$

Script: Siddharth, Narrator: Siddharth

Figure 5.1: Wrong constraints in the `fot_fminbnd` example

- There were typographical errors in the objective function of the assignment problem because of which it became ambiguous as shown in figure 5.2. The same was rectified after referring to the source of the problem, and the slide and narration script was updated. Besides this, the LaTeX rendering was corrected in order to make it more readable and the assignment solution was added to the tutorial files and its answer updated in the slides after verifying it in Scilab.



Assignment I

Solve the following problem:

$$f(x1, xt2) = x_1^2 - 12x_1 + 11 + 10cos(\frac{\pi}{2}x_1)$$
$$+8sin(5\pi x_1) - \frac{1}{\sqrt{5}}exp(\frac{-(x_2) - 0.5)^2}{2})$$

subjected to

$$-30, 10 \leq x_1, x_2 \leq 30, 10$$

Script: Siddharth, Narrator: Siddharth

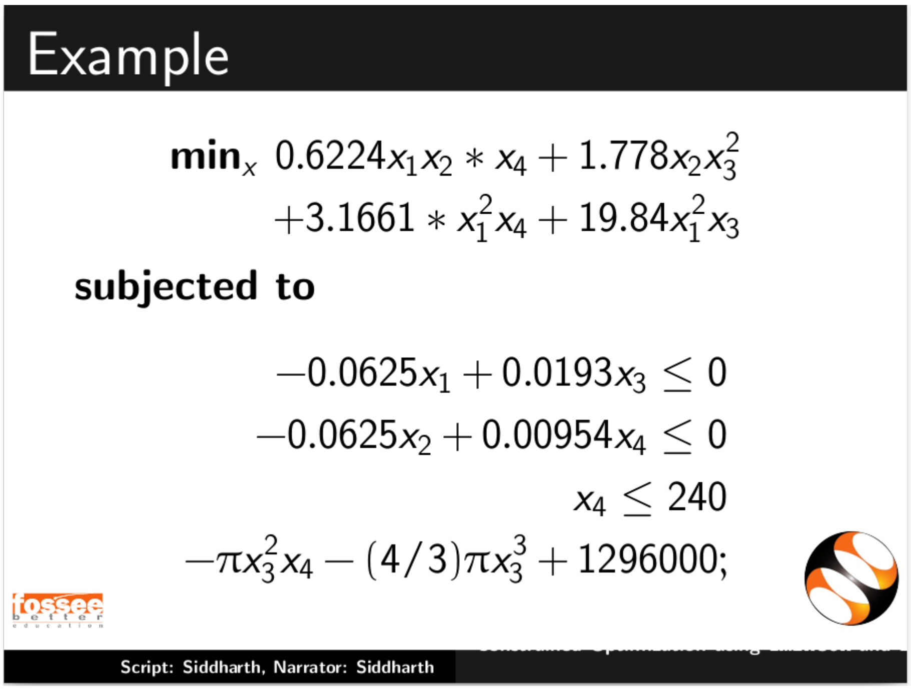
Figure 5.2: Erroneous assignment problem in bounded optimization

- Lastly, certain sections of the script and the comments in the code files `opt_fminbnd.sce` and `opt_intfminbnd.sce` were updated to remove wrong output arguments for the functions and add the correct ones instead.

Similar kind of errors were encountered on examining the files associated with the tutorial on constrained optimization. The changes that were made here are as follows:

- The examples given in the slides for both `fot_fmincon` and `fot_intfmincon` were wrongly written, and did not correspond to their source available here. The examples given had the wrong objective function, wrong specification of the nonlinear constraints, and no mention of the upper and lower bounds for the decision variables as shown in figure 5.3. All these errors were rectified and the slides were updated.

- The integer constraints were being passed in an incorrect manner in the code file `opt_intfmincon.sce` thus giving an "invalid index" error as shown in figure 5.4. The error was due to the integer constraints vector being named as `int` which conflicts with the function available in Scilab for rounding towards zero. It was rectified by simply renaming the vector to `intcon`.



Example

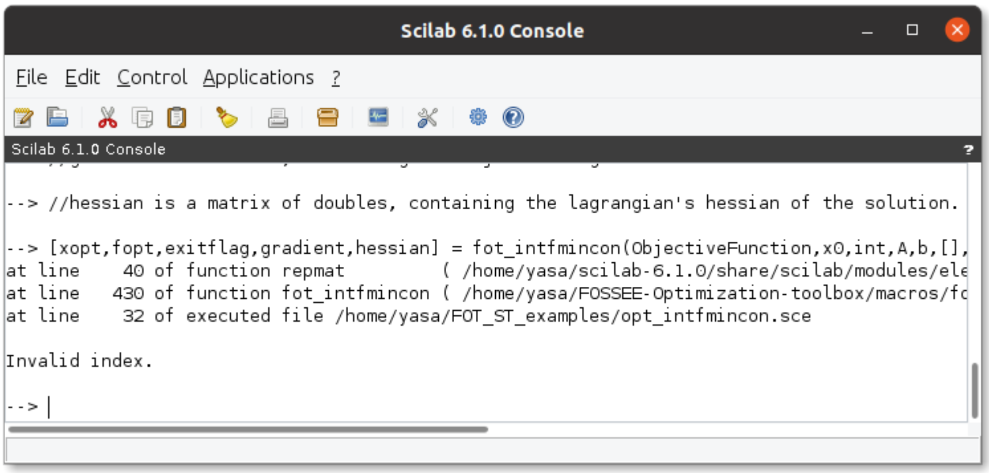
$$\min_x \; 0.6224 x_1 x_2 * x_4 + 1.778 x_2 x_3^2 + 3.1661 * x_1^2 x_4 + 19.84 x_1^2 x_3$$

subjected to

$$-0.0625x_1 + 0.0193x_3 \le 0$$
$$-0.0625x_2 + 0.00954x_4 \le 0$$
$$x_4 \le 240$$
$$-\pi x_3^2 x_4 - (4/3)\pi x_3^3 + 1296000;$$

Script: Siddharth, Narrator: Siddharth

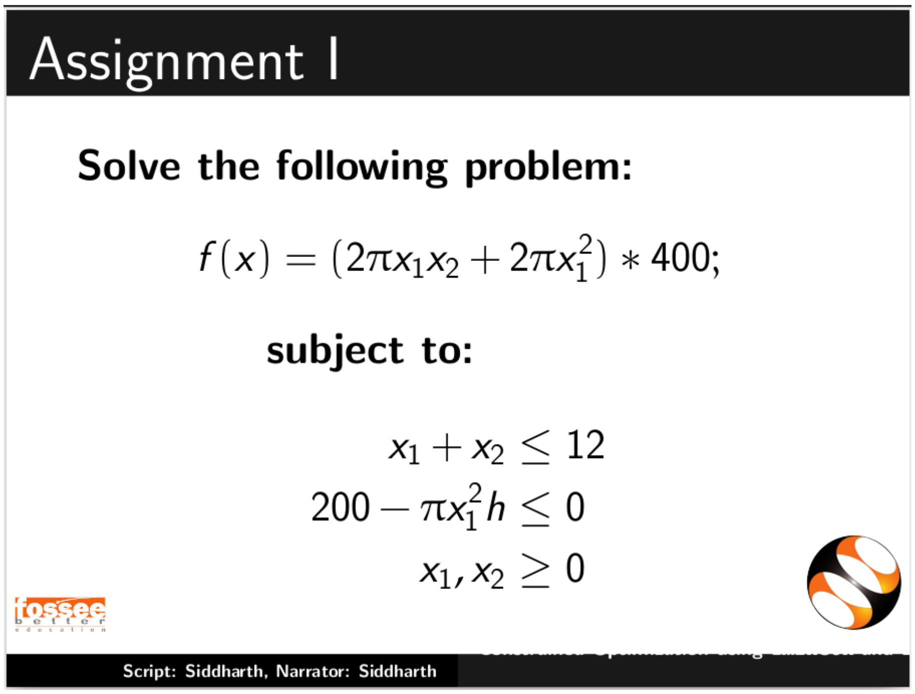
Figure 5.3: Erroneous example for `fot_fmincon` in constrained optimization



```
--> //hessian is a matrix of doubles, containing the lagrangian's hessian of the solution.

--> [xopt,fopt,exitflag,gradient,hessian] = fot_intfmincon(ObjectiveFunction,x0,int,A,b,[],
at line    40 of function repmat         ( /home/yasa/scilab-6.1.0/share/scilab/modules/ele
at line   430 of function fot_intfmincon ( /home/yasa/FOSSEE-Optimization-toolbox/macros/fo
at line    32 of executed file /home/yasa/FOT_ST_examples/opt_intfmincon.sce

Invalid index.

--> |
```

Figure 5.4: Error given in Scilab on executing `opt_intfmincon.sce`

- The assignment problem given in the slides had specified an unknown variable `h` in the constraints, thus making it ambiguous. This is shown in figure 5.5. The source of the problem was referred to deduce the fact that it ought to be replaced with the variable x_2. The correction was made and the slide was updated. Lastly, the assignment solution was added to the tutorial files.



Assignment I

Solve the following problem:

$$f(x) = (2\pi x_1 x_2 + 2\pi x_1^2) * 400;$$

subject to:

$$\begin{aligned} x_1 + x_2 &\le 12 \\ 200 - \pi x_1^2 h &\le 0 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Script: Siddharth, Narrator: Siddharth

Figure 5.5: Assignment in constrained optimization had an unknown variable `h`

- The final changes were made in sections of the narration script and the corresponding comments in the code files, in order to remove the wrong output arguments from the functions and replace them with the correct ones. For instance, the output arguments given in the file `opt_intfmincon.sce`, namely `output` and `lambda` were wrong, and hence they were replaced with the correct ones – `gradient` and `hessian`.

Chapter 6

Conclusion

The contributions made during the various assignments have been able to meet the objectives and deadlines laid out for them by the FOSSEE team. The work done during this internship and the contributions made to the FOSSEE Optimization Toolbox have been quite significant in improving the functionality and correctness of its various aspects. As a result of the changes made to the toolbox by me, the functions `fot_lsqnonlin` and `fot_intlinprog` execute correctly now and return the expected results. The help section has been made free from all kinds of errors thus enhancing its quality and the CUTEst test cases added will aid in the future testing and development of this toolbox. Lastly, the corrections made in the FOT Spoken Tutorial files will help the team recording them do their job correctly, which will in turn prove beneficial to their audience and allow them to successfully use this toolbox for their varied purposes. Thus, to conclude, it is befitting to say that all the work has been done to the best of my abilities and that working with this toolbox has indeed been an enjoyable learning opportunity.

Appendix

Useful Links

- FOT GitHub repository :
  https://github.com/FOSSEE/FOSSEE-Optimization-toolbox
- FOT Website :
  https://scilab.in/fossee-scilab-toolbox/optimization-toolbox
- FOT ATOMS page:
  https://atoms.scilab.org/toolboxes/FOT/0.4.1/
- CUTE set of PrincetonLib:
  http://gamsworld.org/performance/princetonlib/htm/group5stat.htm
- Comments given by CNES on FOT:
  https:
  //drive.google.com/file/d/1V6UCAuEsOfKPok-XplHusKmLUSYQ_AJg/view
- Documentation for Task 1:
  https:
  //drive.google.com/file/d/156fisfx4kcd8h4hCurODk-xbFrEyOyjb/view
- Work Spreadsheet for this Internship:
  https://docs.google.com/spreadsheets/d/
  1xaxIIrAdWMvkfjCuYhF9XRfF9t8APfEZa5BE-Ao2Lpc/edit#gid=0
- My GitHub Repository:
  https://github.com/YasaAliRizvi/FOSSEE-Optimization-toolbox

References

[1] Andreas Wächter (Department of Industrial Engineering and Management Sciences, Northwestern University) Stefan Vigerske (GAMS Software GmbH). Ipopt overview. https://coin-or.github.io/Ipopt/index.html#Overview. [Online; accessed 10-July-2021].

[2] Andreas Wächter (Department of Industrial Engineering and Management Sciences, Northwestern University) Stefan Vigerske (GAMS Software GmbH). Quasi-Newton approximation of second derivatives. https://coin-or.github.io/Ipopt/SPECIALS.html#QUASI_NEWTON. [Online; accessed 10-July-2021].

[3] Hans D. Mittelmann (Arizona State University, Editor in-chief at Performance World). Princeton Library of Nonlinear Programming Models. http://gamsworld.org/performance/princetonlib/princetonlib.htm. [Online; accessed 10-July-2021].

[4] Kannan M. Moudgalya (Department of Chemical Engineering, IIT Bombay and PI, The Spoken Tutorial Project) Nancy Varkey (Content Creation Team Head, The Spoken Tutorial Project). The Spoken Tutorial Project. https://spoken-tutorial.org/about-us/. [Online; accessed 10-July-2021].