Software Requirements Specification for : subtitle describing software

February 8, 2024

Contents

1 Reference Material **iii**
- 1.1 Table of Units . . . iii
- 1.2 Table of Symbols . . . iii
- 1.3 Abbreviations and Acronyms . . . iv
- 1.4 Mathematical Notation . . . iv

2 Introduction **1**
- 2.1 Purpose of Document . . . 1
- 2.2 Scope of the Family . . . 2
- 2.3 Characteristics of Intended Reader . . . 3
- 2.4 Organization of Document . . . 3

3 General System Description **4**
- 3.1 System Context . . . 4
- 3.2 User Characteristics . . . 5
- 3.3 System Constraints . . . 5

4 Specific System Description/ Commonalities **5**
- 4.1 Problem Description . . . 6
  - 4.1.1 Terminology and Definitions . . . 6
  - 4.1.2 Physical System Description . . . 7
  - 4.1.3 Goal Statements . . . 7
- 4.2 Solution Characteristics Specification . . . 7
  - 4.2.1 Assumptions . . . 8
  - 4.2.2 Theoretical Models . . . 8
  - 4.2.3 General Definitions . . . 9
  - 4.2.4 Data Definitions . . . 10
  - 4.2.5 Instance Models . . . 12
  - 4.2.6 Input Data Constraints . . . 15

5 Requirements **15**
- 5.1 Functional Requirements . . . 16
- 5.2 Nonfunctional Requirements . . . 16

6 Likely Changes **16**

7 Unlikely Changes **16**

8 Traceability Matrices and Graphs **17**

9 Values of Auxiliary Constants **17**

Revision History

Date	Version	Notes
02/05/2024	1.0	Initial Upload

1 Reference Material

This document is based off of a template of commonality analysis/SRS by Dr Spencer Smith (Smith, 2006).

1.1 Table of Units

Throughout this document there are no physical units. This is due to the abstract mathematical nature of the software.

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units (not applicable in this document). The choice of symbols was made to be consistent with convex optimization literature and with existing documentation for optimization libraries. The symbols are listed in alphabetical order.

symbol	unit	description
$\mathbf{A}$	–	$\mathbf{A}$ is the $n \times n$ matrix that is the leading term in the function (which can be expressed in the quadratic form) which we seek to minimize.
$\vec{\boldsymbol{b}}$	–	$\vec{\boldsymbol{b}}$ is the $n \times 1$ column vector that is the second term in the function (which is in the quadratic form) we are trying to minimize.
c	–	c is the constant (real number) that is the third term in the function (which can be in the quadratic form) we are trying to minimize.
f	–	f is the function of interest which we are trying to minimize.
n	–	n is a natural number and is the dimensionality of our input variable $\vec{\boldsymbol{x}}$, and hence the dimensionality of our problem.
$\vec{\boldsymbol{x}}$	–	$\vec{\boldsymbol{x}}$ is the $n \times 1$ column vector that is the input variable that goes into our function.
$\vec{\boldsymbol{x^*}}$	–	$\vec{\boldsymbol{x^*}}$ is the $n \times 1$ column vector that is optimal (value of $\vec{\boldsymbol{x}}$ that when input into our function results in a minimum value).
$\vec{\boldsymbol{x_0}}$	–	$\vec{\boldsymbol{x^*}}$ is the $n \times 1$ column vector that is the initial (value of $\vec{\boldsymbol{x}}$ that is input into our minimizer).

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
CO	Convex Optimization
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PD	Positive Definite (matrix)
PS	Physical System Description
PSD	Positive Semidefinite (matrix)
R	Requirement
SRS	Software Requirements Specification
TM	Theoretical Model

1.4 Mathematical Notation

In this section we will introduce some mathematical notation which will be seen throughout the project and corresponding documentation. Most of the notation seen in this section and throughout the documentation will be consistent with most texts in the field (Strang, 2009). Boldface uppercase letters will denote matrices as seen in most texts, for example: $\mathbf{M}$ is a matrix. Vectors will always be denoted in boldface, lowercase letters with an arrow above for instance: $\vec{\boldsymbol{x}}$. When a vector or matrix has a superscript T, this denotes the transpose operation; where $\mathbf{M}$ is a $m \times n$ matrix and therefore $\mathbf{M}^T$ becomes a $n \times m$ matrix. We may also utilize the dot product which will be denoted as $\cdot$; and the dot product of two vectors would be denoted as: $\vec{\boldsymbol{x}} \cdot \vec{\boldsymbol{y}}$, given that these two vectors are the same shape. For more discussion on these operations and notation please see the relevant literature (Strang, 2009). The other set of notation which may be seen will be the standard notation for denoting sets, relations, notation of real analysis and convex optimization. This is standardized in math and is consistent throughout textbooks, see for more information (Boyd and Vandenberghe, 2005; Hönig, 1972; Press et al., 2002).

2 Introduction

Function optimization is a critical task in many domains ranging from machine learning to finance. In many of these practical scenarios we are given a function that we seek to minimize or maximize and return an acceptable solution. More specifically when dealing with unconstrained optimization problems the setup is almost always identical; we are given the function we wish to minimize/maximize, we choose a particular optimizer, finally we input a choice of initial value and several hyperparameters relating to the optimizer such as the number of steps, the step size, the initial parameters of the optimizer to name a few. If given an optimization problem that has constraints the first objective is to first convert the problem into an unconstrained one. All our considerations will be assuming no constraints. For the task at hand we will focus specifically on function minimization. In machine learning the corresponding problem would be defining a loss function and then minimizing said loss function. In business the problem becomes defining a cost function in the business and minimizing the cost to maximize profits. Although different domains may have slightly different termiology the abstract problem they are trying to solve is the same.

There are numerous resources to aid in solving problems of this nature, however in this project we would like to focus on a few such optimizers. We would like to develop a library of function minimizers including the David-Fletcher-Powell (DFP), Fletcher-Reeves Conjugate Gradient, and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms.

In this document we would like to outline the Software Requirement Specification (SRS) for our project which is developing a family of optimizers to minimize functions. This section introduces the project, the purpose of this document, the scope of requirements, the characteristics of the intended reader, and the organization of the SRS document.

The introduction section is written to introduce the problem. It starts general and focuses on the problem domain. The general advice is to start with a paragraph or two that describes the problem, followed by a "roadmap" paragraph. A roadmap orients the reader by telling them what sub-sections to expect in the Introduction section.

2.1 Purpose of Document

This document will describe a family of optimizers used for function optimization. This document is meant to act as a reference guide to users who wish to utilize the optimization library. This optimization library will contain several functions that will take a function and corresponding hyperparameters and output a potential solution. This document aims to communicate all necessary details for the design stage and try to communicate the developers' goals throughout this process. This document also acts as a guide to the developers of this document to bring our high level goals to reality and guide development. This document will contain all the major assumptions, goals, theory, and other high level abstractions in one place for users to read. To this end, this document is meant to be abstract and focuses on the high level ideas underpinning the project and not all the technical details required to

actually solve the main problem. This is an initial step in the design of this project and will be heavily referenced in the future especially in the VnV plan (verification and validation).

2.2 Scope of the Family

The scope of the problem starts off with function optimization. Given a mathematical function our goal is to optimize the function which may imply finding all global/local minima or maxima. In the real world function optimization problems may also come with constraints attached to the function, this adds another dimension of difficulty in finding acceptable solutions. the complexity of the problem is hence directly related to the possible functions that can be passed to the optimizer, the convexity (or lack thereof) of the function, the constraints attached to the function, and the choice of optimizer. Given a convex optimization function most optimizers are guaranteed to obtain the global minimum/maximum in a certain number of steps. Even though we may be granted full access to a well defined function obtaining the optimum value is still a computationally difficult task.

We will limit ourselves to problems that are unconstrained. Also, instead of focusing on problems of minimization and maximization we will constrain the library to solve minimization problems specifically. Also to ensure the problem remains feasible we wish to limit the class of possible functions. We limit ourselves to the class of functions that can be expressed in quadratic form; that is quadratic, linear and constant functions. We will also assume these functions will be convex (this comes in later where we assume input matrices are positive semi-definite). Also functions will be described using matrices as we see later on.

Relatively recently there is huge growth in developing optimizers, therefore there is a plethora of optimization algorithms available. However, most algorithms fall into two main families: *Quasi-Newton methods* and *Conjugate-Gradient methods. Quasi-Newton methods* are methods that are based on newton's method when trying to minimize/maximize functions. The basic iterative formula of Newton's method utilizes the Gradient and Hessian (multivariable generalizations of 1st and 2nd order derivatives) to update the search. In *Quasi-Newton methods* we assume we do not have access to the Hessian so we approximate it, utilizing an initial choice of Hessian and updating according to hyperparameters of a particular algorithm. *Conjugate-Gradient methods* on the other hand utilizes conjugate directions to find a minimum/maximum. For our particular case, as we will see, given a symmetric matrix $\mathbf{A}$ we compute conjugate directions with respect to this matrix. These methods utilize just the gradient of the function and conduct a one-dimensional sub-minimization routine.

Both families, generally all optimization problems, boil down to given initial conditions, a function, and necessary hyperparameters computing a search direction then updating and repeating this process until an acceptable solution is found. The 'acceptability' of the solution, the search direction, and conducting updates are determined by the choice of family and moreso the particular algorithm. We can organize the optimizers by the style of choices made. As we discussed both families and their differences there are some key similarities.

After obtaining a search direction update the iteration (this is the current iteration of the initial vector choice $\vec{\boldsymbol{x}}$) both families involve one-dimensional sub-minimization routines.

Clearly this problem domain is very general and abstract. To simplify the problem we apply several restrictions that keeps our problem feasible. We restrict the class of possible functions to simply those that can be expressed in the quadratic form this is our first assumption A1. We also restrict the dimensionality of the input vector to be finite and more explicitly define the max dimension in A3. We also have an assumption, A2, that cater well to our *Conjugate-Gradient methods* where we assume that the matrix $\mathbf{A}$ is postive semi-definite (PSD), which implies it is symmetric (Boyd and Vandenberghe, 2005). This is key because as we stated earlier the nature of the optimum; whether or not there exists local minima/maxima is dependent on this assumption. By assuming the matrix is PSD we know we are dealing with a convex optimization problem and this ensures we will have a global optimum. Finally, we choose to focus on function minimization, this task can be easily extended to function maximization but for this project we will have this sole focus.

Modelling the real world requires simplification. The full complexity of the actual physics, chemistry, biology is too much for existing models, and for existing computational solution techniques. Rather than say what is in the scope, it is usually easier to say what is not. You can think of it as the scope is initially everything, and then it is constrained to create the actual scope. For instance, the problem can be restricted to 2 dimensions, or it can ignore the effect of temperature (or pressure) on the material properties, etc.

The scope section is related to the assumptions section (Section 4.2.1). However, the scope and the assumptions are not at the same level of abstraction. The scope is at a high level. The focus is on the "big picture" assumptions. The assumptions section lists, and describes, all of the assumptions.

2.3 Characteristics of Intended Reader

The intended readers should have preliminary knowledge of function optimization. This would definitely be achieved if the individual has taken a course in convex optimization, numerical optimization, numerical methods, or discrete optimization. The reader should definitely have completed a high school mathematics curriculum and have a good grasp of preliminary mathematics taught in undergraduate degrees such as linear algebra and calculus. A reader should have knowledge on matrix operations, gradients, and Hessians.

2.4 Organization of Document

This document adheres to the Commonality Analysis Template which is related to the SRS template for the development of scientific computing software described by (**?**). This document will follow a top-down approach of going from high-level abstractions down to more specific details. This document contains the goals, assumptions, theory, necessary documentation for the software project. Note that although written with a particular flow of

abstractions to instance models this document can be read ad-hoc for users who seek to utilize the software and readers can feel free to simply go to sections relevant to them.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

The first context can be seen in Figure 1 where a user (can be a program interfacing with the library) uses the library to minimize a function with given parameters. In the figure arrows represent the flow of data when interacting with the library. The user through the programming interface will give the function in appropriate format along with the relevant parameters to the chosen minimizer in the library. This is represented by the arrow from user to the box. The arrow from the box to the user is then represented as the acceptable solution or whatever the final result of the minimizer is back to the user.

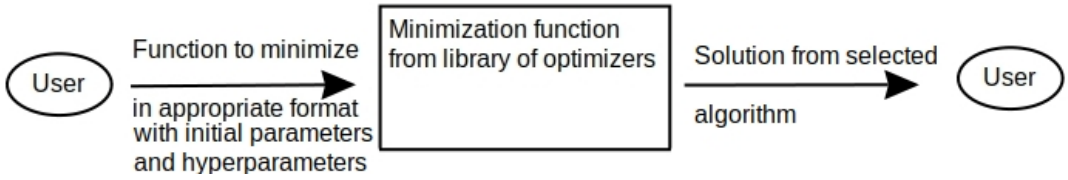

Figure 1: System Context

For each of the entities in the system context diagram its responsibilities should be listed. Whenever possible the system should check for data quality, but for some cases the user will need to assume that responsibility. The list of responsibilities should be about the inputs and outputs only, and they should be abstract. Details should not be presented here. However, the information should not be so abstract as to just say “inputs” and “outputs”. A summarizing phrase can be used to characterize the inputs. For instance, saying “material properties” provides some information, but it stays away from the detail of listing every required properties.

- User Responsibilities:
  - Ensure that the function they input can be expressed in quadratic form

  - Ensure the problem's dimensionality is consistent
  - Ensure the problem's dimensionality does not exceed the limitations of the library
  - Ensure that the problem is convex; $\mathbf{A}$ is PSD
  - Provide valid values of the initial value of $\vec{\boldsymbol{x}}$)
  - Provide valid values of hyperparameters for the algorithm they employ
  - Declare which minimizer they utilize from the library
  - Ensure the function library use complies with the user guide

- Optimization Library Responsibilities:
  - Calculate the minimum value of a valid function if possible.
  - Update the values of the current iteration according to the specific algorithm
  - Determine the step size for the particular algorithm (if applicable for that algorithm)
  - Detect data type mismatch, such as a string of characters instead of a floating point number

3.2 User Characteristics

The user should be well acquainted with interacting with a program library. Since the project will be a library of functions the user should be able to know how to call a particular function and feed it input parameters. The user should also have knowledge of matrices and vectors and matrix operations such as transpose, matrix/vector multiplication etc. Knowledge of Linear Algebra of undergraduate level 2 is needed. The user should also understand the nature of the inputs and their affects on the result. For instance providing a PSD matrix can guarantee a correct result in some finite number of steps, whereas providing a function that is not convex and running a minimizer may not return an acceptable solution. Therefore knowledge of convex optimization at the undergraduate level is needed.

3.3 System Constraints

We constrain our family of minimization solvers to be restricted to 6 dimensions. This is due to the demanding computational costs of conducting matrix operations on high dimensional matrices.

4 Specific System Description/ Commonalities

This section first presents the problem description, which gives a high-level view of the problem. This section contains the common assumptions, theories, definitions and instance models relevant to the family of solvers.

4.1 Problem Description

Function optimization is a problem that is found in a wide variety of fields. Minimization of a cost or loss function is a very common problem in business and machine learning to name a few examples. Therefore, it is imperative to have a set of efficient and correct algorithms to solve this task. The common problem of our optimization library will be to minimize a given function. More specifically, given a function we wish to find an $n-$dimensional vector that is the minimum value of the provided function (subject to no constraints). This is known as an unconstrained function minimization problem.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections. This is to alleviate any confusion in proceeding sections where these terms will show up frequently.

- **Convex function:** is a function where the line between any 2 distinct points on the graph will lie above the graph.
- **Quadratic Form:** is a transformation where given a matrix and a vector; the quadratic form is the transpose of the vector times the matrix times the vector. This returns a real number given that both the vector and the matrix have real entries and are of appropriate shape.
- **Positive Semi-Definite (PSD) Matrix:** A matrix is PSD if it symmetric with real entries and if for any vector such that the quadratic form of this matrix and vector (this is a real number) is either positive or 0.
- **Positive Definite (PD) Matrix:** A matrix is PD if the same conditions hold for PSD except if for any vector the real number derived from the quadratic form of the matrix and vector must strictly be positive.
- **Local Minimum:** A local minimum is a value of a function that is lesser than or equal to all other values of that function within a specified range (hence local).
- **Global Minimum:** A global minimum is a value of a function that is lesser than or equal to all other values of that function.
- **Local Minimum Point:** A local minimum point is a value such that when input into a function, the function returns a value that is lesser than or equal to all other values of that function within a specified range (hence local).
- **Global Minimum Point:** A global minimum point is a value such that when input into a function, the function returns a value that is that is lesser than or equal to all other values of that function.

- **Search Direction:** At each iteration, a minimizer has a current input value and if it is not deemed to be a minimum point the minimizer computes a search direction which should be a vector that 'points' to a point that outputs a function value smaller than the current iteration.

- **Step-Size:** After obtaining a search direction a step size is a real number which indicates the magnitude of how far along the search direction we should go to obtain the next iteration point.It is important to note that when told to use full-steps this indicates always using a step size of 1.

4.1.2 Physical System Description

Given a function that can be expressed in the quadratic form. Assuming this function is also convex we wish to apply one of the family of minimizers to obtain a global minimum. The commonalities between all members of the family are conducting a line search via a search direction and a step size. Both the step size and the search direction are computed in a specific way by a particular algorithm.

The main scenario includes the following elements:

PS1: Step-Size: as defined prior and computed specifically via a particular algorithm

PS2: Search Direction: as defined prior and computed specifically via a particular algorithm

PS3: Function: the function we are trying to minimize

PS4: Initial guess: the first initial 'guess' of the global minimum

4.1.3 Goal Statements

Given a function in an appropriate format, initial starting vector, and the corresponding initial parameters for the particular minimizer the goal statements are:

GS1: Given the function, initial value, input parameters, and stopping conditions minimize the function and return the value for which the function reaches a minimum.

4.2 Solution Characteristics Specification

In this section, information regarding the solution domain of the system is stated. The common themes of the minimizers in the family of minimizers is stated. There are two main families of solvers the *Quasi-Newton* and the *Conjugate-Gradient* methods. Even though there are two families they follow a similar template. They both start off with a starting 'guess' of the point that returns the minimum. They then use their own specific methodology to compute a search direction and a step size (if applicable). They then obtain a new 'guess' of the minimum by adding the search direction times the step size to the initial guess. This

process iterates until a minimum is reached or we have exceeded the maximum number of steps. Eitehr way a value is returned to the user. *Conjugate-Gradient* methods utilize conjugate search directions in the computation of search direction step. *Quasi-Newton* methods compute the search direction with information utilizing a term that is an approximation to the Hessian.

The instance models that govern the Optimization Library are presented in Subsection 4.2.5. The information to understand the meaning of the instance models and their derivation is also presented, so that the instance models can be verified.

4.2.1 Assumptions

This section states assumptions made in defining and setting up the problem. Assumptions are a refinement of the scope and are used when tackling GS1

This section simplifies the original problem into a high level abstraction. It helps in developing the theoretical model by filling in the missing information for main problem of function optimization. In the following sections acronyms in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used.

A1: The input function is convex.

A2: The input function can be expressed in the quadratic form.

A3: The dimensionality of the problem is finite and below the specified max dimension

4.2.2 Theoretical Models

This section provides theoretical models relevant to the optimization library.

RefName: **TM:CON**

Label: Function Convexity

Equation: $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$

Description: The above inequality is the definition of function convexity. For a convex function f, 2 points x_1 and x_2, and $0 \leq \lambda \leq 1$

Notes: None.

Source: https://en.wikipedia.org/wiki/Convex_function

Ref. By: GD1

Preconditions for TM:CON: None

Derivation for TM:CON: Not Applicable

4.2.3 General Definitions

This section collects the general high-level ideas that will be used in building the instance models.

Number	GD1
Label	**Function Minimization**
SI Units	-
Equation	$\arg\min_{\vec{\boldsymbol{x}} \in \mathbb{R}^n} f(\vec{\boldsymbol{x}})$
Description	The standard statement of function minimization. Given a function f, we wish to find the value of $\vec{\boldsymbol{x}}$ that minimizes the function f. For convex functions we are guaranteed to not have local minima only a global minimum.
Source	(Boyd and Vandenberghe, 2005)

Detailed derivation of simplified rate of change of temperature

This may be necessary when the necessary information does not fit in the description field. Derivations are important for justifying a given GD. You want it to be clear where the equation came from.

4.2.4 Data Definitions

This section has data definitions used to build instance models for the optimization library.

Number	DD1
Label	**Dot Product**
Symbol	$\vec{\boldsymbol{x}} \cdot \vec{\boldsymbol{y}}$
SI Units	-
Equation	$\sum_{i=1}^{n} x_i \cdot y_i$
Description	The dot product is an operation between two vectors of the same size. In our equation n is the shared dimension of vectors $\vec{\boldsymbol{x}}$ and $\vec{\boldsymbol{y}}$. The dot product is computed where each corresponding scalar entry from each vector is multiplied together and summed for all entries.
Sources	(Strang, 2009)
Ref. By	IM1,IM2,IM3

Number	DD2
Label	**Transpose**
Symbol	$\vec{\boldsymbol{x}}^T, \mathbf{A}^T$
SI Units	-
Equation	$[\mathbf{A}^T]_{i,j} = [\mathbf{A}]_{j,i}$
Description	The transpose of a matrix (or vector) results in another matrix where the element of the jth row and ith column is the element of the ith row and jth column of the original matrix.
Sources	(Strang, 2009)
Ref. By	IM2,IM3

Number	DD3
Label	**Gradient**
Symbol	∇f
SI Units	-
Equation	$\nabla f = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}]$
Description	The gradient of a scalar function is a n-dimensional vector. Each component of the gradient is the partial derivative of the scalar function with respect to the corresponding one of the n components.
Sources	(Strang, 2009)
Ref. By	IM1,IM2,IM3

Number	DD4
Label	**Hessian**
Symbol	$\mathbf{H}_f$
SI Units	-
Equation	$[\mathbf{H}_f]_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$
Description	The Hessian is a square $n \times n$ matrix, which contains second order partial derivatives of the function f. Each (i, j)th entry is given as the second order partial with respect to the ith and jth component.
Sources	(Strang, 2009)
Ref. By	IM3

4.2.5 Instance Models

This section takes the prior sections' abstractions and generates instance models. This builds off of prior data definitions, assumptions, theoretical models to present a high level view of how to specifically tackle the problem of function minimization.

Number	IM1
Label	**Multi-Dimensional Line Search**
Input	$\vec{\boldsymbol{x}}_0$, ϵ, f, $\max_{\text{steps}}$ The input is constrained so that f can be in the quadratic form (A2) and so that $\vec{\boldsymbol{x}}_0$ is of the proper dimension (A3)
Output	$\vec{\boldsymbol{x}}_{k+1}$, such that $\|\|\nabla f(\vec{\boldsymbol{x}}_{k+1})\|\| \leq \epsilon$ (A1) or $\max_{\text{steps}} > k+1$
Equation	$\vec{\boldsymbol{x}}_{k+1} = \vec{\boldsymbol{x}}_k + \alpha_k \vec{\boldsymbol{s}}_k$, $\vec{\boldsymbol{x}}_{k+1}$ next iteration
Description	α_k is the step-size computed through a sub-routine $\vec{\boldsymbol{x}}_{k+1}$ is the $k+1$-th iteration of our potential minimum . $\vec{\boldsymbol{s}}_k$ is the search direction computed through a sub-routine ϵ is a tolerance parameter (0 implies global minimum is reached for convex functions). $\max_{\text{steps}}$ is a constant term that dictates the maximum number of steps the algorithm can run for. This keeps the problem computationally feasible for test cases that are convex but may require too many steps.
Sources	(Boyd and Vandenberghe, 2005)

Number	IM2
Label	**Conjugate-Gradient search direction**
Input	$\vec{\boldsymbol{x}}_k : \in \mathbb{R}^n$, $\vec{\boldsymbol{x}}_{k-1} : \in \mathbb{R}^n$, $\vec{\boldsymbol{s}}_{k-1} : \in \mathbb{R}^n$
Output	$\vec{\boldsymbol{s}}_k : \in \mathbb{R}^n$ the kth iteration's search direction
Description	$\vec{\boldsymbol{x}}_k$ is the (k)-th iteration of our potential minimum . $\vec{\boldsymbol{x}}_{k-1}$ is the $(k-1)$-th iteration of our potential minimum . $\vec{\boldsymbol{s}}_{k-1}$ is the search direction for the $(k-1)$th iteration.
Sources	(Boyd and Vandenberghe, 2005)
Ref. By	IM1

Number	IM3
Label	**Quasi-Newton search direction**
Input	$\vec{\boldsymbol{x}}_k : \in \mathbb{R}^n$, $\vec{\boldsymbol{x}}_{k-1} : \in \mathbb{R}^n$, $\vec{\boldsymbol{s}}_{k-1} : \in \mathbb{R}^n$ $\boldsymbol{H}_{k-1} : \in \mathbb{R}^{n \times n}$
Output	$\vec{\boldsymbol{s}}_k : \in \mathbb{R}^n$ the kth iteration's search direction
Description	$\vec{\boldsymbol{x}}_k$ is the (k)-th iteration of our potential minimum . $\vec{\boldsymbol{x}}_{k-1}$ is the $(k-1)$-th iteration of our potential minimum . $\vec{\boldsymbol{s}}_{k-1}$ is the search direction for the $(k-1)$th iteration. $\boldsymbol{H}_{k-1}$ is the approximation to the current Hessian used to compute the search direction
Sources	(Boyd and Vandenberghe, 2005)
Ref. By	IM1

4.2.6 Input Data Constraints

The vectors inputted into the minimizer should not exceed $\max_{\text{dim}}$ specified by the developer. This is to keep the problem feasible. The input function should also be convex, this is to guarantee convergence to a global minimum.

5 Requirements

The requirements refine the goal statement. They will make heavy use of references to the instance models.

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: The function and input vector are in the specified dimension which does not exceed maximum defined dimension.

R2: The function provided can be expressed in quadratic form.

R3: Calculation related requirements.

R4: Verification related requirements.

R5: Output related requirements.

Every IM should map to at least one requirement, but not every requirement has to map to a corresponding IM.

5.2 Nonfunctional Requirements

Listed here are the nonfunctional requirements for the optimization library.

NFR1: **Accuracy** The accuracy of the output should be within the provided accuracy parameter provided. This should meet the level required for scientific and engineering applications.

NFR2: **Usability** The software should be usable via interfacing with the program library. See the User Characteristics section for what is expected when interfacing with said library.

NFR3: **Maintainability** The effort required to make any of the likely changes listed for Optimization Library should be less than t_{main} of the original development time. t_{main} is then a symbolic constant that can be found at the end of the report.

NFR4: **Portability** This library should be able to run on any system that can host an operating system and conduct computations from standard programming libraries.

NFR5: **Interpretability** The library should be easily readable to domain experts.

6 Likely Changes

LC1: Potentially downsize number of minimizers in the family.

7 Unlikely Changes

LC2: will contain both Conjugate-Gradient and Quasi-Newton methods.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well. Table 1 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 2 shows the dependencies of instance models, requirements, and data constraints on each other. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

You will have to modify these tables for your problem.

The traceability matrix is not generally symmetric. If GD1 uses A1, that means that GD1's derivation or presentation requires invocation of A1. A1 does not use GD1. A1 is "used by" GD1.

The traceability matrix is challenging to maintain manually. Please do your best. In the future tools (like Drasil) will make this much easier.

	TM:CON	DD1	DD2	DD3	DD4	IM1	IM2	IM3
TM:CON	X							
GD1	X							
DD1		X						
DD2			X					
DD3				X				
DD4					X			
IM1		X	X	X		X		
IM2		X	X	X			X	
IM3		X	X	X	X			X

Table 1: Traceability Matrix Showing the Connections Between Items of Different Sections

9 Values of Auxiliary Constants

$t_{\text{main}} = 0.1$
$\text{max}_{\text{dim}} = 6$

	IM1	IM2	IM3	R1	R2
IM1	X	X	X	X	X
IM2		X		X	
IM3			X	X	
R1				X	
R2					X

Table 2: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A1	A2	A3
TM:CON	X		
GD1	X	X	X
DD1			X
DD2			X
DD3			X
DD4		X	
IM1	X	X	X
IM2	X	X	X
IM3	X	X	X
LC1			
LC2			

Table 3: Traceability Matrix Showing the Connections Between Assumptions and Other Items

References

Stephen P. Boyd and Lieven Vandenberghe. Convex optimization. *Journal of the American Statistical Association*, 100:1097 – 1097, 2005.

Chaim Samuel Hönig. Real analysis. *Boletim da Sociedade Brasileira de Matemática - Bulletin/Brazilian Mathematical Society*, 3:79–80, 1972.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. Numerical recipes in c. 2002.

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. *In Proceedings of the 14th IEEE International Requirements Engineering Conference*, page 209–218, 2006.

Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA, fourth edition, 2009. ISBN 9780980232714 0980232716 9780980232721 0980232724 9788175968110 8175968117.