Reflection Report on OptLib

Fasil Cheema

April 15, 2024

1 Changes in Response to Feedback

1.1 Problem Statement

Included an abstract definition of function optimization (specifically minimization) in the problem statement. This is in accordance with Dr Smith's comments.

1.2 SRS

Fixed multiple typos and grammatical mistakes.

Added more information to intended reader characteristics.

Modified unclear statements and phrases.

Added a theoretical model for the quadratic form which is very commonly referenced throughout the project.

1.3 Design Documentation (MG)

We modified the anticipated changes (AC) of the MG. Initially following Dr Smith's advice we note that AC's are not parameters but algorithms, datastructures requirements etc that are subject to change. In accordance to this and further analyzing our modules and how they behave with each other we decided on a few ACs that seemed to improve the maintainability, modularity, readability, and our code quality.

We also added an AC for removing the output verify module. This is mainly due to Dr Smiths comment on combining modules that only contain one function. Removing the output verify and incorporating it into the OptLib library is to increase the clarity of the code and avoid having unnecessary modules.

We removed the main module as discussed with Dr Spencer Smith. This is not necessary for this project. The entry point of our library and the control and order of execution will be all done through the OptLib module.

As per Dr Smith's comments chose to change the type of module to Library from ADT for several of the computational modules such as StepSizeCalc, GradCalc, and Search Dir.

Added another AC for the beta computation in the search direction module. This is a heavy computation and to have it in the searchdirFR computation makes the project prone to errors. Having the heavy computation of the beta value in a separate function increases the modularity of the project and allows for errors arising from this subroutine to remain confined to this subroutine. This allows us to test for its validity individually.

The dot product is an AC we have added. This is due to ensuring the result is in a standardized accepted format as opposed to how it was initially. It is documented in the AC2.

The constants module was added as an AC. This is due to having the paramconfigs module act as a singleton that is populated by important variables that are used by many modules. The constants module on the other hand define the behaviour we want our library to be restricted by. For instance, max dimensionality, max number of steps, and min error threshold.

Dr Smith made a comment on combining the multiple modules that may contain a single function. We did this for the output verify module, but chose not to do this for the stepsizecalc, gradcalc, and search dir methods. This is due to each modules' distinct purpose we believe keeping these modules separate allows for better understanding of each modules' purpose and improves clarity of code. The search direction module has multiple functions solely responsible to compute a search direction for the corresponding algorithm, this module already is well encapsulated with the singular purpose of computing this search direction. The stepsizecalc module is used by all algorithms and uses an analytic definition to compute the step size, keeping this seperate highlights this module's key responsibility of computing the step size. Finally the gradcalc is also responsible for the analytic computation of the gradient of a minimization problem in the quadratic form. Keeping this in a separate module highlights this module's key responsibility in computing this important result many other modules depend on.

Added a MG module heirarchy usage figure. This flowchart was not in the initial upload.

1.4 Design Documentation (MIS)

Changed the inconsistency of the vector notation as per Dr Smith's comments. The vectors are denoted as lowercase letters with an arrow above, previously some vectors were also bold faced. Changed so vectors are denoted as lowercase letters with an arrow above (no boldface).

Removed the optimizer solver module and integrated into the main module. This is in accordance to Dr Smith's comments and discussion. This library does not have a main library. We employ a similar design to standard libraries where a module contains all optimizers.

As per Dr Smith's comments we added more details to the inputverify module. We show how an instance of this class initializes the state variables. Responding to Dr Smith's comments we did not just use exceptions for a good reason. This is mainly due to the program accepting invalid outputs which would be accepted without exception only for them to cause issues later in the execution of the program. For instance having a column vector instead of a row vector for the choice of b_vec is accepted without exception but the final result is NaN. This is not the only example, as this problem is persistent due to how numy arrays are defined and how they still work together when they should not. Thus requiring a manual check.

Cleaned up the parameters configuration module. Did this by introducing an initialization function so readers understand how this record was filled.

In the vector math module as per Dr Smith's comments we stated more details on the exported access programs. How many input parameters there are exactly, and what they are.

In the Vecmath module also added state variables as per Dr Smith's comments. This module contains all functions for the vector math. Implemented this in numpy.

In the gradient calculator module added more detailed information about variable and parameter types.

As stated earlier deleted the optimizer module and integrated into a optlib module replacing the main module. Since there is no main module in this project.

In the search direction module we added a local function to take on the role of the beta computation.

1.5 VnV Plan and Report

Added text to the general information section.

Added additional information regarding reviewers as per Dr Smith's comments.

Added more specificity as per Dr Smith and reviewer (Xinyu) comments. In the implementation verification plan we added details of who will implement pylint, how it is used, and what it will do to ensure we are correctly testing.

Added References to the VnV plan.

Added more details to the tests for the functional requirements. Specifically the tests for the input parameters etc.

Fixed test cases, minimized numberof cases to only test if necessary. Original VnV had too many uneccesary tests!

Added many new tests to individually test the gradient calculator, step size calculator, search direction calculator. Correspondingly improved the traceability matrix. Also added tests for the non functional requirements that were not tested prior.

2 Design Iteration (LO11)

Many design changes were made throughout the process. Initially I had a main module where everything was executed. However as discussed with Dr Smith this was not the library I was trying to build so I later changed to a OptLib module which acted as the entry point to the library. I also had many design choices that changed such as having three separate optimizers for each algorithm but decided on one optimizer class with three different optimization algorithm methods. This is to stay standard with the way scipy and other optimization libraries are defined.

3 Design Decisions (LO12)

Dr Smith made a point throughout the course to perhaps think about implementing Software Design Patterns in the library. Specifically, the abstract factory design pattern which is for when creating closely related methods. Although this seemed very appropriate I decided against this after trying to implement this. Since other optimizers are designed within a single class with submethods corresponding to different optimizers, it was difficult to fit the abstract optimizer class into this framework. For this reason I chose to not use this design pattern. That being said the discussion on design patterns was quite useful and I employed the singleton design pattern to enumerate the parameters configuration for easy access to other modules.

4 Economic Considerations (LO23)

N/A

4.1 What Went Well?

When doing the design document, when defining the modules and how they interacted with each other I was forced to really think about the design of this project. I was able to define the mathematical modules in encapsulated methods/functions that their respective jobs. Doing so provided a well defined high level concept of how each module will interact. This also forced me to think about the mathematical details of the specific algorithms and thus made the design document the most critical part of the development process. After finishing the MIS in a rigorous well defined manner I was able to easily create an implementation.

4.2 What Went Wrong?

The technical details that I did not forsee caused me many issues. For example numpy arrays of incompatible shapes can sometimes multiply together and return unexpected answers. This resulted in an additional layer of difficulties as I had to ensure the numpy arrays were of appropriate shapes otherwise unexpected behaviour can occur.

4.3 What Would you Do Differently Next Time?

I would ensure the technical details of each variable, and function are well defined. This is to avoid unexpected type errors and unexpected behaviour.