A bi-level multi-objective optimization algorithm with a bounded multi-variate conjugate gradient method

by

Hong Yoon Kim

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Naval Architecture and Marine Engineering)
in The University of Michigan
2015

For my family

ACKNOWLEDGEMENTS

"Those who stayed will be champions" - Bo Schembechler

I came to Ann Arbor as an eighteen year old boy, and I stayed. My time in Michigan has been truly transformative, and I am thoroughly humbled by love and support I have received. I will not attempt to name everyone who helped me along the way. The list will be too long, and I will inadvertently omit many names that I should not. Dear my friends, I am grateful for your friendship.

I thank United States Department of Defense for its generous financial support via National Defense Science Engineering Graduate fellowship. I would like to thank my advisor Professor Nickolas Vlahopoulos. Prof. Nick, I am so fortunate to have you as my mentor. You treated me with respect, kindness, patience, and dignity. I could not have finished PhD without your encouragement in times of difficulties. By the way, your wife is inexplicably so much better looking than you are; I hope I will do the same.

I would like to thank my family. Dad, you are my inspiration. No words can describe how much I appreciate your sacrifice for our family. Mom, I could not have asked a better mom and friend. Hyung, thank you for being kind and generous to me. I am looking forward to being a better friend with you. I would also like to thank my church in Ann Arbor. HMCC, thank you for helping me to fall in love with Jesus again and again. Lastly, I want to thank Jesus Christ, my Lord and savior, whose great love has inspired me to do the same.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A bi-level multi-objective algorithm with a bounded multi-variate conjugate gradient method

by

Hong Yoon Kim

Chair: Prof. Nickolas Vlahopoulos

Recent growth in computational power has enabled engineers to analyze and to synthesize increasingly more complex systems. The field of Naval Architecture and Marine Engineering is no exception to this trend, and various optimization techniques have been applied to synthesize ship designs and offshore structures with greater complexity.

Designing any complex system requires engaging many areas. Thus, it is necessary for engineers from multiple disciplines, such as hydrodynamics, structures, etc., to collaborate. Although this increased collaboration across multiple disciplines has yielded tremendous benefit, it makes the design process substantially more difficult. Engineers must communicate frequently across all the disciplines, and they can no longer design in isolation. Therefore, efficient algorithms that are capable of facilitating interaction across multiple engineering disciplines are required.

Moreover, the collaboration across multiple disciplines tends to increase the size of the optimization problems. When multiple disciplines are considered, more elements of the system have to be accounted with greater accuracy. As a result, size of engineering optimization problems has increased exponentially in recent years. However, many classical optimization algorithms are not suited to solve optimization problems with a large number of design variables (large-scale).

The goal of this research is to create a flexible multidisciplinary optimization algorithm that is capable of solving large-scale optimization problem by improving known optimization techniques. First, a new bi-level multi-objective optimization algorithm is developed. Engineering systems are designed in a distributed environment where multiple departments design each respective sub-system. However, these departments often have mutually conflicting objectives, and it is necessary to measure the trade-off between different objectives that the system needs to achieve. The new bi-level multi-objective optimization framework finds the trade-off of multiple objectives in a distributed environment. The performance of the new algorithm is demonstrated by solving two multi-objective numerical problems and identifying successfully the Pareto front.

A numerical optimization method called 'Conjugate Gradient (CG) method' is modified to solve optimization problems with a large number of design variables. The CG method is known for its low memory requirement and strong convergence properties. It is one of the earliest large-scale optimization algorithms; the modifications are made to improve its computing time and the rate of convergence for large-scale optimization problems. The performance of the modified CG method is compared with a standard CG method in three numerical problems. The modified Conjugate Gradient is shown to increase the rate of convergence in some cases.

In the last phase of research, the bi-level multi-objective optimization algorithm and the modified conjugate gradient method are combined to create a multi-disciplinary optimization capability suitable for solving problems with a large number of design variables.

A structural multi-objective analysis of a beam is conducted for demonstrating the utility of the new algorithm and its ability to consider many design variables. Technical details and a discussion of the results are presented.

CHAPTER I

Introduction

The profession of engineering revolves around engineers' abilities to design and to build new products. In the past, the engineers heavily relied on legacy designs and empirical (heuristic) methods. As the performance requirements of the new products become more demanding, engineers are often required to design complex systems in absence of legacy designs and empirical methods. Thus, computer analysis tools are integrated into the design processes for exploring the design space of complex engineering systems. However, the synthesis and the decision making process still presents big challenges due to the high number of design variables and constraints, the inter-dependency of engineering disciplines, and the presence of mutually competing objectives [5; 44]. Therefore, a sound design synthesis methodology is required to guide such design process. The main body of the dissertation contains research that improves algorithms in two areas of engineering optimization: multi-disciplinary optimization (MDO) and numerical optimization [27]. In the last phase of the research, the improvements in each respective area are combined to create a MDO algorithm that is capable of solving a multidisciplinary optimization problem with a large number of design variables.

1.1 Multidisciplinary optimization

Multi-disciplinary optimization (MDO) technology synthesizes a design by creating a mathematical framework that encompasses multiple aspects of an engineering system [1]. MDO has emerged as an important engineering field that focuses on optimization strategies for complex engineering systems, and it has been found useful in many industries. Many known MDO architectures, such as Bi-Level Integrated System Synthesis (BLISS) [58; 59], Multi-Objective Collaborative Optimization (MOCO) [60], are distributed, multi-level, and multi-objective. A distributed MDO architecture decomposes an engineering design problem into multiple disciplines, and these disciplines exchange information until a satisfactory design is found. This decomposition is inspired by industry practices in which the design of a system is determined by multiple engineering groups. For instance, ship design involves multiple engineering groups such as hydrodynamics, structure, survivability, etc. Each engineering group controls its own design procedures and uses in-house expertise to solve its engineering problems. Thus, creating a uniform decision making environment is challenging. A distributed MDO algorithm allows each discipline to use its own analysis and optimization tools, and it does not require a uniform computational environment. In addition, the computational load of the MDO problem is distributed among multiple disciplines; this is advantageous when a concentrated computational power is difficult to achieve [44].

A computation sequence and information flow of an MDO algorithm must be carefully considered. Many of current MDO architectures, such as BLISS [58], Collaborative Optimization (CO) [60], Enhanced Collaborative Optimization (ECO) [55], Concurrent Subspace Optimization (CSSO) [57], and Analytical Target Cascading (ATC) [33], use hierarchical structures to organize an MDO problem. In a hierarchical structure, MDO problems are decomposed into multiple levels reflecting the hierarchy of components in a real engineering system. For instance, a landing gear of an airplane has multiple smaller components, such as tire, hydraulics, etc. The design of individual components is not independent of a larger component, and vice versa. A hierarchical optimization propagates information from a higher level to lower levels. Once the computation at the lowest level is completed, the results are sent back to the higher levels. This process repeats until the convergence at the top level is achieved. This structure provides a clear computation sequence and information flow [33].

Multi-Objective (MO) optimization is necessary when a MDO problem contains mutually competing objectives. Multi-Objective MDO optimizes all objectives simultaneously rather than optimizing one objective at a time. In contrast to a single-objective optimization, which finds a single optimal point, the goal of multi-objective optimization is to find a set of multiple Pareto optimal points, which is called 'Pareto front'. An improvement of one objective of a Pareto optimal point cannot be achieved without sacrificing the performance of another objective, and the set of all Pareto optimal points comprises the 'Pareto front'. Identifying the Pareto front is more useful than a determining single optimal point when there exist mutually competing objectives [41; 42; 43].

1.2 Numerical optimization

Engineers aim to design a system with maximum utility. Although finding an optimal design is of great interest in practical applications, it is often difficult to identify a design with an optimal utility using elementary calculus techniques. Thus, finding such solution requires iterative numerical algorithms. The field of numerical optimization deals with mathematic formulations and performances of iterative algorithms. As numerical optimization becomes more prevalent, numerical optimization algorithms are required to solve large-scale problems (i.e. problems with a large number of design variables). Many classical numerical optimization algorithms are ill-suited for large-scale applications. Thus, many algorithms are adapted to handle large-scale problems in recent years [49].

A typical numerical optimization algorithm starts with an initial guess, and it generates a sequence of candidate solutions until a satisfactory solution is found. In general, numerical algorithms can be categorized into two major branches: gradient-based optimization and heuristic (derivative-free) optimization. Gradient-based optimizations use gradient (and Hessian) of the objective function and of the constraints to find an optimal point. Heuristic optimizations are used when the gradient of the objective function is not readily available or when design variables are not continuous [26].

One of the earliest attempts of the gradient-based optimization is the 'steepest decent' method. The algorithm searches along the gradient evaluated at the most recent candidate to find the optimal function value. Although the algorithm is intuitive, the slow rate of convergence made the 'steepest decent' method impractical. Therefore, many alternatives are proposed to increase the rate of convergence. Some known gradient-based optimization algorithms includes Newton's method [48], quasi-Newton's method [69], and Conjugate Gradient (CG) method [26].

Newton's method uses both gradient and Hessian matrix (a matrix that contains partial derivative of second order) to find an optimal solution. In theory, Newton's method has one of the fastest rates of convergence. However computing the Hessian matrix, especially with a large number of design variables, is computationally expensive. To circumvent the difficulty of finding the Hessian matrix, the quasi-Newton's method is proposed. Quasi-Newton's method estimates the Hessian matrix of the objective function as opposed to computing the exact value of Hessian matrix at each iteration. Quasi-Newton's method is considered one of the most successful optimization algorithms, and it has revolutionized the field of numerical optimization since its inception [49].

The Conjugate gradient (CG) method is considered to be the first successful large-scale optimization method. In contrast to the 'steepest decent' method, CG method searches along the eigenvectors of the objective function's contour to reduce the number of iterates. This approach is shown to increase the rate of convergence with relatively low memory requirement. CG method has been successfully implemented in commercial optimization software [63]. Because of the algorithm's historical success in large-scale optimization, it has chosen as a main optimizer in the main body of work.

$$\min \{f(x) : \mathbf{x} \in \mathbb{R}^n\}, \quad lb \leq x \leq ub \tag{1.1}$$

Recently, numerical optimization algorithms are adapted to solve box-constrained optimization problems as shown in Eqn. 1.1. Solving box-constrained optimization problems is important because many practical problems can be converted into this form. In addition, many constrained optimization algorithms, such as augmented Lagrangian or penalty schemes, treats Eqn. 1.1 as a sub-problem [66]. Development of efficient algorithms to solve large-scale box-constrained optimization problems has been an important engineering optimization topic [18; 32].

1.3 Contribution and overview

This dissertation is the result of three distinctive but interrelated phases of research. First is the development of a bi-level multi-objective MDO algorithm. The new algorithm contains a novel multi-objective transformation method that can accommodate the global and local design variable decomposition. In this way, the improvements of each objective can be measured without bias. The proposed algorithm is demonstrated by solving two multi-objective problems. The performance of the algorithm is compared with other multi-objective algorithms and with Monte Carlos solutions.

Then, the development of modified the CG method for large-scale, box-constrained optimization problem is pursued. The modifications to the CG method accelerates the rate of convergence for optimization problem with a large number of box constraints. It also circumvents scaling issues commonly observed in the CG method. The modifications are implemented and tested with three numerical optimization problems with varying degrees of complexity. Once again, the modified algorithm is compared with a standard CG method to demonstrate utility of the modified CG method.

In the last phase of the research, the bi-level multi-objective algorithm is merged with the modified CG method to create a bi-level multi-objective optimization algorithm. The new algorithm is capable of solving multi-objective, large-scale, box-constraints optimization problems with global and local design variables. A structural multi-objective analysis of a beam is conducted for demonstrating the utility of the new algorithm and its ability to consider a large number of design variables.

Chapter 2 through Chapter 5 contain technical information relevant to the new research. Chapter 6 through 8 present the new research in each one of the three aforementioned technical areas.

1.4 Nomenclature

$()_0$: Functions or variables that are shared by more than one discipline

$()_{\hat{i}}$: Functions or variables that apply only to $\hat{i}^{th}$ discipline

$()^*$: Functions or variables at their optimal value

x: Vector of design variables

$x_{(k)}$: k^{th} element of design variables x

x_k: k^{th} iteration of design variables x

$x_{\hat{0}}$: Vector of global design variables

$x_{\hat{i}}$: Vector of local design variables for $\hat{i}^{th}$ discipline

$x_{\hat{0}}^{\hat{i}}$: Vector of global design variables corresponding to $\hat{i}^{th}$ discipline optimum design variables

$(x_{\hat{0}}^{\hat{i}}, x_{\hat{i}})^*$: Vector of optimum design variables for $\hat{i}^{th}$ discipline objective function

$f_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}})$: $\hat{i}^{th}$ discipline objective function

$f_{\hat{i}}^*$: $\hat{i}^{th}$ discipline optimum value

$c_{\hat{i}}$: Vector of $\hat{i}^{th}$ disciple constraints

$ceq_{\hat{i}}$: Vector of $\hat{i}^{th}$ disciple equality constraints

$f_{sys}(x_{\hat{0}} \cdots x_{\hat{k}})$: system level objective function

U: Utopia point, vector of all discipline optimum value $[f_{\hat{1}}^*,\ f_{\hat{2}}^*, \cdots,\ f_{\hat{k}}^*]$

$\langle\langle \mathbf{a} \cdot \mathbf{b} \rangle\rangle$: a dot product of two vectors **a** and **b**.

There are few overlapping standard notations of $\mathbf{x_i}$. Here, we introduce variations of x_i.

- $x_{(i)}$: i^{th} **element** of the design variables, x
- $x_{\hat{i}}$: Vector of $\hat{i}^{th}$ **discipline** local design variables.
- x_i : i^{th} **iteration** of the design variables, x

CHAPTER II

Multi-objective optimization

The process of optimizing a vector of objective functions is called *multi-objective optimization* (MOO) or *vector optimization*. Many engineering design problems have multiple objectives; these objectives are often mutually conflicting. For instance, a ship design often needs to achieve both *maximum stability* and *minimal resistance*. In general, ships with wide beams have good stability, but high resistance. On the other hand, ships with narrow beams have low resistance, but they are not as stable. When there are multiple competing objectives, finding trade-off relationships among the objectives is more important than finding one 'optimal' point. This trade-off among multiple objectives is represented by constructing the 'Pareto front' of the problem. The 'Pareto front' is a collection of all 'Pareto optimal' points. Each one represents a design configuration for which no objective can be improved without sacrificing the performance of any other objective [51]. Multi-objective optimization is a sub-field of optimization that focuses on identifying Pareto front of optimization problems [41; 42; 43]. In this chapter, various methods to find the Pareto optimal points are discussed. The first section of the chapter introduces mathematic notations and basic concepts of multi-objective optimization. Then, a review of the main existing multi-objective optimization methods is presented.

2.1 Definition of a multi-objective optimization problem

The general multi-objective optimization problem is defined as follows:

$$\min_{x\in\mathbb{R}^n} \mathbf{F}(x) = [f_{\hat{1}}(x),\ f_{\hat{2}}(x),\ \cdots, f_{\hat{k}}(x)] \tag{2.1a}$$

subject to

$$c(x) \leq 0 \quad x \in I \tag{2.1b}$$

where multi-objective optimizer identifies a set of 'Pareto' optimal solutions. Here, we use $\hat{i}$ to indicate i^{th} objective function. The formal mathematic definition of *Pareto optimal* is

Definition II.1. Pareto Optimal: A point $x^* \in \mathbb{R}^n$, is Pareto optimal iff there does not exist another point, $x \in \mathbb{R}^n$, such that $\mathbf{F}(x) \leq \mathbf{F}(x^*)$, and $f_{\hat{i}}(x) < f_{\hat{i}}(x^*)$ for at least one function.

In constrained optimization problems with at least one active constraint, all Pareto optimal points lie on the boundary of the feasible space [41]. Often, algorithms provide solutions that may not be Pareto optimal for practical application. Methods for determining Pareto optimality of a point are given in Benson [6], Brosowski and da Silva [8]. Miettinen [47] summarizes the work of Benson [6] with the following common simple test point x^*:

$$\min_{x\in\mathbb{R}^n,\delta\geq 0} \sum_{\hat{i}=1}^{\hat{k}} \delta_{\hat{i}} \tag{2.2}$$

subject to

$$f_{\hat{i}}(x) + \delta_{\hat{i}}, \quad \hat{i} = \hat{1}, \hat{2}, \cdots, \hat{k} \tag{2.3}$$

If all $\delta_{\hat{i}} = 0$, then x^* is a Pareto optimal point.

To find a Pareto optimal point, many optimizers reduce a distance to an ideal case where all objectives attain its respective optimal value. This ideal case is called a *Utopia point* (U) [65].

Definition II.2. Utopia point: A point, $U \in \mathbf{Z}^k$, is a utopia point iff for each $\hat{i} = \hat{1}, \hat{2}, \cdots \hat{k}, \quad U_{\hat{i}} = \min_{x} \{f_{\hat{i}}(x) \mid x \in \mathbb{R}^n\}$

where $\mathbf{Z}^k$ is a k^{th} dimensional feasible set.

In general, U is not feasible. The next best thing is a solution that is as *close* as possible to the utopia point. The precision mathematic definition of *closeness* is open to interpretation. A popular definition of *closeness* is a Euclidean distance to U. The Euclidean distance to the U ($N(x)$) is defined as follows:

$$N(x) = |\mathbf{F}(x) - U| = \left\{ \sum_{\hat{i}=1}^{\hat{k}} [f_{\hat{i}}(x) - U_{\hat{i}}]^2 \right\}^{\frac{1}{2}} \tag{2.4}$$

However, it is not necessary to restrict the definition of closeness to the case of a Euclidean distance [64]. Especially when the objectives are in different units, minimizing $N(x)$ is biased toward reducing the objective function that is on the greatest order of magnitude. Thus, the Euclidean distance alone is insufficient in many MO problems. To accurately measure the closeness to the Utopia point, the function values are often transformed to be non-dimensional.

2.1.1 Function transformations

Here, some common function transformation methods are presented. The first approach is given as follows [54]:

$$f_{\hat{i}}^{\text{trans}} = \frac{f_{\hat{i}}(x)}{f_{\hat{i}}^{\max}} \tag{2.5}$$

which results in a non-dimensional objective function with an upper limit of one (or negative one) and an unbounded lower limit assuming that $f_{\hat{i}}^{\max} \neq 0$.

There are two approaches for determining $f_{\hat{i}}^{\max}$. One can define $f_{\hat{i}}^{\max}$ such that $f_{\hat{i}}^{\max} = \max\limits_{\hat{1} \leq \hat{j} \leq \hat{k}} f_{\hat{i}}(x_{\hat{j}}^{*})$, where $(x_{\hat{j}}^{*})$ minimizes $\hat{j}^{th}$ objective function. Subsequently, $x_{\hat{j}}^{*}$ is a vertex of the Pareto optimal set in the design space whereas $f_{\hat{i}}(x_{\hat{j}}^{*})$ is a vertex of the Pareto optimal set in the objective space. An alternative to Eqn. 2.5 is given as follow [50; 56; 65]:

$$f_{\hat{i}}^{\text{trans}} = \frac{f_{\hat{i}}(x) - f_{\hat{i}}^{*}}{f_{\hat{i}}^{*}} \tag{2.6}$$

This approach also provides a non-dimensional objective function. However, in this case, the lower value of $f_{\hat{i}}^{\text{trans}}$ is restricted to zero, while the upper value is unbounded. Eqn. 2.6 is often referred to as the *relative deviation* or *fractional deviation.* Computational difficulties can arise not only if the denominator is zero but also if it is negative. Consequently, one assumes that the denominator is positive, or uses its absolute value. A variation of Eqn. 2.6 is proposed by Koski and Silvennoinen [35] and Chen et al [11]:

$$f_{\hat{i}}^{\text{trans}} = \frac{f_{\hat{i}}(x)}{f_{\hat{i}}^{*}}, \quad f_{\hat{i}}^{*} > 0 \tag{2.7}$$

This approach yields non-dimensional objective function values with a lower limit of one. The most robust approach to transforming objective functions, regardless of their original range, is given as follows [34; 54]:

$$f_{\hat{i}}^{\text{trans}} = \frac{f_{\hat{i}}(x) - f_{\hat{i}}^{*}}{f_{\hat{i}}^{\max} - f_{\hat{i}}^{*}}. \tag{2.8}$$

This approach is commonly referred to as *normalization.* In this case, $f_{\hat{i}}^{\text{trans}}$ generally has values between zero and one, depending on the accuracy and method with which $f_{\hat{i}}^{\max}$ and $f_{\hat{i}}^{*}$ are determined.

2.2 Weighted methods

The methods presented in this section allow the user to specify preferences, which may be articulated in terms of goals or the relative importance of different objectives. Most of these methods incorporate *parameters*, which are coefficient, exponents, constraint limits, etc. that can either be set to reflect preferences, or be continuously altered to represent the complete Pareto optimal set.

2.2.1 Weighted global criterion method

One of the most common general scalarization methods for multi-objective optimization is the *global criterion method* in which all objective functions are combined to form a single utility function (f_{mo}). In this way, multi-objective problems can be solved with many known single objective optimization algorithms. One of the most general utility functions is expressed in its simplest form as the *weighted exponential sum*:

$$f_{mo} = \textstyle\sum_{\hat{i}=\hat{1}}^{\hat{k}} w_{\hat{i}} \, [f_{\hat{i}}(x)]^p, \qquad f_{\hat{i}}(x) > 0 \quad \forall \hat{i}, \tag{2.9a}$$

$$f_{mo} = \textstyle\sum_{\hat{i}=\hat{1}}^{\hat{k}} [w_{\hat{i}} \, f_{\hat{i}}(x)]^p, \qquad f_{\hat{i}}(x) > 0 \quad \forall \hat{i} \tag{2.9b}$$

The most common extensions of Eqn. 2.9 [71; 72; 10] are

$$f_{mo} = \left\{ \textstyle\sum_{\hat{i}=\hat{1}}^{\hat{k}} w_{\hat{i}} \, [f_{\hat{i}}(x) - f_{\hat{i}}^*]^p \right\}^{\frac{1}{p}}, \tag{2.10a}$$

$$f_{mo} = \left\{ \textstyle\sum_{\hat{i}=\hat{1}}^{\hat{k}} w_{\hat{i}}^p \, [f_{\hat{i}}(x) - f_{\hat{i}}^*]^p \right\}^{\frac{1}{p}}. \tag{2.10b}$$

where $\mathbf{w} = [w_{\hat{1}}, w_{\hat{1}}, \cdots, w_{\hat{k}}]$ is a vector of weights typically set by the decision-maker such that $\sum_{\hat{i}=\hat{1}}^{\hat{k}} w_{\hat{i}} = 1$ and $w_{\hat{i}} > 0 \; \forall \; \hat{i}$. As with most methods that involve objective function weights, setting one or more of the weights to zero can result in weak Pareto optimality where Pareto optimality may be achievable. In general, the value of weights reflects the relative importance of the objectives.

2.2.2 Weighted sum method

The most common approach to multi-objective optimization is the weighted sum method:

$$f_{mo} = \sum_{\hat{i}=1}^{\hat{k}} w_{\hat{i}} f_{\hat{i}}(x). \tag{2.11}$$

This is a form of Eqn. 2.9 with $p = 1$. If all of the weights are positive, the minimum of Eqn. 2.11 is sufficient for Pareto optimality. However, the formulation does not provide a necessary condition for Pareto optimality [73].

The processing of selecting non-arbitrary weights can be inefficient. Consequently, many methods have been developed to systematically select weights, and survey of such methods are provided by Eckenrode [17], Hobbs [29], and Hwang and Yoon [31]. A satisfactory selection of weights can be difficult, and it does not guarantee that the final solution will be acceptable. Varying the weights does not consistently necessarily result in an even distribution of Pareto optimal points and an accurate, complete representation of the Pareto optimal set [46]. Thus, an adaptive weights varying method has been proposed to mitigate the deficiency. The most glaring weakness of the weighted sum approach is that it is impossible to obtain points on non-convex portions of the Pareto optimal set. Das and Dennis [13] and Messac et al [45] give theoretical reasons for this deficiency.

2.2.3 Exponential weighted criterion

In response to the inability to capture points on non-convex portions of the Pareto optimal surface in the weighted sum method, Athan and Papalambros [4] propose the *exponential weighted criterion*, as follows:

$$f_{mo} = \sum_{\hat{i}=\hat{1}}^{\hat{k}} (e^{pw_{\hat{i}}} - 1)e^{pf_{\hat{i}}(x)}, \tag{2.12}$$

Although large values of p can lead to numerical overflow, minimizing Eqn. 2.12 provides a necessary and sufficient condition for Pareto optimality.

2.2.4 Weighted product method

This approach is used for functions with different orders of magnitude to have similar significance while avoiding having to transform the objective functions [7; 21]:

$$f_{mo} = \prod_{\hat{i}=\hat{1}}^{\hat{k}} [f_{\hat{i}}]^{w_{\hat{i}}}, \tag{2.13}$$

where $w_{\hat{i}}$ is the weights assigning the relative importance of the objective functions. However, this approach has not been used extensively, and the characteristic of the weights are unclear. The low adaptation could be the result of potential nonlinearities and consequent computational difficulties [41].

2.3 Non-weighted method

Often the decision-maker cannot concretely define the relative importance of the objectives. This section describes methods that do not require any *weights*, which represent the decision maker's preferences. The fundamental idea behind most of non-weighted methods is the use of an *exponential sum* or *objective products*. These methods form a single objective function f_{mo}.

2.3.1 Non-weighted global criterion method

Non-weighted global criterion method is analogous to its respective weighted method counterpart. This section presents three types of different non-weighted global criterion method. The *Non-weighted objective sum method* is equivalent with optimizing Eqn. 2.11 with $p = 1$ and $w_{\hat{i}} = 1 \; \forall \; \hat{i}$. In fact, this is a special case of weighted sum method. The general principle to find a Pareto optimal point using the weighted sum method is discussed in depth in the previous section. The *Non-weighted min-max method* is derived by excluding weights in Eqn. 2.11, and using $p = \inf$. Then, Eqn. 2.9 yields an L_{inf}-norm, which does not necessarily yield a Pareto optimal point. *Non-weighted distance to Utopia point method* measures how similar the solution is to the ideal solution without assigning weights. One notable example of such method is 'the technique for order preference by similarity to ideal solution' (TOPSIS) [37]. It seeks not only a point that is as close as possible to the utopia point but also a point that is as far away as possible from the worst case possible. The utopia point is the *positive ideal* solution, and the worst solution is the *negative ideal* solution.

2.4 Multi-objective genetic algorithm

A Genetic Algorithm (GA) is a heuristic algorithm that resembles evolutionary processes in biology. In each iteration (generation), multiple candidates (population) are evaluated. The best performing candidates (elites) proceed to the next iteration along with newly-generate candidates that are similar to elites. Multi-objective GA was inspired by the fact that GA evaluates multiple candidates at any given time. By adding the closeness as a measure of fit among the elites, multi-objective GA [14] aims to find a set of solutions that are Pareto optimal to one another when analysis is completed.

2.5 Discussion

The utility of MO formulation depends on the characteristics of each MO problem, and there is no single formulation that works well for every MO problem. In practice, various concepts of MO are combined to define a MO objective function. Defining a proper MO objective function is not a trivial task, and it requires a sound mathematic decision. In this work, the MO function is constructed by combining three elements: distance to Utopia point (2.4), function normalization (2.8), and the weighted sum (2.11). The full description of the MO formulation is described in Chapter 6.

CHAPTER III

Multi-level optimization

Multi-level optimization has been developed to organize complex Multidisciplinary Design Optimization (MDO). MDO synthesizes a design by creating a mathematical framework that encompasses multiple aspects of an engineering system. MDO has emerged as an important technical area that focuses on optimization strategies for complex systems.

To ease optimization effort, 'distributed' MDO algorithms are developed to decompose an engineering design problem into multiple sub-problems. The decomposition is inspired by industry practices in which the design of a system is determined by multiple engineering groups. For instance, ship design is synthesized by collaboration of multiple engineering groups such as hydrodynamics, structure, survivability, etc. In distributed MDO algorithms, each engineering group controls its own design procedures and uses in-house expertise to solve its engineering problems. In this way, engineering design can be distributed across multiple engineering groups [44]. However, when a MDO problem is decomposed to multiple sub-problems, computation sequence and information flow must be carefully coordinated to synthesize the final design. One of the most common ways to organize the MDO problem is to emulate the hierarchical structure of the decision making processes. The final design of an engineering system requires a synthesis of works from multiple departments; a central authority (upper management) usually facilitates such synthesis by examining various trade-offs. Multi-level optimization deals with the MDO decomposition method, computational sequences, and proper treatment of global design variables and constraints in a hierarchical design process.

This chapter contains a brief summary of known multi-level optimization algorithms: Collaborative Optimization (CO) [55], Concurrent Subspace Optimization (CSSO) [57], Analytical Target Cascading (ATC) [33], and BLISS [58]. A general MDO problem definition and notation is introduced first, then the various multi-level algorithms are described.

3.1 Multi-level optimization and notation

In general, multi-level optimization has a single objective function (f), which is often called *system level* objective function. Here, we assume that the problem is decomposed to $\hat{k}$ disciplines. Global design variables ($x_{\hat{0}}$) is a subset of design variables x which appears at more than one disciplines. Local design variables ($x_{\hat{i}}$) is a subset of design variables x which appears at only one discipline ($\hat{i}$). The coupling variables (y) are a set of variables that must be computed within discipline analyses. The coupling variables are not design variables, but it can be seen as design parameters, which are functions of design variables, $y(x_{\hat{0}}, x_{\hat{1}}, ..., x_{\hat{k}})$. For instance, the center of gravity of a ship is not a design variable; however, it is an important parameter that is used in other engineering disciplines, such as ship motion prediction and structural load analysis. In the final ship configuration, the center of the gravity must agree across all engineering disciplines.

Different multi-level optimization algorithms handle global design variables, local design variables, and coupling variables differently, and the algorithms have different strategies to achieve consistency of the global design variables and the coupling variable across multiple disciplines. A general multi-level mathematic optimization is defined as follows:

$$\min_{x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}} f(x_{\hat{0}}, x_{\hat{1}}, \cdots, x_{\hat{k}}, y) \tag{3.1a}$$

subject to

$$c_{\hat{0}}(x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}, y) \leq 0 \tag{3.1b}$$

$$c_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}}, y) \leq 0 \quad \forall \quad \hat{i} = \hat{0}, \cdots, \hat{k} \tag{3.1c}$$

3.2 Collaborative optimization

Collaborative optimization (CO) [55] is a multi-level method which allows the disciplines to perform independent optimization problems. The collaborative optimization includes a system level optimization statement with compatibility constraints to ensure that $x_{\hat{0}}$ are consistent across multiple disciplines. The top level optimization statement is defined as follows:

$$\min_{x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}} f(x_{\hat{0}}, x_{\hat{1}}, \cdots, x_{\hat{k}}) \tag{3.2a}$$

subject to

$$J^*_{\hat{i}} = 0 \quad \forall \quad \hat{i} = \hat{0}, \cdots, \hat{k} \tag{3.2b}$$

$J_{\hat{i}}$ represents the interdisciplinary compatibility of discipline $\hat{i}$, and $J^*_{\hat{i}}$ is the solution to

$$\min_{x_{\hat{0}},x_{\hat{1}},\cdots x_{\hat{k}}} J_{\hat{i}} = \sum_{\hat{i}=0}^{k}(x_{\hat{i}} - x_{\hat{i}}^t)^2 + \sum_{\hat{i}=0}^{k}(y_{\hat{i}} - y_{\hat{i}}^t)^2 + \sum_{\hat{j}\neq\hat{i}}^{k}(y_{\hat{j},\hat{i}} - y_{\hat{j},\hat{i}}^t)^2 \tag{3.3a}$$

subject to

$$J_{\hat{i}}^* = 0 \quad \forall \quad i = 1, \cdots, k \tag{3.3b}$$

$$c(x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}, y_{\hat{i}}(x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}, y_{\hat{j},\hat{i}})) \leq 0 \tag{3.3c}$$

CO utilizes target values (denoted with superscript t) for the design variables that are used in the system. CO attempts to achieve consistency of design variables and coupling variables across the multiple disciplines by minimizing discrepancy between the target values and the actual values. Thus, a solution to the Eqn. 3.3 ($J_{\hat{i}}^* = 0$) implies that local design variables and coupling variables are consistent across all disciplines.

The use of target values allows the disciplines to optimize more independently which resembles real engineering development processes in a distributed environment. However, solutions to the compatibility constraint (J^*) is often difficult to find. When the compatibility constraint is not found, the disciplines will not agree on the *global* properties. Thus, CO often leads to non-convergent solutions, and it is also known to have comparatively high computational cost to other known methods.

3.3 Concurrent subspace optimization

Unlike CO, which requires a solution to compatibility equation, concurrent subspace optimization (CSSO) [57] uses an approximation for the coupling variables ($\tilde{y}$). Using the approximation for the coupling variables ($\tilde{y}$) can accelerate the system optimization significantly. However, the approximation model must be constructed prior to the optimization. The construction of such model can be computationally expensive. The system level optimization problem is defined as follows:

$$\min_{x_{\hat{0}},x_{\hat{1}},\cdots x_{\hat{k}}} f(x_{\hat{0}}, x_{\hat{1}}, \cdots, x_{\hat{k}}, \tilde{y}) \tag{3.4a}$$

subject to

$$c_{\hat{0}}(x_{\hat{0}}, x_{\hat{1}}, \cdots, x_{\hat{k}}, \tilde{y}) \leq 0 \tag{3.4b}$$

where $\tilde{y}$ is the approximation for the coupling variables y. The discipline optimization problem for discipline $\hat{\imath}$ is defined as follows:

$$\min_{x_{\hat{0}},x_{\hat{\imath}}} f(x_{\hat{0}}, x_{\hat{\imath}}, y_{\hat{\imath}}(x_{\hat{0}}, x_{\hat{\imath}}, \tilde{y}_{\hat{\jmath}}), \tilde{y}_{\hat{\jmath}}) \tag{3.5a}$$

subject to

$$c_{\hat{\imath}}(x_{\hat{0}}, x_{\hat{\imath}}, y_{\hat{\imath}}(x_{\hat{0}}, x_{\hat{\imath}}, \tilde{y}_{\hat{\jmath}}), \tilde{y}_{\hat{\jmath}}) \leq 0 \tag{3.5b}$$

where $x_{\hat{0}}$ is held constant and $\tilde{y}_{\hat{\jmath}}$ is the approximation for the coupling variables from the other disciplines. The convergence of y and $\tilde{y}$ is not always guaranteed, and the performance of the overall optimization is sensitive to the fidelity of the approximation model [57].

3.4 Analytical target cascading

Analytical target cascading (ATC) [33] is a multi-level hierarchical method that is designed to propagate target values across the multiple levels. ATC minimizes the discrepancy between the target value ($\mathbf{T}$) and the response values from the system analysis (**R**). The system level objective function is defined as follows:

$$\min_{x\in\mathbb{R}} ||\mathbf{T} - \mathbf{R}(x)|| \tag{3.6a}$$

$$\text{subject to}$$

$$c(x) \leq 0 \tag{3.6b}$$

Similar to CO, the target values are used.

The formulation of ATC includes optimizations in three levels: supersystem, system, and subsystem (disciplines). The supersystem (subscript "sup") level optimization minimizes the different between the supersystem level response $\mathbf{R}_{sup}$ and thetarget values $\mathbf{T}_{sup}$. The system level constraints include the system design constraints (c_{sup}), and the tolerance constraints which include the deviation tolerance $\epsilon_{\mathbf{R}}$ and ϵ_y. The deviation tolerances are supersystem-level design variables that coordinate the responses and the coupling variables; when the optimization had converged the deviation tolerances should reach zero to achieve consistency. The supersystem optimization statement is defined as follows:

$$\min_{x,y,\epsilon_{\mathbf{R}},\epsilon_y} ||\mathbf{T}_{sup} - \mathbf{R}_{sup}(x)|| + \epsilon_{\mathbf{R}} + \epsilon_y \tag{3.7a}$$

$$\text{subject to}$$

$$\sum_{\hat{i}=1}^{\hat{k}} ||\mathbf{R}_{s,\hat{i}} - \mathbf{R}^L_{s,\hat{i}}|| \leq \epsilon_{\mathbf{R}} \tag{3.7b}$$

$$c_{sup}(x_{sup}) \leq 0 \tag{3.7c}$$

where the superscript 'L' indicates target values determined from the system level optimization. The subscript 's' indicates elements evaluated at the system optimization.

The system level optimization coordinates the supersystem and subsystem (discipline) optimizations. The system level optimization problem is defined as follows:

$$\min_{x_s, y_s, y_{ss}, \epsilon_{\mathbf{R}}, \epsilon_y} ||\mathbf{R}_s - \mathbf{R}_s^U|| + ||y_s - y_s^U|| + \epsilon_{\mathbf{R}} + \epsilon_y \tag{3.8a}$$

subject to

$$\sum_{\hat{i}=1}^{\hat{k}} ||\mathbf{R}_{ss,\hat{i}} - \mathbf{R}_{ss,\hat{i}}^L|| \leq \epsilon_{\mathbf{R}} \tag{3.8b}$$

$$\sum_{\hat{i}=1}^{\hat{k}} ||y_{ss,\hat{i}} - y_{ss,\hat{i}}^L|| \leq \epsilon_y \tag{3.8c}$$

$$c_s(x_s, y_s) \leq 0 \tag{3.8d}$$

where the subscript 'ss' indicates the subsystem level and the superscript 'U' indicates the targets from the supersystem level.

The subsystem optimization minimizes the difference between the values of responses and linking variables at the subsystem level and system target values. The subsystem optimization statement is defined as follows:

$$\min_{x_{ss}, y_{ss}} ||\mathbf{R}_{ss} - \mathbf{R}_{ss}^U|| + ||y_{ss} - y_{ss}^U|| \tag{3.9a}$$

subject to

$$c_{ss}(x_{ss}, y_{ss}) \leq 0 \tag{3.9b}$$

In ATC, targets values are propagated to multiple levels. The capability to handle multiple objective functions with target values and the use of deviation tolerances (ϵ) greatly enhances the degree of freedom at each level. However, achieving consistency is shown to be difficult in some application using ATC [33].

3.5 Bi-level integrated system synthesis

Bi-level integrated system synthesis (BLISS) [58] is a single objective multi-level method with a system level optimization and multiple discipline optimizations. The algorithm optimizes the system level objective function (f), and the optimization problem is decomposed to two levels (bi-level). The system level optimization optimizes its objective function with respect to *global design variables* ($x_{\hat{0}}$).

$$\min_{\Delta x_{\hat{0}}} f(x_{\hat{0}}, x_{\hat{1}}, \cdots, x_{\hat{k}}) = f_0 + \frac{df}{dx_{\hat{0}}} \Delta x_{\hat{0}} \tag{3.10a}$$

subject to

$$lb_{x_{\hat{0}}} \leq x_{\hat{0}} + \Delta x_{\hat{0}} \leq ub_{x_{\hat{0}}} \tag{3.10b}$$

$$lb_{\Delta x_{\hat{0}}} \leq \Delta x_{\hat{0}} \leq ub_{\Delta x_{\hat{0}}} \tag{3.10c}$$

where Δ indicates an incremental step and lb and ub indicate the lower and upper bounds of the design variables.

On the other hand, each discipline optimization optimizes with respect to *local design variable*, ($x_{\hat{i}}$) while treating the global design variable ($x_{\hat{0}}$) constant. The discipline level optimization does not optimize the objective function (f), but it optimizes a function that measures the influence of $x_{\hat{i}}$ on the system level objective function. The discipline objective function is a weighted sum of the local design variable ($x_{\hat{i}}$) weighted by an approximation for the derivative of f with respect to $x_{\hat{i}}$. The discipline optimization statement is defined as follows:

$$\min_{\Delta x_{\hat{i}}} \left(\frac{df}{d\Delta x_{\hat{i}}} \right)^{\top} \Delta x_{\hat{i}} \tag{3.11a}$$

subject to

$$c_{\hat{i}}(x_{\hat{i}} + \Delta x_{\hat{i}}, x_{\hat{0}}, y) \leq 0 \tag{3.11b}$$

where the discipline $\hat{i}$ is assumed to have access to the coupling variables (y) from all disciplines. If the coupling variables are not accessible, response functions or estimate models of y are required.

CHAPTER IV

Conjugate Gradient method

The Conjugate Gradient (CG) method is a class of numerical optimization algorithms. The CG method was proposed four decades ago [28], and many variants of CG methods have been developed since. The properties and convergence of CG methods have been well-documented. Because of its robust theoretical justification and historical successes, the CG method is selected as the foundations of the research in this areaThis chapter provides in-depth discussion of the CG method [2; 26].

THe CG method encompasses many fundamental concepts of the numerical optimization. Thus, the basics of the numerical optimization are reviewed first. Then, the detail description of CG method is presented. Much of this chapter is a brief summary of an excellent numerical optimization textbook written by Nocedal and Wright [49].

4.1 Introduction to numerical optimization

Optimization problems appear often in engineering design therefore, finding optimal solutions to those problems is extremely valuable. Many of these optimization problems are too difficult to solve analytically. Thus, iterative numerical optimization methods are used to find an optimal solution. This chapter explains basic concepts of numerical optimization.

4.2 Mathematical formulation

The standard mathematic notations are defined as follows:

- x is the vector of *design variables*;
- $f(x)$ is the *objective function*, a scalar function of x;
- $c(x)$ are the *constraint* functions, which are scalar functions of x that define certain equations and inequalities that x must satisfy.

The optimization problem can be written as follows:

$$\min_{x \in \mathbb{R}^n} \; f(x) \tag{4.1a}$$

$$\text{subject to}$$

$$ceq_{(i)}(x) = 0 \quad i \in \varepsilon \tag{4.1b}$$

$$c_{(i)}(x) \leq 0 \quad i \in I \tag{4.1c}$$

where I and ε are sets of indices for equality and inequality constraints respectively. Without loss of generality, only *inequality constraints* (4.1c) are considered for the remainder of the thesis because *equality constraints* (4.1b) are identical to two identical inequality constraints with opposite signs.

4.3 Fundamentals of constrained optimization

For *unconstrained* optimization problem (i.e., $c(x) = \emptyset$), the optimality conditions are

- **Necessary conditions**: A local optimum, x^* has $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \geq 0$

- **Sufficient conditions**: Any point, x^* at which $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \geq 0$. is a strong local optimum

Analogous conditions for a *constrained* optimization problem is called Karush-Kuhn-Tucker (*KKT*) condition. KKT condition is a set of mathematic equations that the solution (x^*) must satisfy when the objective function value, $f(x)$, cannot improve along every feasible direction in the neighborhood of x. It is necessary to introduce *Lagrange* function of constrained optimization problem as KKT condition is expressed as a function of Lagrange function. The Lagrange function of a general constrained optimization problem is defined as follows:

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i \in I} \lambda_{(i)} c_{(i)}(x). \tag{4.2a}$$

where $\lambda_{(i)}$ is a Lagrange multiplier for a corresponding constraint function $c_{(i)}$. Then, KKT condition is defined as follows:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0 && (4.3a) \\ c_{(i)}(x^*) &\leq 0 \quad \forall\, i \in I, && (4.3b) \\ \lambda^*_{(i)} &\geq 0 \quad \forall\, i \in I, && (4.3c) \\ \lambda^*_{(i)}\, c_{(i)}(x^*) &= 0 \quad \forall\, i \in I. && (4.3d) \end{aligned}$$

KKT condition is the necessary and sufficient first order condition for an optimum solution of $f(x)$. Each Lagrange multiplier $\lambda_{(i)}$ indicates the *sensitivity* of the optimal objective value $f(x^*)$ to the presence of the constraints $c_{(i)}$. Many numerical optimization algorithms for constrained problems find a solution (x^*, λ^*) to Eqn. 4.3, and there are two main branches of gradient-based numerical optimizations that find solution of Eqn. 4.3. The 'line-search' methods finds a search direction and an optimal step iteratively in sequence. On the other hand, 'trust-region' methods find a search direction and an optimal step simultaneously within a region where the quadratic model of the objective function can be trusted.

4.4 Line-search methods

The line-search methods start with an initial guess x_0, then the optimizer updates the value of x until converge criteria are satisfied. In one iteration of a line-search method, the algorithm decides a search direction (d_k) first. Then, it decides how far to move along the direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k \, d_k, \tag{4.4}$$

where x_k refers to current k^{th} iteration of design variable x, and the positive scalar α_k is called the *step length*. The effectiveness of line-search algorithms are determined by effective choices of both the direction (d_k) and the step length (α_k).

Most line search algorithms required d_k to be a *decent direction*; the function value must decrease along the search direction ($d_k^\top \nabla f_k < 0$). In most line-search algorithms, the search direction has a form of

$$d_k = -B_k^{-1} \nabla f_k, \tag{4.5}$$

where B_k is a symmetric and nonsingular matrix. Different choices of B_k corresponds to different line-search methods: steepest decent, Newton's method, quasi-Newton's method, and CG method. As mentioned, line-search algorithms have two major components: step length selection method and search direction computation method.

4.5 Selection of step length

The performance of step length (α_k) selection methods is critically important to the overall performance of line-search algorithm. It is impossible to have a well-performing line-search algorithm with poor step length selection algorithms. Finding an optimal step length requires solving a minimization problem as shown in Eqn. 4.6.

$$\min_{\alpha>0} \phi(\alpha) = f(x_k + \alpha d_k) \tag{4.6}$$

where the step length is strictly greater than zero ($\alpha > 0$).

4.5.1 Step length selection convergence condition

There exists many convergence conditions for one dimension step length search algorithm. Two popular convergence conditions, the Wolfe condition and the Goldstein condition, are presented in this section. The *Wolfe conditions* [67; 68] is consist of two mathematic inequalities, The first inequality ensures that the choice of α_k should sufficiently reduce the function value.

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^{\top} d_k \tag{4.7}$$

for some constant $c_1 \in (0, 1)$. In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f_k^{\top} d_k$. Eqn. 4.7 is called *sufficient decrease* condition or often called *Armijo condition.*

The second inequality ensures that the step length is reasonably big. In practice, the sufficient decrease condition alone cannot ensure that the algorithm is making reasonable progress because all α that is sufficiently small satisfy Eqn. 4.7. To remove unacceptably small step lengths, the second inequality, *curvature condition*, is introduced.

$$\nabla f(x_k + \alpha_k d_k)^\top d_k \geq c_2 \nabla f_k^\top d_k \tag{4.8}$$

for some constant $c_2 \in (c_1, 1)$, where c_1 is the constant from Eqn. 4.7. Note that left-hand-side of Eqn. 4.8 is equal to the derivative $(\phi'(\alpha_k))$, so the curvature condition ensures that the slope of ϕ at α_k is greater than c_2 times the initial slope $\phi'(0)$. In this way, the algorithm rejects α that is too small.

Combining both Eqn. 4.7 and Eqn. 4.8, we have Wolfe condition

$$f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha \nabla f_k^\top d_k, \tag{4.9a}$$

$$\nabla f(x_k + \alpha_k d_k)^\top d_k \geq c_2 \nabla f_k^\top d_k, \tag{4.9b}$$

Like the Wolfe conditions, the *Goldstein conditions* ensure that the step length α achieves sufficient decrease but it not too small. The Goldstein conditions can also be stated as a pair of inequalities as shown in Eqn. 4.10

$$f(x_k) + (1-c)\alpha_k \nabla f_k^\top d_k \leq f(x_k + \alpha_k d_k) \leq f(x_k) + c\alpha_k \nabla f_k^\top d_k, \tag{4.10}$$

with $0 < c < 1/2$.

A disadvantage of the Goldstein conditions is that the first inequality may exclude all minimizers of ϕ. However, the Goldstein and Wolfe conditions have much in common. The Goldstein conditions are often used in Newton-type methods, but they are not well suited for quasi-Newton methods that maintain a positive definite Hessian approximations.

Recently, Hager and Zhang [26] proposed a new convergence conditions called *approximate Wolfe conditions.*

$$\alpha g_k^\top d_k \le g_{k+1}^\top d_k \le (2\delta - 1) g_k^\top d_k, \tag{4.11}$$

where $0 < \delta < 1/2$ and $\delta < \alpha < 1$. The approximate Wolfe conditions are used in efficient, high accuracy implementations of step length selection algorithms. Although there is no convergence theory for the approximate Wolfe conditions, the performance is often much better in practice than that of the Wolfe condition.

4.5.2 Step length selection algorithms

One dimensional search algorithm determines the optimal step length that satisfies the convergence conditions, such as Eqn. 4.9 and Eqn. 4.10. *Backtracking* method starts with an initial guess α_0, which is sufficiently large, then it reduces the step length by a contraction factor ($0 < \rho < 1$) until the sufficient decrease condition (4.7) is satisfied. Backtracking method starts with a sufficiently large initial guess α_0 to remove the *curvature* condition (4.8), which ensures a sufficiently large step length of α_k. Once an acceptable step length is found, the algorithm is terminated [22]. In Newton and quasi-Newton methods, the initial step length α_0 is typically set to 1 , but other methods can take different value. In practice, the contraction factor (ρ) is often allowed to vary at each iteration of the line search.

Algorithm 1 Backtracking Line Search

```
Choose $\alpha_0 > 0$, $\rho \in (0,1)$, $C \in (0,1)$; Set $\alpha \leftarrow \bar{\alpha}$;
while $f(x_k + \alpha d_k) \le f(x_k) + C\alpha \nabla f_k^\top d_k$ do
    $\alpha \leftarrow \rho\, \alpha$;
end while
return $\alpha_k = \alpha$
```

The *bi-section search* method is a one-dimension search method that searches a suitable step length (α^*) within a search space $\alpha^* \in [0, \alpha_{max}]$. Bi-section starts with an initial interval $[0, \alpha_{max}]$ and divide the interval into two halves, (i.e. $[0, \frac{\alpha_{max}}{2}]$ and

$[\frac{\alpha_{max}}{2}, \alpha_{max}]$). By comparing function value at extreme points of each interval, bi-section search method selects one interval that is more likely to contain the optimal value. The halving of the search space continues iteratively until a step length that satisfies convergence conditions is found. The bi-section search is guaranteed to converge, but it converges slowly. The *golden section search* method is similar to the bi-section search method. It starts with an initial interval $\alpha^* \in [0, \alpha_{max}]$, and it uses the positive golden ratio to reduce the search interval.

4.6 Steepest decent

As mentioned above, most line-search algorithms compute a search direction in a form of Eqn. 4.12

$$d_k = -B_k^{-1} \nabla f_k, \tag{4.12}$$

In the steepest descent method, B_k is the identity matrix I. Thus, the search direction d_k is identical to $-\nabla f(x_k)$. Although this is an intuitive choice of a search direction, it exhibits poor rate of convergence due to *zigzaging* behaviors near the optimum. Consider when the objective function is quadratic and when the line searches are exact. Suppose that

$$f(x) = \frac{1}{2} x^\top Q x - b^\top x, \tag{4.13}$$

where Q is symmetric and positive definite. The gradient is given by $\nabla f(x) = Qx - b$ and the minimizer x^* is the unique solution of the linear system $Qx = b$.

The step length α_k that minimizes $f(x_k - \alpha \nabla f_k)$ can be computed by setting the derivative, with respect to α_k, to zero.

$$f(x_k - \alpha \nabla f_k) = \frac{1}{2}(x_k - \alpha \nabla f_k)^\top Q (x_k - \alpha \nabla f_k) - b^\top (x_k - \alpha \nabla f_k) \tag{4.14}$$

By setting the derivative to zero,

$$\alpha_k = \frac{\nabla f_k^\top \nabla f_k}{\nabla f_k^\top Q \nabla f_k} \tag{4.15}$$

Subsequently,

$$x_{k+1} = x_k - \left(\frac{\nabla f_k^\top \nabla f_k}{\nabla f_k^\top Q \nabla f_k} \right) \nabla f_k \tag{4.16}$$

To quantify the rate of convergence, introduce the weighted norm $||x||_Q^2 = x^\top Q x$, then

$$\frac{1}{2}||x - x^*||_Q^2 = f(x) - f(x^*), \tag{4.17}$$

Theorem IV.1. *When the steepest descent method with exact line search (4.15) is applied to the strongly convex quadratic function, the error norm satisfies*

$$||x_{k+1} - x^*||_Q^2 \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right)^2 ||x_k - x^*||_Q^2, \tag{4.18}$$

where $0 < \lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$ *are the eigenvalues of* Q

The proof is given by Luenberger [40]. In general, as the condition number $\kappa(Q) = \lambda_n/\lambda_1$ increases, the zigzag behavior becomes more pronounced in the steepest decent method. Eqn. 4.18 is a worst-case bound, but it gives an accurate indication of the behavior of the algorithm when $n > 2$.

Theorem IV.2. *Suppose that* $f : \mathbb{R}^n \rightarrow \mathbb{R}$ *is twice continuously differentiable, and that the iterates generated by the steepest-descent method with exact line searches converges to a point* x^* *at which the Hessian matrix* $\nabla^2 f(x^*)$ *is positive definite. Let* r *be any scalar satisfying*

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1 \right) \tag{4.19}$$

where $0 < \lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$ *are the eigenvalues of* $\nabla^2 f(x^*)$*. Then for all* k *sufficiently* *large,*

$$f(x_{k+1}) - f(x^*) \leq r^2[f(x_k) - f(x^*)] \tag{4.20}$$

Theorem IV.2 shows that the steepest method can have an unacceptably slow rate of convergence even if the Hessian is reasonably well conditioned. In practice, the steepest descent method is not popular because of its slow rate of convergence.

4.7 Newton's method

In the Newton's method, B_k is the exact Hessian $\nabla^2 f(x_k)$.

$$B_k = \nabla^2 f(x_k), \quad d_k = \frac{\nabla f_k}{-\nabla^2 f_k}. \tag{4.21}$$

Since the Hessian matrix $\nabla^2 f_k$ may not always be positive definite, d_k may not always be a descent direction. To find a global minimum, modifications to Newton's method search direction is required. There are various modification techniques. In general, the modified Hessian is obtained by adding either a positive diagonal matrix or a full matrix to the true Hessian ($\nabla^2 f(x_k)$).

$$B_k = \nabla^2 f(x_k) + E_k \tag{4.22}$$

where $E_k = 0$ if $\nabla^2 f(x_k)$ is sufficiently positive definite; Otherwise, E_k is chosen to ensure that B_k is sufficiently positive definite.

Theorem IV.3. *Suppose that f is twice differentiable and that the Hessian $\nabla^2 f(x)$ is Lipschitz continuous in a neighborhood of a solution x^* at which the sufficient conditions are satisfied. Consider the iterations $x_{k+1} = x_k + d_k$, where d_k is given by (4.21). Then*

1. *if the starting point x_0 is sufficiently close to x^*, the sequence of iterates converges to x^*;*

2. *the rate of convergence of $\{x_k\}$ is quadratic; and*

3. *the sequence of gradient norms $\{||\nabla f_k||\}$ converges quadratically to zero.*

A proof of Thm. IV.3 can be found in [49]. The theorem shows that the convergence of the Newton's method is known to be significantly faster than that of the steepest decent method, but computation advantage is hard to achieve when the Hessian matrix of the objective function is not available in explicit equations. Computing the Hessian matrix numerically becomes prohibitively expensive as the number of the variables increases [69]. Thus, the Hessian matrix is often estimated in quasi-Newton's methods.

4.8 Quasi-Newton's method

In quasi-Newton method, B_k is an approximation to the Hessian matrix ($\nabla^2 f(x)$) that is updated at every iteration to reduce the computational load of finding Hessian matrix at every iteration. To simplify the notation, the following terms are defined first.

$$s_k = x_{k+1} - x_k = \alpha_k d_k \tag{4.23}$$

$$y_k = \nabla f_{k+1} - \nabla f_k \tag{4.24}$$

$$\rho_k = \frac{1}{y_k^\top s_k} \tag{4.25}$$

$$H_k = B_k^{-1}, \text{ inverse of } B_k \tag{4.26}$$

4.8.1 DFP method

DFP updating formula is named after three contributors: Davidon, Fletcher and Powell. It was originally proposed by Davidon in 1959, and it was implemented and popularized by Fletcher and Powell [49].

$$B_{k+1} = \left(I - \rho_k y_k s_k^\top\right) B_k \left(I - \rho_k s_k y_k^\top\right) + \rho_k y_k y_k^\top \tag{4.27}$$

$$H_{k+1} = H_k - \frac{H_k y_k y_k^\top H_k}{y_k^\top H_k y_k} + \frac{s_k s_k^\top}{y_k^\top s_k} \tag{4.28}$$

Although DFP updating formula is effective, it was superseded by the BFGS formula, which is considered to be the most effective of all quasi-Newton methods.

4.8.2 BFGS method

BFGS method, named after its discoverers: Broyden, Fletcher, Goldfarb, and Shanno.

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k} \tag{4.29}$$

$$H_{k+1} = (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_k^\top \tag{4.30}$$

In theory, Newton's method converges more rapidly, but its cost per iteration usually higher because the computation of the Hessian matrix. If explicit closed formula of the Hessian matrix is not available, quasi-Newton's method often converges faster in practice.

BFGS method requires the user to choose the initial approximation H_0, but there is no one formula that works well for all problems. Even if the initial Hessian approximation (H_0) is wrong, the Hessian approximation tends to correct itself within a few steps. The DFP method is less effective in correcting bad Hessian approximations, and this property is believed to be the reason for its poorer practical performance.

The self-correcting properties of BFGS hold only if an adequate line search is performed. Please note that DFP and BFGS updating are *duals* of each other, in the sense that one can be obtained from the other by the interchanges $s \leftrightarrow y$, $B \leftrightarrow H$. This insight led to development of a new class of quasi-Newton method called Broyden class.

4.8.3 Broyden class

Broyden class is a family of B_k updating methods specified by the following general formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k} + \phi_k (s_k^\top B_k s_k) v_k v_k^\top, \tag{4.31}$$

where ϕ_k is a scalar parameter and

$$v_k = \left[\frac{y_k}{y_k^\top s_k} - \frac{B_k s_k}{s_k^\top B_k s_k} \right]. \tag{4.32}$$

.

When $\phi_k = 0$, B_{k+1} is identical to B_{k+1} of BFGS method ($B_{k+1} = B_{k+1}^{BFGS}$), and when $\phi_k = 1$, B_{k+1} is identical to B_{k+1} of DFP method ($B_{k+1} = B_{k+1}^{DFP}$) and Eqn. 4.31 can be rewritten as a linear combination of BFGS and DFP.

$$B_{k+1} = (1 - \phi_k) B_{k+1}^{BFGS} + \phi_k \, B_{k+1}^{DFP}. \tag{4.33}$$

Positive definiteness of the Hessian approximations is preserved in DFP and BGFS when $s_k^\top y_k > 0$, and Eqn. 4.33 implies that the same property will hold for the Broyden family if $0 \leq \phi_k \leq 1$. Therefore, a *restricted Broyden class*, which is obtained by restricting ϕ_k to the interval $[0, 1]$, has gained much attention. However, the best update formula does not alway belong to the restricted Broyden class; allowing ϕ_k to be negative has a superior performance to the BFGS method in some cases.

4.9 Trust region methods

Here, key concepts of the trust region methods are presented. Unlike the line-search methods, trust-region methods select the search direction (d_k) and the step length (α_k) simultaneously. Trust-region methods construct a quadratic model of the objective function around the current iterate. Then, it chooses a solution that approxmiates the minimum of the quadratic model within a 'trust' region where the model accurately represents the real objective function. If the solution is not acceptable, the algorithm reduces the size of the region and finds a new minimum.

The size of the trust region is critical to the effectiveness of each step. If the region is too small, then the algorithm misses on more aggressive step that can move the iterate closer to the minimum of the objective function. If too large, the quadratic model might not represent the objective function accurately. In practice, the size of the trust region is determined according to the performance of the algorithm during previous iterations. If the model is reliable to produce good steps and to predict the behavior of the objective function along these steps, the size of the trust region may be increased to allow more ambitious steps. A failed step implies that the model is inadequate to represent the objective function over the current trust region. Subsequently, the size of the 'trust' region may be reduced after a failed step.

The model function (m_k) is the Taylor-series expansion of the objective function (f) around the current iterate (x_k). Moreover, m_k is based on the Taylor-series expansion of f around x_k.

$$f(x_k + p) = f_k + g_k^\top p + \frac{1}{2} p^\top \nabla^2 f(x_k + tp)p, \tag{4.34}$$

where $f_k = f(x_k)$ and $g_k = \nabla f(x_k)$, and t is some scalar in the interval (0,1). By using an approximation B_k to the Hessian in the second-order term, m_k is

$$m_k(p) = f_k + g_k^\top p + \frac{1}{2} p^\top B_k p, \tag{4.35}$$

The solution p_k^* of (4.36) is the minimizer of m_k in the ball of radius Δ_k.

$$\min_{p \in \mathbb{R}^n} m_k(d) = f_k + g_k^\top d + \frac{1}{2} p^\top B_k p \tag{4.36a}$$

such that

$$||P|| \leq \Delta_k \tag{4.36b}$$

where Δ_k is the trust-region radius. $|| \cdot ||$ is the Euclidean norm.

As mentioned above, the size of the trust region affects the performance of the trust-region algorithm. The choice of the trust region radius, Δ_k, is based on the agreement between the model function m_k and the objective function f at previous iterations. Given a step p_k, a ratio between *actual reduction* and *predicted reduction* is

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} = \frac{\textit{actual reduction}}{\textit{predicted reduction}} \tag{4.37}$$

Note that since the step p_k is obtained by minimizing the model m_k over a region that includes $p = 0$, the predicted reduction will always be nonnegative. Hence, if ρ_k is negative, the new objective value $f(x_k + p_k)$ is greater than the current value $f(x_k)$, so the step must be rejected. On the other hand, if ρ_k is close to 1, there is good agreement between the model m_k and the function f over this step, so it is safe to expand the trust region for the next iteration.

4.10 Conjugate Gradient method

Conjugate Gradient (CG) method is a class of algorithms that solves *unconstrained* optimization problems. CG method is one of the most successful numerical optimization methods, and the *linear* conjugate gradient method was proposed by Hestenes and Stiefel in the 1950s [28] as an iterative method for solving linear systems with positive definite coefficient matrices. CG method was expanded to solve *nonlinear* equation by Fletcher and Reeves [20] in the 1960s.

The basic notion of CG method is to search along the *eigenvectors* of the objective function contour. As mentioned above, the steepest decent method exhibits the *zigzagging* behavior, which slows down the rate of convergence. The zigzagging behavior can be avoided by searching along the eigenvectors of the objective function contour. CG method numerically generates the sequence of search directions that are conjugated to one another. Because all other search directions in previous iterates are conjugate to one another, CG method only requires the previous search direction (d_{k-1}) to compute a new search direction (d_k) [49]. This property allows the algorithm to function with low memory; and CG method is one of the earliest known methods for solving large-scale nonlinear optimization problems effectively.

For the remainder of the section, the nonlinear CG method is discussed with greater detail. The nonlinear CG methods solve nonlinear unconstrained optimization problems shown in Eqn. 4.38.

$$\min \left\{ f(x) : x \in \mathbb{R}^n \right\}, \tag{4.38}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, bounded from below. The nonlinear CG method starts with an initial guess $x_0 \in \mathbb{R}^n$, then it generates a sequence of x_k as:

$$x_{k+1} = x_k + \alpha_k d_k \tag{4.39}$$

where the steplength ($\alpha_k > 0$) is obtained by the step length selection algorithms outlined in Section 4.5.2 , and the search direction (d_k) is generated by the following rule:

$$\begin{cases} d_0 = -g_0 \\ d_k = -g_k + \beta_k \, d_{k-1} \end{cases} \tag{4.40}$$

where g_k is the $\nabla f(x_k)$.

Different CG methods correspond to different choices of β_k. Many CG methods are known, and an excellent survey of various CG methods is given by Hagar and Zhang [26]. Table 4.1 summarizes equations to compute different β_k of many known CG methods.

Table 4.1: Summary of β_k update method [26]

Method	Equation	Year
Hestenes and Stiefel	$\beta_k^{HS} = \dfrac{g_{k+1}^\top y_k}{d_k^\top y_k}$	1952
Fletcher-Reeves	$\beta_k^{FR} = \dfrac{\|g_{k+1}\|^2}{\|g_k\|^2}$	1964
Daniel	$\beta_k^{D} = \dfrac{g_{k+1}^\top \nabla^2 f(x_k) d_k}{d_k^\top \nabla^2 f(x_k) d_k}$	1967
Polak-Ribiere-Polyak	$\beta_k^{PRP} = \dfrac{g_{k+1}^\top y_k}{\|g_k\|^2}$	1969
Conjugate-Descent	$\beta_k^{CD} = \dfrac{\|g_{k+1}\|^2}{-d_k^\top g_k}$	1987
Liu-Storey	$\beta_k^{LS} = \dfrac{g_{k+1}^\top y_k}{-d_k^\top g_k}$	1991
Dai-Yuan	$\beta_k^{DY} = \dfrac{\|g_{k+1}\|^2}{d_k^\top y_k}$	1999
Hagar-Zhang	$\beta_k^{N} = \left(y_k - 2 d_k \dfrac{\|y_k\|^2}{d_k^\top y_k} \right) \dfrac{g_{k+1}}{d_k^\top y_k}$	2005

Let $\|\cdot\|$ denote the Euclidean norm, and $y_{k-1} := g_k - g_{k-1}$

All 8 choices for the update parameter in Table 4.1 are equivalent with an exact line search if the objective function (f) is a strongly convex quadratic. For non-quadratic objective functions, each choice of update parameter results in difference performance. Fletcher-Reeves [20], Dai-Yuan [12], and Conjugate-Descent [19] methods have strong convergence properties, but jamming diminishes their performances in practice. Polak-Ribiere-Polyak [52; 53] and Liu-Storey [39] methods have weaker convergence properties, but they often perform better. Recently, various hybrid conjugate gradient methods [3; 38] are proposed to have advantages of both kinds of CG methods. Touati-Ahmed and Store [61] suggested the following hybrid method:

$$\beta_k = \begin{cases} \beta_k^{PRP} & \text{if } 0 \le \beta_k^{PRP} \le \beta_k^{FR} \\ \beta_k^{FR} & \text{otherwise} \end{cases} \tag{4.41}$$

Thus, when the iterations jam, the PRP update parameter is used. By the same motivation, Hu and Storey [30] suggested to take

$$\beta_k = \max\{0, \min\{\beta_k^{PRP}, \beta_k^{FR}\}\}. \tag{4.42}$$

In an effort to extend the allowed choices for the PRP update parameter, while retaining global convergence, Nocedal and Gilbert [23] suggest taking

$$\beta_k = \max\{-\beta_k^{FR}, \min\{\beta_k^{PRP}, \beta_k^{FR}\}\}. \tag{4.43}$$

'CG_DESCENT' method, proposed by Hagar and Zhang [26], is one of the best performing CG methods known to date.

$$\beta_k^{\theta} = \beta_k^{HS} - \theta_k \left(\frac{\|y_k\|^2 g_{k+1}^\top d_k}{(d_k^\top y_k)^2} \right), \tag{4.44}$$

where $\theta_k \ge 0$. A truncated version of 'CG_DESCENT' method is proposed to obtain global convergence for general nonlinear functions.

$$\beta_k^{\theta +} = \max\{\beta_k^N, \eta_k\}, \quad \eta_k = \frac{-1}{\|d_k\| \min\{\eta, \|g_k\|\}}, \tag{4.45}$$

where $\eta > 0$ is a constant.

4.11 CG method with penalty functions

As mentioned above, CG method can only solve unconstrained optimization problems. Penalty function method converts constrained optimization to unconstrained optimization problems, so that CG method can be used in constrained optimization problems [63]. Many commercial optimization software combines CG method with penalty function to find a solution to constrained optimization problems. The basic idea of penalty function is to create a Pseudo objective function ($\Psi(x)$) by adding constraints violation as penalty [62].

$$\Psi(x) = f(x) + p(x) \tag{4.46}$$

where $p(x)$ is the penalty function. $p(x) = 0$ implies that the x is a feasible solution with no constraint violation.

An optimal solution of Eqn. 4.46 with $p(x) = 0$ implies that such solution is both *optimal* and *feasible.* The performance of penalty function method depends on choice of $p(x)$. The following terms are introduced to simplify the notation.

$$c^{+}_{(i)}(x) \ := \ \max(0, c_{(i)}(x)) \tag{4.47}$$

4.11.1 Exterior penalty method

The exterior penalty adds the square of the penalty violation. Here, the penalty function $p(x)$ is defined as follows:

$$p(x) = r_k \sum_{i=1}^{m} c^{+}_{(i)}(x)^2 \tag{4.48}$$

The subscript k indicates the current iteration of the numerical optimization (i.e., k^{th} iteration of the optimization).

If $f(x)$ and $c_{(i)}(x)$ are continuous and twice-differential, then the contour of pseudo objective function ($\Psi(x)$) is also continuous and twice-differentiable. In general, the penalty multiplier (r_k) starts with a small value. The value of the multiplier increases as the optimization process continues. In this way, constraints violation is more severely penalized as the optimization is getting closer to the convergence.

4.11.2 Interior penalty method

Interior penalty method drives the design away from the boundaries of the constraints as it approaches the boundaries from the negative side. Here, the penalty function $p(x)$ is defined as follows:

$$p(x) = r_k \sum_{i=1}^{m} \frac{-1}{c_{(i)}(x)} \tag{4.49}$$

r_k is initially a large number, and it decreases as the optimization progresses. The penalty value gets arbitrarily bigger as $c_{(i)}$ approaches zero, and the value of $c_{(i)}$ cannot be identical to zero. In interior penalty method, the converged solution always resides within the feasible space (i.e, $c_{(i)} < 0$).

A log barrier is an alternative form of Eqn. 4.49. The log barrier method defines that penalty term as follows:

$$p(x) = r_k \sum_{i=1}^{m} -\log[c_{(i)}(x)] \tag{4.50}$$

Similar to the interior penalty method, the penalty term gets arbitrarily bigger as $c_{(i)}$ approaches zero. The log barrier method is recommended when the problem is numerically better conditioned.

In practice, the interior penalty method is difficult to use. The method requires the optimizer to start from a feasible point (x_0), and all subsequent iterations (x_k) must stay within the feasible domain. This is hard to attain in a problem where it is difficult to identify a feasible domain. Moreover, the interior penalty method introduces singularities at $c_{(i)} = 0$. Singularities are undesirable in numerical computations because they often cause overflow.

CHAPTER V

Box-constrained optimization

A box-constrained optimization problem is a class of optimization problems with upper and lower bounds of the design variables. Many practical engineering problems are box-constrained problems, and many constrained optimization algorithms, such as augmented Lagrangian or penalty schemes, treat box-constrained problems as sub-problems [66]. Thus, efficient algorithms to solve box-constrained optimization problems have a significant practical application [18; 32].

A typical constrained optimization algorithm computes an optimal set of design variables and a set of Lagrange multipliers (x^*, λ^*) that satisfy the KKT condition. When each design variables has upper and lower bounds, there exist twice as many Lagrange multipliers as the number of design variables. Subsequently, the size of constrained optimization grows quickly as the number of design variables increases. Hence, a modification is required to handle the box-constraints efficiently.

This chapter introduces the *projected gradient* method and its convergence property. The projected gradient method is a well-recognized search direction modification algorithm that solves box-constrained optimization problems. The algorithm and a convergence analysis of the algorithm is presented in this chapter, and the convergence analysis is the result of Calamai and More's work [9]. In the original paper, Calamai and More proved the convergence properties of projected gradient in the ‘steepest decent’ method with the Wolfe conditions first. Then, the proof is extended to any line-search algorithm with a decent direction.

5.1 Projected gradient method

The box constrained problem is defined as follows:

$$\min_{x\in\mathbb{R}^n}\{f(x) : x \in \Omega\} \tag{5.1}$$

$$\Omega = \{x \in \mathbb{R}^n : lb \leq x \leq ub\} \tag{5.2}$$

Given an inner product norm $||\cdot||$ and a nonempty closed convex set Ω, the projection into Ω is the mapping $P : \mathbb{R}^n \to \Omega$ defined by

$$P(x) = \text{argmin}\{||z - x|| : z \in \Omega\} \tag{5.3}$$

For the remainder of the text, $P(x)$ denotes the projection of x into Ω. Given the projection P into Ω, the gradient projection algorithm is defined by

$$x_{k+1} = P(x_k + \alpha_k \nabla f(x_k)) \tag{5.4}$$

where $\alpha_k > 0$ is the steplength, and ∇f is the gradient of f with respect to the inner product associated with the norm $||\cdot||$.

5.2 Convergence analysis of the projected gradient method

The convergence analysis of the projected gradient method is organized as follows. Basic properties of gradient projection operators are proven first, then the projected gradient method’s convergence properties are proven assuming that each step length satisfies the Wolfe conditions at each iterate. In the analysis of the projected gradient method, it is assumed that Ω is a nonempty closed convex set and that mapping $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable on Ω. Given an iterate x_k in Ω, the steplength α_k is obtained by search the path.

$$x_k(\alpha) := P(x_k - \alpha \nabla f(x_k)), \tag{5.5}$$

where P is the projection into Ω. Given a step $\alpha_k > 0$, the next iterate is defined by $x_{k+1} = x_k(\alpha_k)$ where α_k is chosen to satisfy the Wolfe conditions, which can be rewritten as follows:

$$f(x_{k+1}) \leq f(x_k) + \mu_1 \big\langle\big\langle \nabla f(x_k) \cdot x_{k+1} - x_k \big\rangle\big\rangle \tag{5.6}$$

$$\alpha_k \geq \gamma_1 \quad \text{or} \quad \alpha_k \geq \gamma_2 \bar{\alpha}_k > 0 \tag{5.7}$$

$$f(x_k(\bar{\alpha}_k)) > f(x_k) + \mu_2 \big\langle\big\langle \nabla f(x_k) \cdot x_k(\bar{\alpha}_k) - x_k \big\rangle\big\rangle \tag{5.8}$$

where $\big\langle\big\langle x \cdot y \big\rangle\big\rangle$ denotes a dot product of two vectors x and y. Here, the Wolfe condition is identical to the one in Chapter IV. However, the notation is modified to be consistent with the original Calamai and More's paper [9]. The analysis of the gradient projection method requires some basic properties of the projection operator.

Lemma V.1. *Let P be the projection into Ω.*

1. *If $z \in \Omega$, then $\big\langle\big\langle P(x) - x \cdot z - P(x) \big\rangle\big\rangle \geq 0$ for all $x \in \mathbb{R}^n$*

2. *P is monotone operator, that is, $\big\langle\big\langle P(y) - P(x) \cdot y - x \big\rangle\big\rangle \geq 0$ for $x, y \in \mathbb{R}^n$. If $P(y) \neq P(x)$, then strict inequality holds.*

3. *P is a non-expansive operator, that is, $||P(y) - P(x)|| \leq ||y - x||$ for $x, y \in \mathbb{R}^n$.*

Lemma V.1 provides additional information on projections.

$$\big\langle\big\langle \nabla f(x_k) \cdot x_k - x_k(\alpha) \big\rangle\big\rangle \geq \frac{||x_k(\alpha) - x_k||^2}{\alpha}, \alpha > 0. \tag{5.9}$$

In particular, this implies that

$$\left\langle\left\langle \nabla f(x_k) \cdot x_k - x_{k+1} \right\rangle\right\rangle \geq \frac{||x_{k+1} - x_k||^2}{\alpha_k} \tag{5.10}$$

Note that part *2* of Lemma V.1 implies that

$$\left\langle\left\langle \nabla f(x_k) \cdot x_k - x_{k+1} \right\rangle\right\rangle \geq \frac{||x_{k+1} - x_k||^2}{\alpha_k} \tag{5.11}$$

Lemma V.2. *Let P be the projection into Ω. Given $x \in \mathbb{R}^n$ and $d \in \mathbb{R}^n$, the function ϕ defined by*

$$\phi(\alpha) = \frac{||P(x + \alpha d) - x||}{\alpha}, \quad \alpha > 0, \tag{5.12}$$

is non-increase (antitone).

Proof. Let $\alpha > \beta > 0$ be given. If $P(x + \alpha\ d) = P(x + \beta\ d)$ then $\phi(\alpha) \leq \phi(\beta)$, so only consider the case $P(x + \alpha) \neq P(x + \beta d)$. If $\left\langle\left\langle v \cdot u - v \right\rangle\right\rangle > 0$, then

$$\frac{||u||}{||v||} \leq \frac{\left\langle\left\langle u \cdot u - v \right\rangle\right\rangle}{\left\langle\left\langle v \cdot u - v \right\rangle\right\rangle} \tag{5.13}$$

Let

$$u = P(x + \alpha d) - x, \quad v = P(x + \beta d) - x, \tag{5.14}$$

then the first part of Lemma V.1 implies that

$$\left\langle\left\langle u \cdot u - v \right\rangle\right\rangle \leq \alpha \left\langle\left\langle d \cdot P(x + \alpha d) - P(x + \beta d) \right\rangle\right\rangle, \tag{5.15}$$

and that

$$\left\langle\left\langle v \cdot u - v \right\rangle\right\rangle \geq \beta \left\langle\left\langle d \cdot P(x + \alpha d) - P(x + \beta d) \right\rangle\right\rangle. \tag{5.16}$$

Moreover, since $\alpha > \beta$ and $P(x + \alpha d) \neq P(x + \beta d)$, the second part of Lemma V.1

shows that

$$\langle\langle d \cdot P(x+\alpha d) - P(x+\beta d)\rangle\rangle > 0. \tag{5.17}$$

Hence $\langle\langle v \cdot u - v\rangle\rangle > 0$. □

The convergence analysis of the gradient projection method is presented next. In the following result, it is assumed that $\{x_k\}$ is bounded but assume that ∇f is uniformly continuous.

Theorem V.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω, and let $\{x_k\}$ be the sequence generated by the gradient projection method. If f is bounded below on Ω and ∇f is uniformly continuous on Ω then*

$$\lim_{k\to\infty} \frac{||x_{k+1} - x_k||}{\alpha_k} = 0. \tag{5.18}$$

Proof. Assume that there is an infinite subsequence K_0 such that

$$\frac{||x_{k+1} - x_k||}{\alpha_k} \geq \epsilon > 0, \quad k \in K_0 \tag{5.19}$$

First note that if $k \in K_0$, then

$$\frac{||x_{k+1} - x_k||^2}{\alpha_k} \geq \epsilon \max\{\epsilon\alpha_k, ||x_{k+1} - x_k||\}, \tag{5.20}$$

and that since $\{f(x_k)\}$ converges, $\{\nabla f(x_k), x_k - x_{k+1}\}$ also converges to zero. Hence

$$\lim_{k\in K_0, k\to\infty} \alpha_k = 0 \quad \text{and} \quad \lim_{k\in K_0, k\to\infty} ||x_{k+1} - x_k|| = 0. \tag{5.21}$$

Set $\bar{x}_{k+1} := x_k(\bar{\alpha}_k)$ and $\beta_k := \min(\alpha_k, \bar{\alpha}_k)$ where $\bar{\alpha}_k$ satisfies Eqn. 5.6. Lemma V.2 implies that

$$\frac{||x_k(\beta_k) - x_k||^2}{\beta_k} \geq \beta_k\Big(\frac{||x_{k+1} - x_k||}{\alpha_k}\Big)\Big(\frac{||\bar{x}_{k+1} - x_k||}{\bar{\alpha}_k}\Big) \tag{5.22}$$

and since $||x_{k+1} - x_k|| \geq \epsilon\alpha_k$ and $\alpha_k \geq \gamma_2\bar{\alpha}_k$ for $k \in K_0$, obtain that

$$\frac{||x_k(\beta_k) - x_k||^2}{\beta_k} \geq \epsilon \min\{1, \gamma_2\}||\bar{x}_{k+1} - x_k||. \tag{5.23}$$

This inequality, together with Eqn. 5.9 and Eqn. 5.11 imply that for $k \in K_0$,

$$\begin{aligned} &\min\{\langle\langle\nabla f(x_k) \cdot x_k - x_{k+1}\rangle\rangle, \langle\langle\nabla f(x_k) \cdot x_k - \bar{x}_{k+1}\rangle\rangle\} \\ &\qquad \geq \epsilon \min\{1, \gamma_2\}||\bar{x}_{k+1} - x_k||. \end{aligned} \tag{5.24}$$

Use Eqn. 5.24 to obtain the desired contradiction. Since $\{\langle\langle\nabla f(x_k) \cdot x_k - x_{k+1}\rangle\rangle\}$ converges to zero. Eqn. 5.24 implies that $\{||\bar{x}_{k+1} - x_k|| : k \in K_0\}$ converges to zero. Thus, the uniform continuity of ∇f shows that if

$$\rho_x(\alpha) := \frac{f(x_k) - f_k(x_k(\alpha))}{\langle\langle\nabla f(x_k) \cdot x_k - x_k(\alpha)\rangle\rangle} \tag{5.25}$$

then

$$|\rho_x(\bar{(\alpha)}_k) - 1| \leq \frac{\mathrm{o}(||\bar{x}_{k+1} - x_k||)}{\langle\langle\nabla f(x_k) \cdot x_k - x_{k+1}\rangle\rangle} \tag{5.26}$$

Hence, Eqn. 5.24 establishes that $\rho_k(\bar{\alpha}_k) > \mu_2$ for all $k \in K_0$ sufficiently large. This is the desire contradiction because Eqn. 5.8 guarantees that $\rho_k(\bar{\alpha}) < \mu_2$ □

Note that in Theorem V.3 the assumption that f is bounded below on Ω is only needed to conclude that $\{f(x_k)\}$ converges, and hence, that $\{\langle\langle\nabla f(x_k) \cdot x_k - x_{k+1}\rangle\rangle\}$ converges to zero. The assumption that ∇f is uniformly continuous is used to conclude that

$$|f(\bar{x}_{k+1}) - f(x_k) - \langle\langle\nabla f(x_k) \cdot \bar{x}_{k+1} - x_k\rangle\rangle = \mathrm{o}(||\bar{x}_{k+1} - x_k||). \tag{5.27}$$

These observations lead to the following result.

Theorem V.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω and let $\{x_k\}$ be the sequence generated by the gradient projection method defined by Eqn. 5.6 and Eqn. 5.7. If some subsequence $\{x_k : k \in K\}$ is bounded then*

$$\lim_{k \in K, k \to \infty} \frac{||x_{k+1} - x_k||}{\alpha_k} = 0 \tag{5.28}$$

Moreover, any limit point of $\{x_k\}$ is a stationary point of Eqn. 5.1.

Proof. Assume that there is an infinite subsequence $K_0 \subset K$ such that

$$\frac{||x_{k+1} - x_k||}{\alpha_k} \geq \epsilon > 0, \quad k \in K_0, \tag{5.29}$$

and reach a contradiction as in the proof of Theorem V.3. The only difference is that since $\{x_k : k \in K\}$ is bounded, $\{f(x_k)\}$ converges and K_0 can be chosen so that $\{x_k : k \in K_0\}$ converges. Hence, continuity of ∇f is sufficient to show that $\rho_k(\bar{\alpha}_k > \mu_2$ for all $k \in K_0$ sufficiently large.

Assume now that $\{x_k\}$ has a limit point x^*. A short computation shows that first part of Lemma V.1 implies that for any $z \in \Omega$

$$\begin{aligned} \alpha_k \big\langle\big\langle \nabla f(x_k) \cdot x_{k+1} - z \big\rangle\big\rangle &\leq \big\langle\big\langle x_{k+1} - x_k \cdot z - x_{k+1} \big\rangle\big\rangle \\ &\leq \big\langle\big\langle x_{k+1} - x_k \cdot z - x_k \big\rangle\big\rangle \leq ||x_{k+1} - x_k||\ ||x_k - z||. \end{aligned} \tag{5.30}$$

Hence,

$$\big\langle\big\langle \nabla f(x_k) \cdot x_k - z \big\rangle\big\rangle \leq \big\langle\big\langle \nabla f(x_k) \cdot x_k - x_{k+1} \big\rangle\big\rangle + \frac{||x_{k+1} - x_k||}{\alpha_k} ||x_k - z|| \tag{5.31}$$

Since $\{f(x_k)\}$ converges, Eqn. 5.6 implies that $\{\big\langle\big\langle \nabla f(x_k) \cdot x_k - x_{k+1} \big\rangle\big\rangle\}$ converges to zero, and thus the above inequalities shows that $\big\langle\big\langle \nabla f(x_k) \cdot x^* - z \big\rangle\big\rangle \leq 0$. This establishes that x^* is a stationary point of the problem of Eqn. 5.1. $\square$

It was shown that limit points of the gradient projection algorithm are stationary when α_k satisfies Eqn. 5.6 and Eqn. 5.8. In this section, this result is improved by showing that the sequence of projected gradients converges to zero provided that steps $\{\alpha_k\}$ are bounded.

The definition of the projected gradient requires some notions from convex analysis. Assume that Ω is a nonempty closed convex set in $\mathbb{R}^n$ and that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable on Ω. A direction v is feasbile at $x \in \Omega$ if $x + \tau v$ belongs to Ω for all $\tau > 0$ sufficiently small. The *tangent cone* $T(x)$ is defined as the closure of the cone of all feasible directions. The *projected gradient* $\nabla_\Omega f(x)$ uniquely. Also note that for an arbitrary set Ω, the tangent cone at $x \in \Omega$ can also be defined as the set of all $v \in \mathbb{R}^n$ such that

$$v = \lim_{k\to\infty} \frac{x_k - x}{\beta_k} \tag{5.32}$$

for some sequence $\{x_k\}$ in Ω converging to x, and some sequence of positive scalars $\{\beta_k\}$ converging to zero. It is not difficult to show that both definitions lead to the same tangent cone when Ω is convex.

Lemma V.5. *Let $\nabla_\Omega f(x)$ be the projected gradient of f at $x \in \Omega$.*

1. $-\langle\langle \nabla f(x) \cdot \nabla_\Omega f(x) \rangle\rangle = ||\nabla_\Omega f(x)||^2$.

2. *min* $\{\langle\langle \nabla f(x) \cdot v \rangle\rangle : v \in T(x), ||v|| \leq 1\} = -||\nabla_\Omega f(x)||$.

3. *The point $x \in \Omega$ is a stationary point of Eqn. 5.1 if and only if* $\nabla_\Omega f(x) = 0$.

Proof. First, establish the part *1* of the lemma. Since $T(x)$ is a cone, and since $\nabla_\Omega f(x)$ belongs to $T(x)$, the part *1* of Lemma V.1 shows that if $\lambda \leq 0$ then

$$\langle\langle \nabla_\Omega f(x) + \nabla f(x) \cdot (\lambda - 1)\nabla_\Omega f(x) \rangle\rangle \geq 0. \tag{5.33}$$

Setting $\lambda = 0$ and $\lambda = 2$ in this inequalities yields part *1*. Then, prove part *2* by noting that if $v \in T(x)$ and $||v|| \leq ||\nabla_\Omega f(x)||$ then

$$||\nabla_\Omega f(x) + \nabla f(x)||^2 \leq ||v + \nabla f(x)||^2 \leq ||\nabla_\Omega f(x)||^2 + 2\big\langle\big\langle \nabla f(x) \cdot v \big\rangle\big\rangle + || \tag{5.34}$$

This inequality and part *1* yield part *2*. Here, the part *3* is proven. Note that if the point x is a stationary point then

$$\big\langle\big\langle \nabla f(x) \cdot z - x \big\rangle\big\rangle \geq 0, \quad z \in \Omega \tag{5.35}$$

If v is a feasible direction at x then $x + \tau v$ belongs to Ω for some $\tau > 0$ and hence setting $z = x + \tau v$ yields $\big\langle\big\langle \nabla f(x) \cdot v \big\rangle\big\rangle \geq 0$ for all $v \in T(x)$, and thus *2* implies that $\nabla_\Omega f(x) = 0$. Conversely, if $\nabla_\Omega f(x) = 0$ then *2* implies that $\big\langle\big\langle \nabla f(x) \cdot v \big\rangle\big\rangle$ for all $v \in T(x)$, and since $z - x \in T(x)$ for any $z \in \Omega$, this shows that x is a stationary point of Eqn. 5.1. $\square$

Part *2* of Lemma V.1 justifies the definition of the projected gradient by showing that $\nabla_\Omega f(x)$ is a steepest descent direction for f. This result will be used repeatedly in the convergence analysis of the gradient projection method.

At first glance, part *3* of Lemma V.1 suggests that an iterative scheme for Eqn. 5.1 should drive $\nabla_\Omega f(x_k)$ to zero. However, this may not be possible because $\nabla_\Omega f$ can be bounded away from zero in a neighborhood of a stationary point.

Theorem V.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω, and let $\{x_k\}$ be the sequence generated by the gradient projection method defined by Eqn. 5.6 and Eqn. 5.7 with*

$$\alpha_k \leq \gamma_3 \tag{5.36}$$

for some constant γ_3. If f is bounded below on Ω and ∇f is uniformly continuous on

Ω *then*

$$\lim_{k\to\infty} ||\nabla_\Omega f(x_k)|| = 0 \tag{5.37}$$

Proof. Let $\epsilon > 0$ be given and choose a feasible direction v_k with $||v_k|| \leq 1$ such that

$$||\nabla_\Omega f(x_k)|| \leq -\big\langle\big\langle \nabla f(x_k) \cdot v_k \big\rangle\big\rangle + \epsilon \tag{5.38}$$

Now note that part *1* of Lemma V.1 shows that for any $z_{k+1} \in \Omega$

$$\alpha_k \big\langle\big\langle \nabla f(x_k) \cdot x_{k+1} - z_{k+1} \big\rangle\big\rangle \leq \big\langle\big\langle x_{k+1} - x_k \cdot z_{k+1} - x_{k+1} \big\rangle\big\rangle \leq ||x_{k+1} - x_k||\ ||x_{k+1} - z_{k+1}||.$$

Since v_k is a feasible direction, $z_k = x_k + \tau_k v_k$ belongs to Ω for some $\tau_k > 0$. Thus the above inequality and Theorem V.3 show that

$$\limsup_{k\to\infty} -\big\langle\big\langle \nabla f(x_k) \cdot v_{k+1} \big\rangle\big\rangle \leq 0. \tag{5.39}$$

Since α_k is bounded above, Theorem V.3 shows that $\{||x_{k+1} - x_k||\}$ converges to zero, and thus use the uniform continuity of ∇f to conclude that

$$\limsup_{k\to\infty} -\big\langle\big\langle \nabla f(x_k) \cdot v_{k+1} \big\rangle\big\rangle \leq 0. \tag{5.40}$$

The choice of v_k guarantees that

$$\limsup_{k\to\infty} ||\nabla_\Omega f(x_{k+1})|| < \epsilon, \tag{5.41}$$

and since $\epsilon > 0$ is arbitrary, this proves the result. $\square$

A variation on Theorem V.6 is established to conclude the analysis. In this result, the assumptions that f is bounded below on Ω and that ∇f is uniformly continuous on Ω are replaced by the assumption that some subsequence of $\{x_k\}$ is bounded.

Theorem V.7. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω, and let $\{x_k\}$ be the sequence generated by the gradient projection method defined by Eqn. 5.6, Eqn. 5.7, and Eqn. 5.36. If some subsequence $\{x_k : k \in K\}$ is bounded, then*

$$\lim_{k \in K, k \to \infty} ||\nabla_\Omega f(x_{k+1})|| = 0. \tag{5.42}$$

Proof. Assume that there is an infinite subsequence $K_0 \subset K$ and an $\epsilon_0 > 0$ such that

$$||\nabla_\Omega f(x_{k+1})|| \geq \epsilon_0, \quad k \in K_0. \tag{5.43}$$

Since $\{x_k : k \in K\}$ is bounded, choose K_0 so that $\{x_k : k \in K_0\}$ converges. The proof proceeds as in Theorem V.6 except that] Theorem V.4 is used instead of Theorem V.3. Thus,

$$\limsup_{k \in K_0, k \to \infty} ||\nabla_\Omega f(x_{k+1})|| \leq \epsilon \tag{5.44}$$

for any $\epsilon > 0$. This contradicts (3.3) for $\epsilon < \epsilon_0$, and thus establishes the result. □

5.3 Extension to conjugate gradient method

In the previous section, the convergence property of projected gradient method is shown when coupled with the steepest decent method. Now, this result is extended to CG method. Assume that Ω is an arbitrary nonempty closed convex set, and assume that the gradient projected conjugate gradient method choose x_{k+1} by either

- *Condition 1.* $x_{k+1} = P(x_k - \alpha_k \nabla f(x_k))$ where α_k satisfies Eqn. 5.6, Eqn. 5.7, and Eqn. 5.36.
- *Condition 2.* $x_{k+1} \in \Omega$ such that $f(x_{k+1}) \leq f(x_k)$.

Let a set K_{PG} be a set of iterates k which generate x_{k+1} by the projected gradient method.

Theorem V.8. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω, and let $\{x_k\}$ be the sequence generated by the projected gradient conjugate gradient method such that x_{k+1} satisfies Condition 1 and Condition 2. Assume that K_{PG} is infinite. If some subsequence $\{x_k : k \in K\}$ with $K \subset K_{PG}$ is bounded then*

$$\lim_{k \in K, k \to \infty} ||\nabla_\Omega f(x_{k+1})|| = 0. \tag{5.45}$$

Moreover, any limit point of $\{x_k : k \in K_{PG}\}$ is a stationary point of Eqn. 5.1.

Proof. Only outline of the proof is presented here.

$$\lim_{k \in K, k \to \infty} \frac{||x_{k+1} - x_k||}{\alpha_k} = 0 \tag{5.46}$$

holds if

- $\{x_k\}$ satisfies *Condition 1* and *Condition 2.*
- f satisfies the assumptions of Theorem V.4.
- K is an infinite subset of K_{PG}

Since Eqn. 5.46 holds, the arguments used in the proofs of Theorem V.6 and Theorem V.7 yield

$$\limsup_{k \in K, k \to \infty} -\langle\langle \nabla f(x_{k+1}) \cdot v_{k+1} \rangle\rangle \leq 0. \tag{5.47}$$

Since v_{k+1} is any feasible vector with $||v_{k+1}|| \leq 1$, this implies that for any $\epsilon > 0$

$$\limsup_{k \in K, k \to \infty} ||\nabla_\Omega f(x_{k+1})|| \leq \epsilon. \tag{5.48}$$

Hence Eqn. 5.45 holds as desired. $\square$

This shows that the convergence analysis is valid for the projected gradient conjugate gradient method. In the main body of the dissertation, the projected gradient method is integrated to CG method to solve box-constrained optimization problems.

CHAPTER VI

A bi-level multi-objective algorithm

In this chapter, a new bi-level multi-objective algorithm is presented. The new algorithm is created by combining strengths of known algorithms. The new algorithm uses a bi-level structure to decompose an optimization problem, and a new multi-objective function transformation method is proposed to accommodate the presence of global and local design variables in the bi-level structure. The utility of the algorithm is evaluated by solving two multi-objective optimization algorithms.

6.1 Bi-level structure

The algorithm presented in this chapter follows a bi-level hierarchy that contains several discipline level optimizations and a single system level optimization. Each discipline optimization pursues improvement of its own objective function while meeting all corresponding performance expectations which are stated as constraints. The system level optimization consolidates the design by using information from the discipline level optimizations, their objective functions, and their constraints. At the system level, all discipline level objective functions are combined using a multi-objective metric for generating the system level objective function. In this manner, a consolidated and balanced design is identified. All discipline level constraints are combined and considered as system level constraints during the optimization. Thus, the final design is feasible based on all discipline level performance expectations.

The set of all design variables from all disciplines is decomposed into local and global design variables. The design variables which are encountered only in a single discipline comprise the local design variables, while design variables which are used by two or more disciplines are the global design variables. All discipline level optimizations adjust the values of the local design variables and the system optimization determines the values for the global design variables. This decomposition reduces the number of design variables determined at the system level. At the same time control of the local design variables is shifted to several discipline level optimizations. Since the values of the global design variables are determined only at the system level, there are no consistency issues for the values assigned to them (i.e. different disciplines do not have an opportunity to assign different values to the shared design variables).

6.1.1 Discipline level

Each discipline has its own objective function ($f_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}})$), constraints ($c_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}})$), and upper and lower bounds for local design variables ($lb_{\hat{i}} \leq x_{\hat{i}} \leq ub_{\hat{i}}$). The global design variables ($x_{\hat{0}}$) are treated as constant in the discipline level optimizations.

$$\min_{x_{\hat{i}}}(f_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}})) \tag{6.1a}$$

$$\text{subject to}$$

$$c_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}}) \leq 0 \tag{6.1b}$$

$$lb_{\hat{i}} \leq x_{\hat{i}} \leq ub_{\hat{i}} \tag{6.1c}$$

6.1.2 System level

A multi-objective metric is used for combining the objective functions from all the disciplines when defining the system level objective function (f_{sys}). Lower ($lb_{\hat{0}}$)and upper bound ($ub_{\hat{0}}$) are considered for the global design variables. The constraints from all discipline level optimizations ($c_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}}) \leq 0 \; \forall \; \hat{i}$) are imposed as constraints in the system level optimization to satisfy all the discipline level constraints [27]. The system optimization statement is defined as follows:

$$\min_{x_{\hat{0}}} \; f_{sys}(x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}) \tag{6.2a}$$

subject to

$$c_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}}) \leq 0 \quad \forall \quad \hat{i} \tag{6.2b}$$

$$c_{sys}(x_{\hat{0}}, x_{\hat{1}}, \cdots, x_{\hat{k}}) \leq 0 \tag{6.2c}$$

$$lb_{\hat{0}} \leq x_{\hat{0}} \leq ub_{\hat{0}} \tag{6.2d}$$

6.2 Multi-objective function transformation

When defining a system level objective function, multiple concepts from multi-objective function transformation is used. It finds a compromised solution by minimizing a scaled Euclidean distance to the Utopia point. The Euclidean distance is required to be scaled because the optimizer favors the objective function with the largest order of magnitude without proper scaling. The relative importance of each objective is decided by a user, and the Pareto front can be found by running optimizations multiple times with varying weights. Then, exponential function is used to enhance the performance of the objective function.

The system level objective function is defined as follows:

$$f_{sys} = exp\left(\sum_{\hat{i}=1}^{\hat{k}} w_{\hat{i}} \left(\frac{f_{\hat{i}}(x_{\hat{0}}, x_{\hat{1}}, \cdots x_{\hat{k}}) - f_{\hat{i}}^{*}}{prr_{\hat{i}}}\right)^{2}\right) \tag{6.3}$$

An upper-lower bound transformation and the weighted sum approach are used for defining the multi-objective function of the system level optimization. The distance from the Utopia point is used in the definition of the system level objective function.

The Utopia point is determined by the infeasible ideal case where every objective in MDO problem attains its optimal value $([(f_{\hat{1}})^{*}(f_{\hat{2}})^{*} \cdots (f_{\hat{k}})^{*}])$. When each objective is scaled using the upper and lower bound approach, then minimizing the distance to the Utopia point can identify both convex and non-convex Pareto fronts [42; 43]. This concept has been well explored not only in the field of Multi-Objective optimization, but also in the field of decision making [31; 37]. The upper-lower bound method is considered to be one the most robust multi-objective function transformation method.

Exponential terms are used in the multi-objective function when accounting for the difference between the values of each objective function at the current design point vs. the value of the objective function at the utopia point. It has been demonstrated that the use of exponential terms improves the performance of the multi-objective optimizations when finding a non-convex Pareto front [4].

The Euclidean distance to the Utopia point $(f_{\hat{i}}(x_0, x_{\hat{i}}) - (f_{\hat{i}})^{*}\ )$ is scaled using an upper-lower-bound approach. The need for proper scaling factors $(prr_{\hat{i}})$ arises when the objective functions $(f_{\hat{i}})$ are in different orders of magnitude. Without proper scaling, the optimizer favors the improvement in the objective function with the highest order of magnitudes. Therefore, each objective has to be properly scaled for the system level optimizer to measure the improvement in each objective function without bias [43].

The basic notion of the upper-lower-bound approach is to scale each objective by the difference between its highest value on the Pareto front (upper bound) and its optimum value (lower bound). This upper bound is called Zenith point ($Z_{\hat{i}}$), and prr is the difference between the Zenith point and the objective optimal value $(f_{\hat{i}})^*$.

$$prr_{\hat{i}} = Z_{\hat{i}} - (f_{\hat{i}})^* \tag{6.4}$$

In this work, the Zenith point is set equal to the highest value that an objective function acquires at the optimal points of all other disciplines (i.e., evaluating i^{th} objective function at j^{th} objective optimal design variable). The conventional upper-lower bound method considers no local variables across multiple disciplines. When there are no local variables, the calculation of the Zenith point of the i^{th} objective is straight forward:

$$Z_{\hat{i}} = \max\left[f_{\hat{i}}(x_0^{\hat{1}}) \cdots f_{\hat{i}}(x_0^{\hat{k}})\right] \tag{6.5}$$

The method is expanded to accommodate a bi-level structure, which includes local design variables. In the presence of local design variables, identifying the Zenith point for the objective function of each discipline is a two-step process. First, each discipline optimization is conducted multiple times with respect to *both* global and local variables resulting in a set of optimal variables for each discipline $(x_0^i, x_{\hat{i}}^*)$. Then, each discipline optimization is conducted with respect to only the local variables while the global design variables are set equal to each one of the other disciplines optimal global variables (i.e., the i^{th} discipline optimization is conducted multiple times, and each time the global design variables are set to be equal to the optimal values of the j^{th} discipline, x_0^j). The corresponding value for the objective function becomes $(f_i(x_0^j, x_i^j)$. Then, the Zenith point is determined as:

$$Z_{\hat{i}} = \max\left[f_{\hat{i}}(x_{\hat{0}}^{\hat{1}}, x_{\hat{i}}^{\hat{1}}), f_{\hat{i}}(x_{\hat{0}}^{\hat{2}}, x_{\hat{i}}^{\hat{2}}), \cdots f_{\hat{i}}(x_{\hat{0}}^{\hat{k}}, x_{\hat{i}}^{\hat{k}})\right] \tag{6.6}$$

6.3 Computational flow

In this algorithm, the Utopia point is computed first. Then, the $prr_{\hat{i}}$ values are evaluated for each discipline using Eqn. 6.4. The latter along with the Utopia point are needed for calculating the multi-objective function in the system level optimization, Eqn. 6.3.

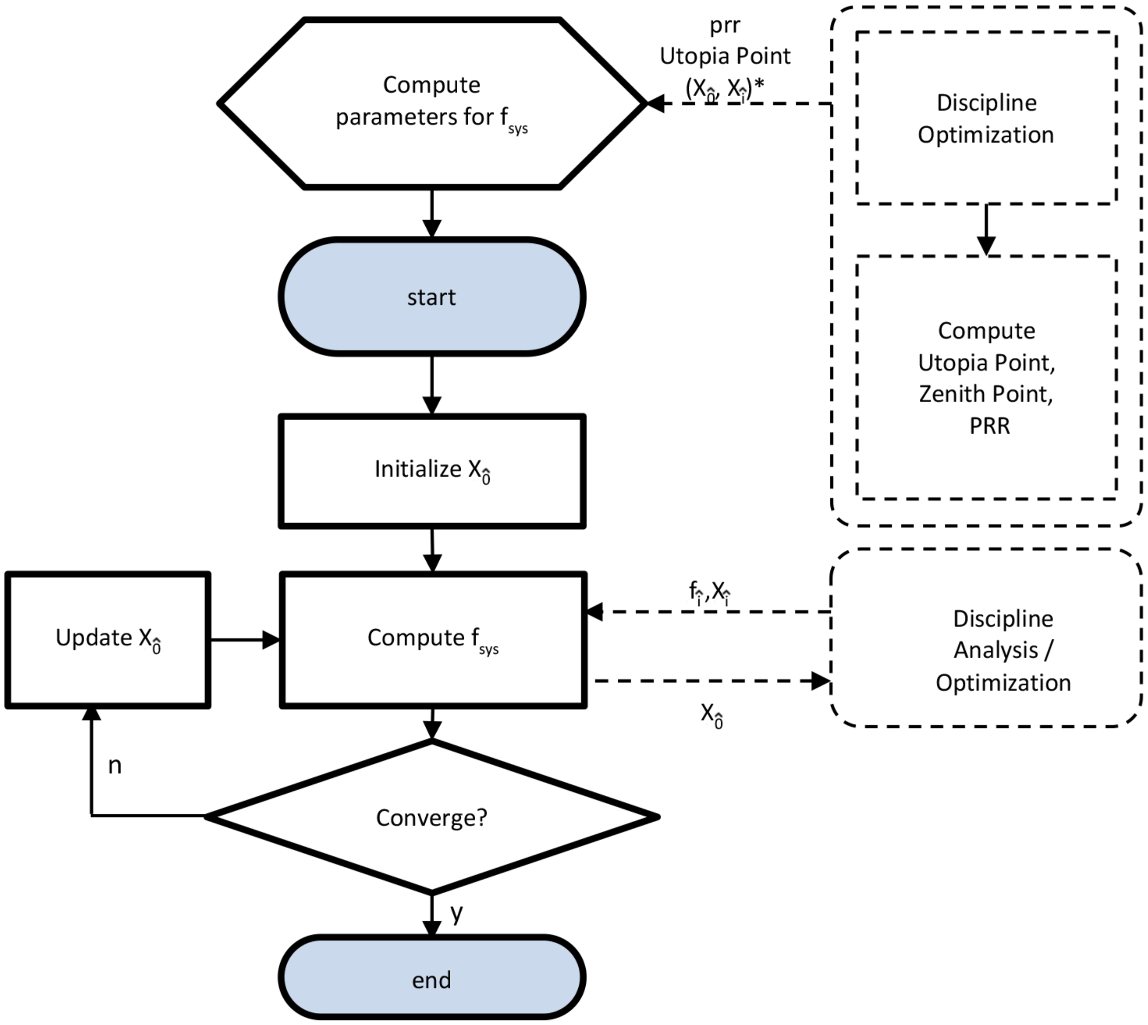
Compute parameters for f_{sys}
prr
Utopia Point
$(X_{\hat{0}}, X_{\hat{i}})^*$
Discipline Optimization
Compute Utopia Point, Zenith Point, PRR
start
Initialize $X_{\hat{0}}$
Update $X_{\hat{0}}$
Compute f_{sys}
$f_{\hat{i}}, X_{\hat{i}}$
$X_{\hat{0}}$
Discipline Analysis / Optimization
n
Converge?
y
end

Figure 6.1: Flow of the Algorithm

Once the MDO analysis starts, all discipline optimizations are executed at every system level iteration. The system level optimization provides the values for the global design variables to the discipline optimizations, and each discipline optimization returns the optimal set of values for the local design variables ($x_{\hat{i}}$), and the optimal value for the corresponding objective function based on Eqn 6.1. The system level optimizer updates the global design variables based on the performance of the previous iteration. The process repeats until the system level optimization satisfies convergence criteria. In this manner, the system level optimization (Eqn. 6.2) is solved. The weights ($w_{\hat{i}}$) assign relative importance of the objectives (Eqn. 6.3). By changing the weights, the system level optimizer converges on a different point of the Pareto front. In all analytical examples presented, the total sum of the weights is equal to 1. ($\sum_{\hat{i}=1}^{\hat{k}} w_{\hat{i}} = 1$)

6.4 Numerical examples

In order to demonstrate the utility of the algorithm, two MDO problems are solved. They are based on analytical optimization problems from the literature. In both cases, the Pareto front is also computed using Monte Carlo simulations for demonstrating convergence of the new algorithm on the Pareto front. The first MDO problem is created by designating the geometric programming optimization [33], and the Golinski speed reducer optimization [24] as each one of the two discipline optimizations of the MDO. Analyses are performed twice, first by considering all the design variables to be continuous and then by using mixed design variables. The second MDO problem is a multi-objective version of the Sellar collaborative optimization [57]. It demonstrates that new algorithm can consider collaborative optimizations as discipline level optimization. A computer with the following spec was used for all optimizations: Intel Core i7-3770 CPU @ 3.40 GHz (8 CPUs) and 16 GB RAM.

6.4.1 GG problem

Two analytical optimization problems from the literature are combined and designated as discipline optimizations in order to generate a MDO analysis that can be solved by the new bi-level multi-objective algorithm. The original geometric programming optimization statement was utilized for demonstrating the utility of the Target Cascading Method [33]. It has six inequality constraints, four equality constraints, and fourteen design variables. Similarly, the Golinski speed reducer optimization [24] has eleven inequality constraints, seven design variables and was proposed as a benchmark optimization problem to compare and contrast various optimization methods. These two optimizations are unrelated physically; they are designated as discipline optimization because the optimal values for their objective function exhibit different orders of magnitude. Thus, the performance of the upper-lower bound transformation utilized in the definition of the system level objective function can be tested. When the two optimizations are combined for defining the MDO statement, three design variables from each discipline are designated as global design variables ($x_{\hat{0}} = [u_1, u_2, u_3]$). This results into eleven local design variables ($x_{\hat{1}} = [s_1, \cdots, s_{11}]$) for discipline 1 (i.e. the geometric programming optimization); and four local design variables ($x_{\hat{2}} = [t_1, t_2, t_3, t_4]$) for discipline 2 (i.e. the Golinski speed reducer optimization).

The structure of the MDO analysis, the definitions of the constraints, and the objective functions are included in Fig. 6.2 along with the ranges for the global and the local design variables. Analysis is performed first by considering continuous values for all design variables. The corresponding utopia point and the plausible reduction ranges which are used in the upper-lower bound transformation are summarized in Table 6.1. Please note that the equality constraints are denoted as $ceq(x)$ and inequality constraints are denoted as $c(x)$. The differences in the magnitudes of the objective functions exhibited by two disciplines at the utopia point ($(f_{\hat{1}})^* \approx 17.7$,

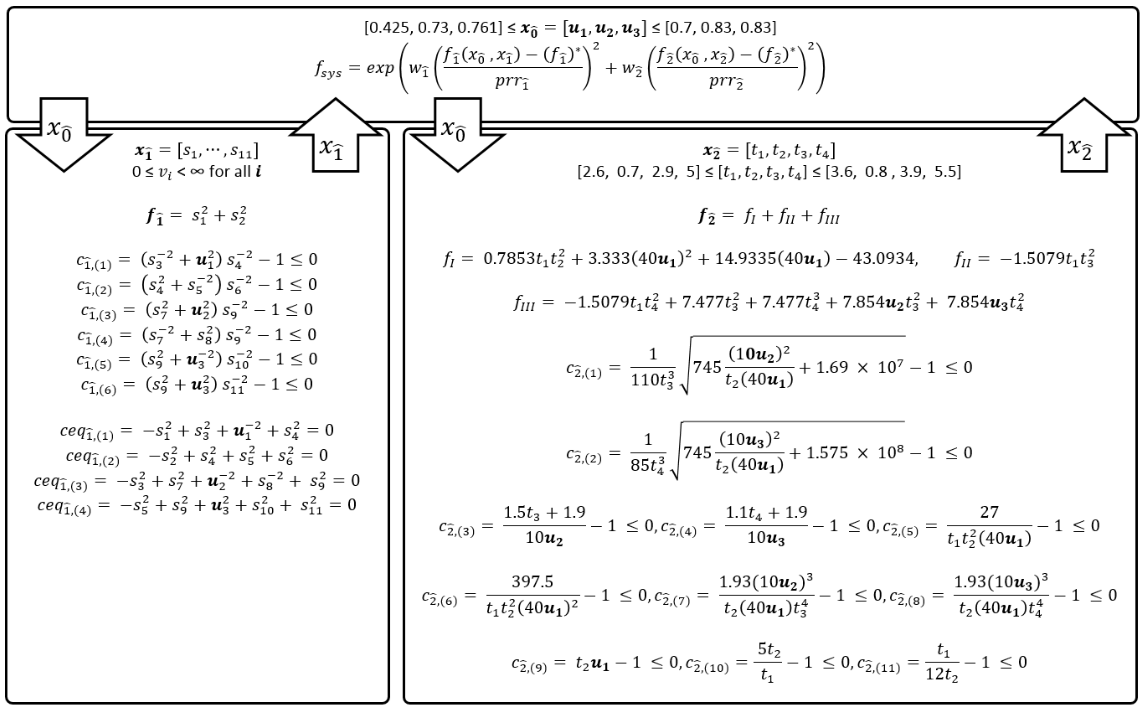
$[0.425, 0.73, 0.761] \leq \boldsymbol{x}_{\hat{0}} = [\boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{u_3}] \leq [0.7, 0.83, 0.83]$
$f_{sys} = exp\left(w_{\hat{1}}\left(\frac{f_{\hat{1}}(x_{\hat{0}}, x_{\hat{1}}) - (f_{\hat{1}})^*}{prr_{\hat{1}}}\right)^2 + w_{\hat{2}}\left(\frac{f_{\hat{2}}(x_{\hat{0}}, x_{\hat{2}}) - (f_{\hat{2}})^*}{prr_{\hat{2}}}\right)^2\right)$
$x_{\hat{0}}$
$x_{\hat{1}}$
$x_{\hat{0}}$
$x_{\hat{2}}$
$\boldsymbol{x}_{\hat{1}} = [s_1, \cdots, s_{11}]$
$0 \leq v_i < \infty$ for all $\boldsymbol{i}$
$\boldsymbol{f}_{\hat{1}} = s_1^2 + s_2^2$
$c_{\hat{1},(1)} = (s_3^{-2} + \boldsymbol{u}_1^2)\, s_4^{-2} - 1 \leq 0$
$c_{\hat{1},(2)} = (s_4^2 + s_5^{-2})\, s_6^{-2} - 1 \leq 0$
$c_{\hat{1},(3)} = (s_7^2 + \boldsymbol{u}_2^2)\, s_9^{-2} - 1 \leq 0$
$c_{\hat{1},(4)} = (s_7^{-2} + s_8^2)\, s_9^{-2} - 1 \leq 0$
$c_{\hat{1},(5)} = (s_9^2 + \boldsymbol{u}_3^{-2})\, s_{10}^{-2} - 1 \leq 0$
$c_{\hat{1},(6)} = (s_9^2 + \boldsymbol{u}_3^2)\, s_{11}^{-2} - 1 \leq 0$
$ceq_{\hat{1},(1)} = -s_1^2 + s_3^2 + \boldsymbol{u}_1^{-2} + s_4^2 = 0$
$ceq_{\hat{1},(2)} = -s_2^2 + s_4^2 + s_5^2 + s_6^2 = 0$
$ceq_{\hat{1},(3)} = -s_3^2 + s_7^2 + \boldsymbol{u}_2^{-2} + s_8^{-2} + s_9^2 = 0$
$ceq_{\hat{1},(4)} = -s_5^2 + s_9^2 + \boldsymbol{u}_3^2 + s_{10}^2 + s_{11}^2 = 0$
$\boldsymbol{x}_{\hat{2}} = [t_1, t_2, t_3, t_4]$
$[2.6,\ 0.7,\ 2.9,\ 5] \leq [t_1, t_2, t_3, t_4] \leq [3.6,\ 0.8,\ 3.9,\ 5.5]$
$\boldsymbol{f}_{\hat{2}} = f_I + f_{II} + f_{III}$
$f_I = 0.7853 t_1 t_2^2 + 3.333(40\boldsymbol{u_1})^2 + 14.9335(40\boldsymbol{u_1}) - 43.0934, \quad f_{II} = -1.5079 t_1 t_3^2$
$f_{III} = -1.5079 t_1 t_4^2 + 7.477 t_3^2 + 7.477 t_4^3 + 7.854\boldsymbol{u_2} t_3^2 + 7.854\boldsymbol{u_3} t_4^2$
$c_{\hat{2},(1)} = \frac{1}{110 t_3^3}\sqrt{745\frac{(10\boldsymbol{u_2})^2}{t_2(40\boldsymbol{u_1})} + 1.69 \times 10^7} - 1 \leq 0$
$c_{\hat{2},(2)} = \frac{1}{85 t_4^3}\sqrt{745\frac{(10\boldsymbol{u_3})^2}{t_2(40\boldsymbol{u_1})} + 1.575 \times 10^8} - 1 \leq 0$
$c_{\hat{2},(3)} = \frac{1.5t_3 + 1.9}{10\boldsymbol{u_2}} - 1 \leq 0, c_{\hat{2},(4)} = \frac{1.1t_4 + 1.9}{10\boldsymbol{u_3}} - 1 \leq 0, c_{\hat{2},(5)} = \frac{27}{t_1 t_2^2(40\boldsymbol{u_1})} - 1 \leq 0$
$c_{\hat{2},(6)} = \frac{397.5}{t_1 t_2^2(40\boldsymbol{u_1})^2} - 1 \leq 0, c_{\hat{2},(7)} = \frac{1.93(10\boldsymbol{u_2})^3}{t_2(40\boldsymbol{u_1})t_3^4} - 1 \leq 0, c_{\hat{2},(8)} = \frac{1.93(10\boldsymbol{u_3})^3}{t_2(40\boldsymbol{u_1})t_4^4} - 1 \leq 0$
$c_{\hat{2},(9)} = t_2\boldsymbol{u_1} - 1 \leq 0, c_{\hat{2},(10)} = \frac{5t_2}{t_1} - 1 \leq 0, c_{\hat{2},(11)} = \frac{t_1}{12t_2} - 1 \leq 0$

Figure 6.2: GG problem Statement

$(f_{\hat{2}})^* \approx 2.57 \cdot 10^3$) is balanced when the multi-objective system function is defined by dividing the terms that include each objective function with the plausible reduction range ($prr_1 \approx 2.68$, $prr_2 \approx 1.81 \cdot 10^3$).

$\hat{i}$	$x_{\hat{0}}^{\hat{i}}$	$x_{\hat{i}}^*$	$f_{\hat{i}}$	$prr_{\hat{i}}$
$\hat{1}$	[0.70, 0.82, 0.82]	[2.90, 3.04, 2.40, 0.81, 2.80, 0.89, 0.99, 0.79, 1.28, 1.77, 1.52]	17.7	2.68
$\hat{2}$	[0.43, 0.76, 0.78]	[3.60, 0.70, 3.34, 5.29]	$2.59 \cdot 10^3$	$1.81 \cdot 10^3$

Table 6.1: GG problem single optimization result of continuous variable

The MATLAB fmincon gradient optimizer was employed for driving the discipline and the system level optimization. The Pareto front is also evaluated through a Monte Carlo simulation and plotted in Fig. 6.3. The Monte Carlo simulation evaluated 10,000 vectors of global design variables that are chosen randomly. The discipline optimizers attempted to find corresponding optimal local design variables. The simulation lasted 622 seconds; it identified 5,919 feasible points and created a convex

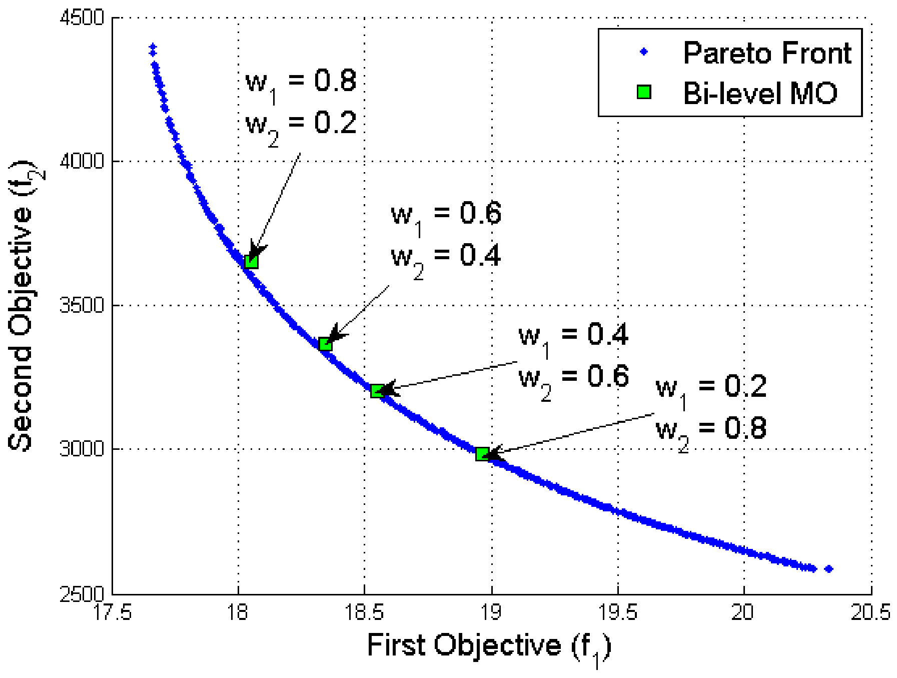

Figure 6.3: GG problem optimization result

Pareto front with 480 non-dominated points.

Analysis is performed by considering four different sets of weights ($w_{\hat{1}}$& $w_{\hat{2}}$) in the definition of the system level objective function. The optimization results are presented in Fig. 6.3 as design points in the space defined by the values of the objective functions from the two disciplines. The four pairs of value used for the weights are also presented in Fig. 6.3.

As anticipated, the optimal solutions that are identified using different sets of weights converge to different locations of the Pareto front. When higher weight is allocated to the first discipline, then the optimal point gravitates towards points on the Pareto front that exhibit lower values for the objective function of the first discipline. By running a series of optimizations with varying set of weights ($w_{\hat{1}}$ & $w_{\hat{2}}$), the algorithm can identify points on the Pareto front.

6.4.2 GG problem with an integer variable

The structure of the bi-level MDO algorithm can accommodate mixed design variables if a genetic algorithm is used for driving each optimization instead of a gradient solver. In order to demonstrate this capability, the GG problem is converted into one with mixed design variables. Specifically, the global design variable u_2 is converted to a discrete design variable that can acquire any one from ten discrete values of [0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.80, 0.81, 0.81, 0.82, 0.83].

Optimization analyses are performed using the MATLAB genetic algorithm function 'ga' with the default setting and population size of 50 and 200. The computational time increases significantly when replacing the gradient solver with the genetic algorithm in MATLAB. A Monte Carlo simulation was also performed for determining the Pareto front. The simulation time was 1250 seconds and 268 Pareto were identified from 10,000 sets of global design variables. The Pareto fronts when considering mixed or continuous design variables are similar but not identical. The results from the MDO analysis that uses a population size of 50 are presented in Fig. 6.4 and the results from a population size of 200 are presented in Fig. 6.5.

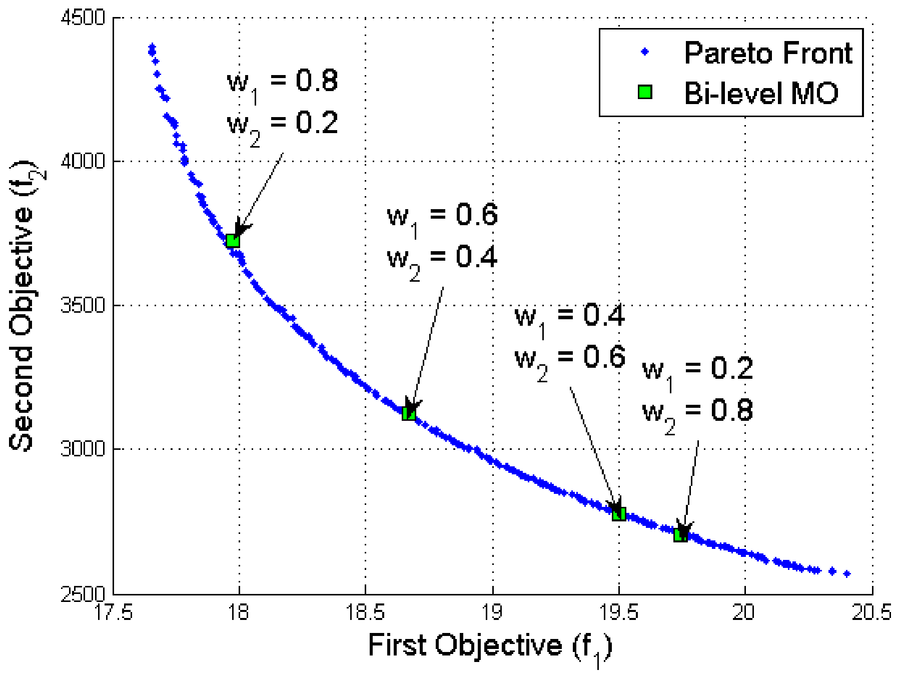

Figure 6.4: Mixed-Integer optimization result w/ population 50

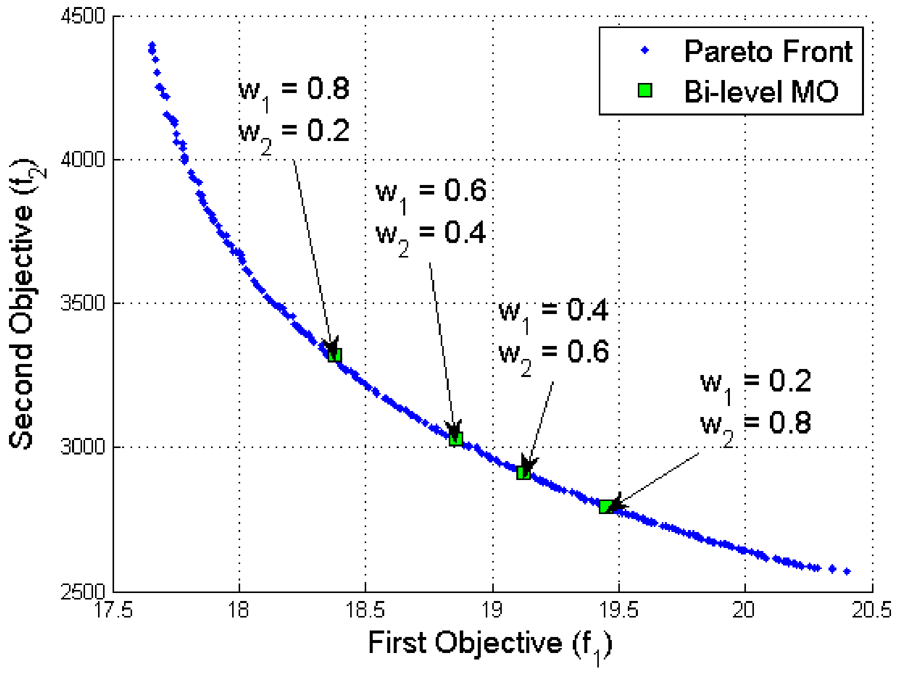

Figure 6.5: Mixed-Integer optimization result w/ population 200

6.4.3 Sellar problem

The original Sellar problem, shown in Eqn. 6.7a, is a minimization problem with three design variables $[s_1 s_2 s_3]$ and two inequality $[c_{\hat{0},(1)} c_{\hat{0},(2)}]$ constraints. Two coupled variables $[t_1\ t_2]$ participate in defining the constraints. t_1 is a function of $[s_1\ s_2\ s_3\ t_2]$ while t_2 is a function of $[s_1\ s_2\ t_1]$. Thus, there is no closed-form expression of t_1 and t_2.

$$\min_{s_1,s_2,s_3} f_{obj} = s_3^2 + s_2 + t_1 + e^{-t_2} \tag{6.7a}$$

$$\text{subject to} \tag{6.7b}$$

$$c_{\hat{0},(1)}(x) = \frac{t_2}{24} - 1 \leq 0 \tag{6.7c}$$

$$c_{\hat{0},(2)}(x) = 1 - \frac{t_1}{3.16} \leq 0 \tag{6.7d}$$

$$t_1(s_1, s_2, s_3, t_2) = s_1^2 + s_2 + s_3 - 0.2\, t_2 \tag{6.7e}$$

$$t_2(s_1, s_2, t_1) = \sqrt{t_1} + s_1 + s_2 \tag{6.7f}$$

$$[0, 0, 0] \leq [s_1, s_2, s_3] \leq [10, 10, 10] \tag{6.7g}$$

In the past, the Sellar problem has been used [25] to demonstrate the performance of a Collaborative Optimization (CO) algorithm comprised of a nested optimization with an inner and outer loop. The inner loop optimization was utilized for determining the values of the coupled parameters and the outer loop optimization was utilized for minimizing the objective function. The original Sellar problem is modified in this application in order to generate a multi-objective statement. All design variables are considered as global design variables. A multi-objective function is created and minimized at the system level optimization (similar to the outer loop of the collaborative optimization) while the values of the coupled parameters are determined at the discipline level of the bi-level structure (similar to the inner loop of the collabo-

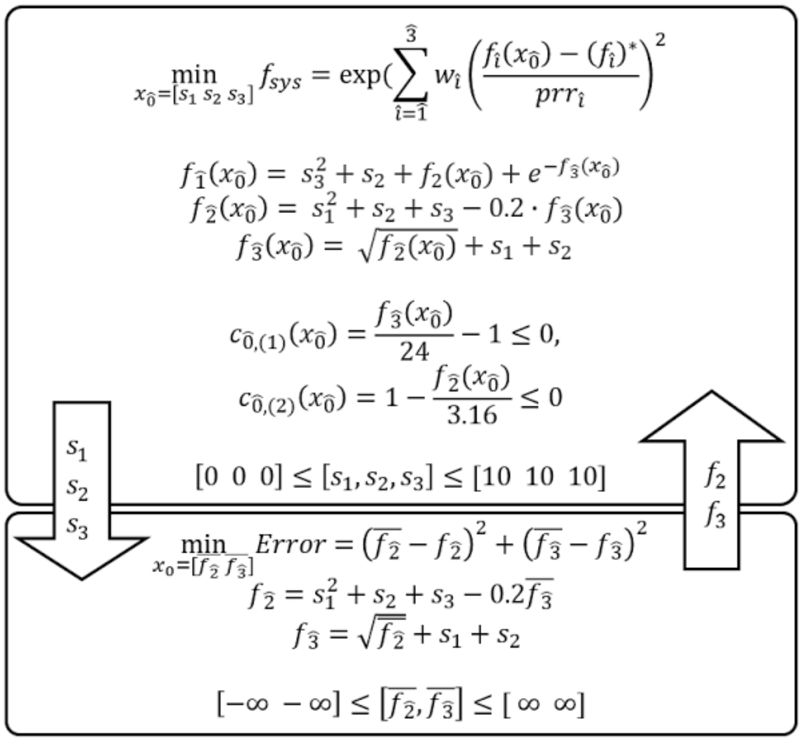
$\min_{x_{\hat{0}}=[s_1\ s_2\ s_3]} f_{sys} = \exp(\sum_{\hat{\imath}=\hat{1}}^{\hat{3}} w_{\hat{\imath}} \left(\frac{f_{\hat{\imath}}(x_{\hat{0}}) - (f_{\hat{\imath}})^*}{prr_{\hat{\imath}}}\right)^2$
$f_{\hat{1}}(x_{\hat{0}}) = s_3^2 + s_2 + f_2(x_{\hat{0}}) + e^{-f_{\hat{3}}(x_{\hat{0}})}$
$f_{\hat{2}}(x_{\hat{0}}) = s_1^2 + s_2 + s_3 - 0.2 \cdot f_{\hat{3}}(x_{\hat{0}})$
$f_{\hat{3}}(x_{\hat{0}}) = \sqrt{f_{\hat{2}}(x_{\hat{0}})} + s_1 + s_2$
$c_{\hat{0},(1)}(x_{\hat{0}}) = \frac{f_{\hat{3}}(x_{\hat{0}})}{24} - 1 \leq 0,$
$c_{\hat{0},(2)}(x_{\hat{0}}) = 1 - \frac{f_{\hat{2}}(x_{\hat{0}})}{3.16} \leq 0$
$[0\ \ 0\ \ 0] \leq [s_1, s_2, s_3] \leq [10\ \ 10\ \ 10]$
s_1
s_2
s_3
f_2
f_3
$\min_{x_0=[\overline{f_{\hat{2}}}\ \overline{f_{\hat{3}}}]} Error = (\overline{f_{\hat{2}}} - f_{\hat{2}})^2 + (\overline{f_{\hat{3}}} - f_{\hat{3}})^2$
$f_{\hat{2}} = s_1^2 + s_2 + s_3 - 0.2\overline{f_{\hat{3}}}$
$f_{\hat{3}} = \sqrt{\overline{f_{\hat{2}}}} + s_1 + s_2$
$[-\infty\ \ -\infty] \leq [\overline{f_{\hat{2}}}, \overline{f_{\hat{3}}}] \leq [\infty\ \ \infty]$

Figure 6.6: MO Sellar Problem Statement

rative optimization). In this manner, it is demonstrated how the bi-level algorithm is versatile enough to solve a typical collaborative optimization problem with coupled parameters. The multi-objective version of the Sellar problem has a small Pareto front that is challenging to identify. Thus, the utility of the multi-objective function transformation is demonstrated.

$\hat{i}$	$x_{\hat{0}}^{\hat{i}}$	$f_{\hat{i}}$	$prr_{\hat{i}}$
$\hat{1}$	[1.97, 0.00, 0.00]	3.18	12.5
$\hat{2}$	[1.78, 0.34, 0.43]	3.16	0.04
$\hat{3}$	[0.00, 0.00, 3.52]	1.78	2.12

Table 6.2: MO Sellar problem single objective optimization result

Fig. 6.6 presents the flow chart of the modified Sellar problem. At the system level the multi-objective function combines three performance metrics ($f_{\hat{1}} = f_{obj}, f_{\hat{2}} = t_1, f_{\hat{3}} = t_2$). The upper bounds ($ub$) for the design variables, and the inequality constraints ($c_{\hat{0}}(x)$) are identical to the original Sellar problem. There are three global design variables $[s_1, s_2, s_3]$, and two new local variables ($x_{\hat{1}} = [\bar{f}_{\hat{2}}, \bar{f}_{\hat{3}}]$) are introduced to compute $f_{\hat{2}}$ and $f_{\hat{3}}$. The local variables are the best estimate of $f_{\hat{2}}$ & $f_{\hat{3}}$. The discipline level optimization in the bi-level structure minimizes the $Error$ between $[\bar{f}_{\hat{2}}, \bar{f}_{\hat{3}}]$ and $[f_{\hat{2}}, f_{\hat{3}}]$ to obtain values for $[f_{\hat{2}}, f_{\hat{3}}]$. This sub-optimization loop to compute coupled variables is identical to 'collaborative optimization,' (i.e., computing $f^*_{\hat{1}}$ is identical to the original Sellar (CO) and the solution from multi-objective method is compared with a published 'collaborative optimization' to highlight the effectiveness of multi-objective transformation method.)

The utopia point and the prrs are summarized in Table 6.2. The prrs exhibit substantially different values. This indicates that the upper-lower bound transformation is important for identifying the Pareto front. An optimization analysis is conducted by assigning equal weight to all three objective functions. The optimal solution converges on the Pareto front. The latter is evaluated using a Monte Carlo simulation which evaluated 248,625 points and created a Pareto front with only 113 points. The small ratio of Pareto front points over the total number of evaluation points is an indication of the small Pareto front exhibited by this optimization problem.

Fig. 6.7 through Fig. 6.9 present three two-dimensional projections of the Pareto front along with results of the optimization analysis. The optimization results either converge on the Pareto front (Fig. 6.8) or provide an optimum that exhibits a slight improvement compared to the non-dominated solutions identified by the Monte Carlo simulation (Fig. 6.7 and Fig. 6.9).

The optimum results from OpenMDAO [25] are also included in Fig. 6.7 through Fig. 6.9 as another point of reference. OpenMDAO uses Collaborative Optimization (CO) to minimize $f_{\hat{1}}$. As expected, the published results exhibit a better performance for $f_{\hat{1}}$ but worse for $f_{\hat{3}}$ compared to the results obtained by the algorithm. A detailed summary of the results is presented in Table 6.3. The difference between the two solutions at each one of the three performance metrics is presented as a percentage of the prr.

	Published Data	Algorithm Result	Change in Value (% of prr)
$f_{\hat{1}}$	**3.18**	**5.51**	**18.6%**
$f_{\hat{2}}$	**3.16**	**3.16**	**0%**
$f_{\hat{3}}$	**3.76**	**3.48**	**-13.4%**

Table 6.3: MO Sellar Problem Result with equal weights

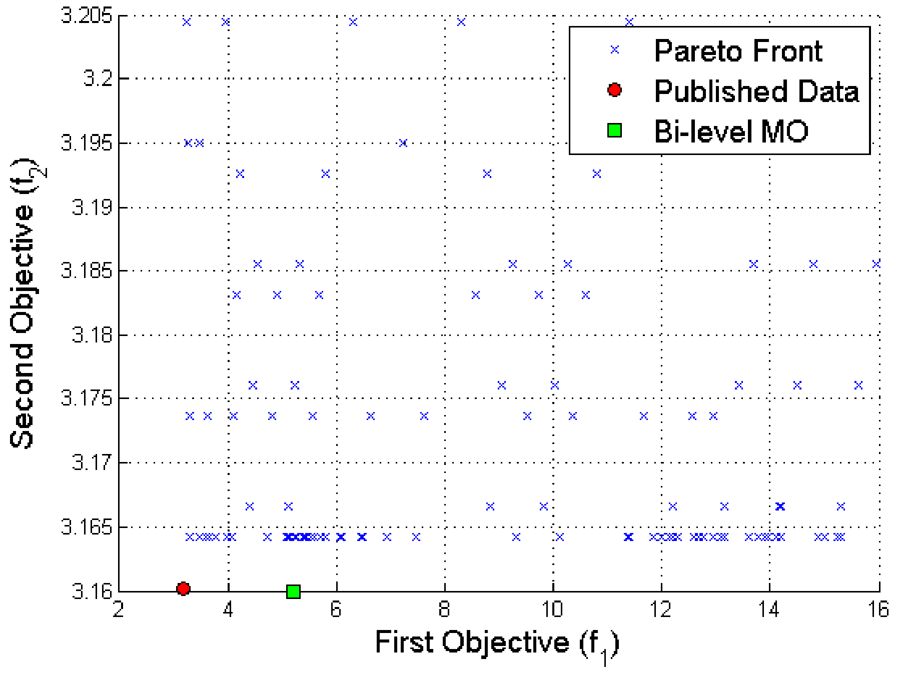

Figure 6.7: Numerical Pareto front of the Sellar problem $f_{\hat{1}}$ vs. $f_{\hat{2}}$

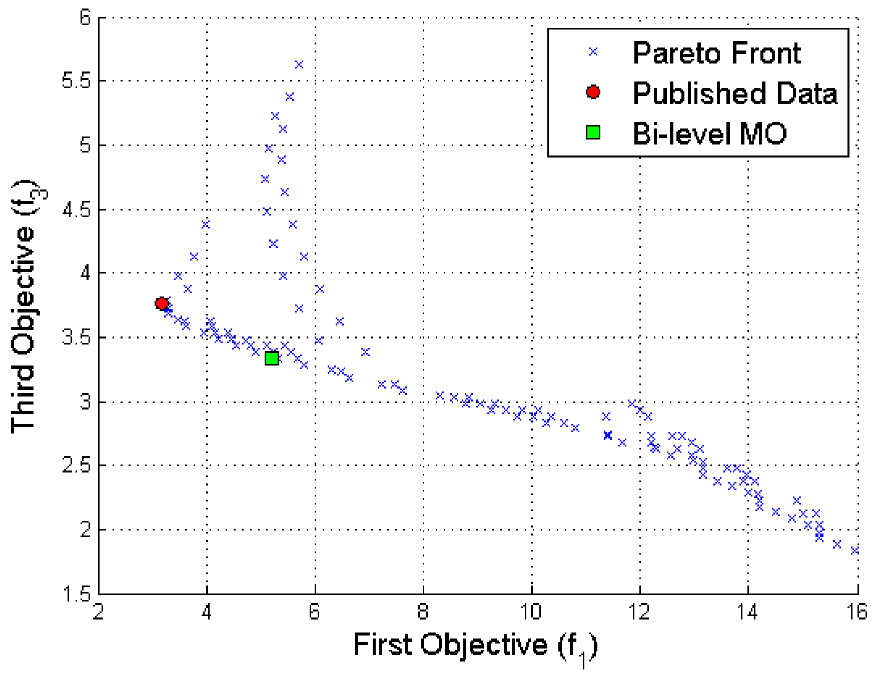

Figure 6.8: Numerical Pareto front of the Sellar problem f_1 vs. f_3

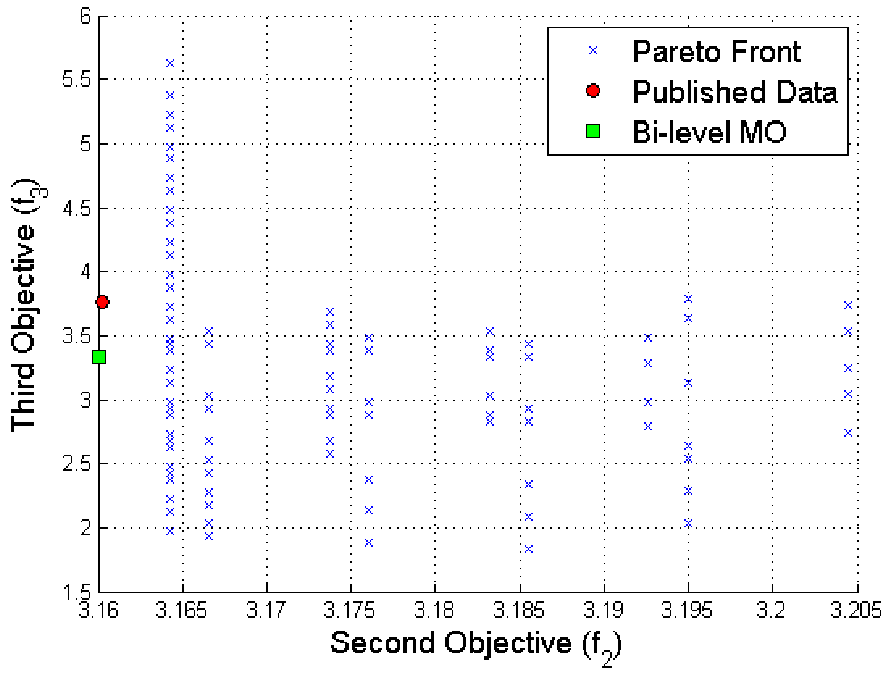

Figure 6.9: Numerical Pareto front of the Sellar problem f_2 vs. f_3

6.5 Conclusion

The structure of a bi-level optimization algorithm that identifies a single consolidated optimum solution while considering multiple disciplines with individual objectives and constraints is presented. Global and local design variables comprise the parameters which are controlled by the system level and the discipline level optimizations, respectively. Both continuous and discrete design variables can be considered. The new algorithm simultaneously optimizes multiple objectives by creating a system level multi-objective function. It minimizes the distance to the utopia point in order to find a solution on the Pareto front, and it uses an upper-lower bound approach to scale all objective functions.

Optimization problems from the literature are utilized for creating three new multi-discipline optimization statements solved by the bi-level multi-objective algorithm. In all cases, the solution converges on the Pareto front. Different weights can be used for converging at different points of the Pareto front. The latter is evaluated for each optimization using a Monte Carlo simulation.

CHAPTER VII

A modified CG method

In this chapter, modifications are introduced to improve the performance of the CG method. Then, the utility of new modifications is demonstrated by comparing it with a baseline algorithm, which is a Fletcher-Reeves CG method with exterior penalty.

7.1 Baseline algorithm

'Fletcher-Reeves' CG method ($\beta_{FR} = \|g_k\|^2/\|g_{k-1}\|^2$) is chosen as a baseline algorithm. The baseline algorithm converts constrained optimization problems to unconstrained optimization problems using 'exterior penalty method.' At each iteration, the algorithm uses the 'golden section line search' to find a step length (α_k).

7.1.1 Exterior penalty method

If constraints exist, such as box constraints ($\text{lb} \leq x \leq \text{ub}$), inequality constraints ($c_{(i)}(x) \leq 0$), penalty terms are added to the objective function to create a pseudo-objective function ($\Psi(x)$). In this way, a constrained problem is converted to an unconstrained problem. One of the simplest penalty function method is the 'exterior penalty method.' As seen in Eqn. 7.1, it adds the *square* of constraint violation to the objective function. This method has been integrated with Fletcher-Reeves nonlinear conjugate gradient method to solve large-scale constrained optimization problems [62].

$$
\begin{aligned}
\min_{x} \ \Psi(x) \quad &= f(x) + \sum_{i=1}^{n} p_{(i)} \left(\max(0, lb_{(i)} - x_{(i)})^2 + \max(0, x_{(i)} - ub_{(i)})^2 \right) \\
&+ \sum_{j=1}^{m} q_{(j)} \ \max(0, -c_{(j)}(x))^2 \qquad (7.1) \\
p_{(i)} \quad &= \text{box constraint penalty multipliers} \\
q_{(j)} \quad &= \text{constraint penalty multipliers}
\end{aligned}
$$

It minimizes a pseudo-objective function ($\Psi(x)$) to find a feasible optimal point, and it follows a procedure outlined in Algorithm 2.

Algorithm 2 Nonlinear conjugate gradient method

```
Define $\Psi(x)$
Initialize the algorithm: $x_0$ and $d_0 = -g_0$
while not converged do
    Compute $\beta_k$ using Table 4.1
    Compute $d_k = -g_k + \beta_k d_{k-1}$
    Line search $\alpha_k = arg\min_{\alpha_k > 0} f(x_k + \alpha_k d_k)$
    Update $x_{k+1} = x_k + \alpha_k d_k$
    Update $k := k + 1$
end while
return $x_{k+1}$, $\Psi(x_{k+1})$
```

To converge at a *feasible optimal* point, the optimizer needs to reduce *both* the objective function and penalty terms. If the penalty terms are much smaller than the objective function, it favors the reduction of the objective function value while ignoring feasibility. On the other hand, the optimizer may terminate prematurely at a sub-optimal point if the penalty terms become too big. Therefore, it often requires calibration of penalty multipliers ($p_{(i)}, q_{(j)}$), such that $\Psi(x)$ has a good balance of optimality and feasibility. This calibration can be challenging especially when the relation between the objective function and the penalty terms are not known.

7.2 Bounded search direction update method

CG methods update the input variables by finding a step length along the search direction ($x_{k+1} = x_k + \alpha_k d_k$). If there exist zeros in the search direction (d_k), the input variables which correspond to the zeros do not change regardless of a choice of step length (α_k). The bounded search direction update method uses the projected gradient method to insert zeros to the search direction (d_k).

Input variables that violate the upper (lower) bounds are identified at each iteration. These variables are called *box-constraints violating* variables. Then, the box-constraints violating variables are fixed to the bounds (i.e., $\{x : x \geq \text{ub}\} = \text{ub}$ and $\{x : x \leq \text{lb}\} = \text{lb}$). If the corresponding search direction is positive for $\{x : x \geq \text{ub}\}$ (negative for $\{x : x \leq \text{lb}\}$), then the search direction is set to zero.

The following terms are defined first for clarity of explanation.

ubx_k : indices of input variables such that $x_k(ubx_k) \geq \text{ub}(ubx_k)$

lbx_k : indices of input variables such that $x_k(lbx_k) \leq \text{lb}(lbx_k)$

$d_k(ubx_k)$: a vector of search direction corresponds to $x(ubx_k)$

$d_k(lbx_k)$: a vector of search direction corresponds to $x(lbx_k)$

In the algorithm, the *new steps are implemented as follows:

1. $d_k = -g_k + \beta_k d_{k-1}$

2. *Modify search directions to zero.

   note that ubx_k and lbx_k are found in step 5.

   $\{d_k : d_k(ubx_k) > 0\} = 0$

   $\{d_k : d_k(lbx_k) < 0\} = 0$

3. $\alpha_k = arg\min_{\alpha_k \geq 0} f(x_k + \alpha_k d_k)$

4. $x_{k+1} = x_k + \alpha_k d_k$

5. *Identify input variables that violate lb and ub

   $\{x_k(ubx_{k+1}) : x_{k+1} \geq \text{ub}\}$

   $\{x_k(lbx_{k+1}) : x_{k+1} \leq \text{lb}\}$

6. *Fix the values of input variables to lb and ub.

   $x_{k+1}(ubx_{k+1}) = \text{ub}(ubx_{k+1})$

   $x_{k+1}(lbx_{k+1}) = \text{lb}(lbx_{k+1})$

7. $k := k + 1$

With these new steps, the input variables are guaranteed to stay within the box constraints. Subsequently, penalty terms for the box-constraints are no longer required. The entire procedure is outlined in Algorithm 3.

7.3 Multi-variate step length method

The baseline algorithm finds a 'single step length' (α_k) for all input variables. In 'multi-variate step-length' method, the input variables are partitioned into multiple regions ($x_k = x_k^1 \cup x_k^2 \cup x_k^3 \cup x_k^4]$) according to the magnitude of corresponding search direction as shown in Eqn. 7.2.

Algorithm 3 Bounded search direction update method

```
* denotes new steps.
*1 Modify $d_k$
*2 Find $x_{k+1}$ that violate $\bar{\text{lb}}$ and $\bar{\text{ub}}$.
*3 Modify $x_{k+1}$
  Define $\Psi(x)$
  Initialize the algorithm: $x_0$ and $d_0 = -g_0$
  while not converged do
      Compute $\beta_k$ using Table 4.1
      Compute $d_k = -g_k + \beta_k d_{k-1}$
      *1a Modify $\{d_k : d_k(ubx_k) > 0\} = 0$
      *1b Modify $\{d_k : d_k(lbx_k) < 0\} = 0$
      Compute $\alpha_k = arg \min_{\alpha_k > 0} f(x_k + \alpha_k d_k)$
      Update $x_{k+1} = x_k + \alpha_k d_k$
      *2 Identify $ubx_{k+1}$ and $lbx_{k+1}$
      *3a Modify $x_{k+1}(ubx_{k+1}) = \text{ub}(ubx_{k+1})$
      *3b Modify $x_{k+1}(lbx_{k+1}) = \text{lb}(lbx_{k+1})$
      update $k := k + 1$
  end while
  return $x_{k+1}$, $\Psi(x_{k+1})$
```

$$\left\{\begin{array}{lcccc} x_k^1 : & & d_k & > & c_1 \\ x_k^2 : & c_1 \geq & d_k & > & \epsilon \\ x_k^3 : & -\epsilon > & d_k & \geq & -c_2 \\ x_k^4 : & -c_2 > & d_k & & \end{array}\right\} \text{ where } c_1, c_2 > 0 \text{ and } 1 >> \epsilon > 0$$

The ‘multi-variate steplength method’ uses vector search methods to find steplength for each region ($\bar{\alpha_k} = [\alpha_k^1,\ \alpha_k^2,\ \alpha_k^3,\ \alpha_k^4]$). Finding a vector of step length is equivalent of modifying a search direction as shown in Eqn. 7.2.

$$\left(\begin{bmatrix} x_k^1 \\ x_k^2 \\ x_k^3 \\ x_k^4 \end{bmatrix} + \begin{bmatrix} \alpha_k^1 \\ \alpha_k^2 \\ \alpha_k^3 \\ \alpha_k^4 \end{bmatrix} \begin{bmatrix} d_k^1 \\ d_k^2 \\ d_k^3 \\ d_k^4 \end{bmatrix}\right) = \left(\begin{bmatrix} x_k^1 \\ x_k^2 \\ x_k^3 \\ x_k^4 \end{bmatrix} + \hat{\alpha}_k \begin{bmatrix} \eta_k^1 d_k^1 \\ \eta_k^2 d_k^2 \\ \eta_k^3 d_k^3 \\ \eta_k^4 d_k^4 \end{bmatrix}\right) \tag{7.2}$$
$$\text{where } [\alpha_k^1 \cdots \alpha_k^4]^\top = \hat{\alpha}_k [\eta_k^1 \cdots \eta_k^4]^\top$$

To find a vector of step lengths, two multi-variate step length methods, *sequential* and *simultaneous* methods, are proposed. The *sequential method* solves a line search optimization for each region *sequentially* as shown in Eqn. 7.3.

$$\min_{0 \le \alpha_k^j \le \alpha_{max}} \quad \Psi \quad (x_k + \alpha_k^j d_k^j) \quad \text{for all} \quad j = 1, \cdots, n \tag{7.3a}$$
$$\bar{\alpha_k} = [\alpha_k^1, \cdots, \alpha_k^n], \quad x_{k+1} = x_k + \bar{\alpha_k} d_k \tag{7.3b}$$

The *simultaneous method* uses the ‘bounded Nealder-Mead simplex method’ to find a *vector* of step length. The bounded Nelder-Mead simplex method is a heuristic vector search method. It expands and contracts the size a simplex in the search space to find the minimum within the search bound ($0 \le \alpha_k \le \alpha^{max}$). The lower bound ($0 \le \alpha_k$) is required because a negative step implies that it searches the opposite of the search direction. The upper bound is also required because the algorithm diverges when the search area becomes too large.

The original Nelder-Mead simplex method imposes no bounds. Thus, the variable transformation method is required to ensure that the Nelder-Mead simplex method stays within the bounds. The bounds ($0 \le \alpha_k^j \le \alpha_{max}$) are implemented using a variable transformation method shown in Eqn. 7.4. $\bar{\alpha_k}$ has a maximum value of α_{max}, and a minimum value of zero. [15; 36]. The variables are transformed as shown in

Eqn. 7.4.

$$\bar{\alpha_k} = \alpha_{max} \left(\frac{sin(\bar{z_k}) + 1}{2} \right) \tag{7.4}$$

The Nealder-Mead simplex method solves for $\bar{z_k}$ as shown in Eqn. 7.5.

$$\bar{z_k} = arg \min_{z_k} f(x_k + \bar{\alpha_k}(\bar{z_k}) d_k) \tag{7.5}$$

The procedure of the multi-variate steplength method is outlined in Algorithm 4.

Algorithm 4 Multi-variate steplength method

```
* denotes new steps.
*1 Partition $x_k$ and $d_k$
*2 Find $\bar{\alpha_k}$

  Define $\Psi(x)$
  Initialize the algorithm: $x_0$ and $d_0 = -g_0$
  **while** not converged **do**
      Compute $\beta_k$ using Table 4.1
      Compute $d_k = -g_k + \beta_k d_{k-1}$
      *1 Partition the input variable
      *2a Perform golden sectioin search sequentially
      **for** i=1:n **do**
          Find $\alpha_k^i = arg \min_{0 \leq \alpha_k^i \leq \alpha_{max}} f(x_k + \alpha_k^i d_k)$
      **end for**
      *2b Perform Bounded Nelder-Mead search
      Find $\bar{\alpha_k} = \alpha_{max} \frac{sin(\bar{z_k}) + 1}{2}$ ,  $\bar{z_k} = arg \min_{\bar{z_k}} f(x_k + \bar{\alpha_k}(\bar{z_k}) d_k)$
      Update $x_{k+1} = x_k + \bar{\alpha_k} d_k$
      Update $k = k + 1$
  **end while**
  **return** $x_{k+1}$, $\Psi(x_{k+1})$
```

7.4 Numerical examples

The utility of the new elements are evaluated by solving three constrained problems with varying degrees of difficulties. The problem descriptions and optimization results are presented in this section. Each problem is solved in multiple cases to compare and to contrast with the ‘baseline’ algorithm. All problems exhibit a large number of design variables.

The termination criteria are

1. maximum iteration $k_{max} = 100$
2. $||g_k|| \leq 10^{-6}$
3. $||d_k|| \leq 10^{-6}$
4. $|\psi_{k+1} - \psi_k| \leq 10^{-6}$

   where $|| \cdot ||$ is the maximum absolute component of a vector.

The penalty terms multipliers are fixed at constant, and the box-constraints penalty terms are removed when solving the problems with the bounded search direction update method. In the ‘multi-variate steplength method’, c_1 and c_2 are equal to 0.4 while ϵ is fixed at 10^{-6}.

The objective function values are non-dimensionalized. ‘1’ is equal to $\Psi(x_0)$ whereas ‘0’ is equal to the minimum. The *unbounded single steplength* method is the ‘baseline’ algorithm, and the following abbreviations are used

1. UNB = Unbounded (with box constraints penalty terms)
2. BND = Bounded search direction update method
3. Seq. = Sequential multi-variate steplength
4. Sim. = Simultaneous multi-variate steplength

All codes are written in MATLAB and run with a computer with the following spec: Intel CoreTM i7-3770 CPU @ 3.40 GHz (8 CPUs) with 16GB of RAM.

7.4.1 500 sine problem

The objective function of the problem is comprised of 500 sine functions. It has 500 bounded design variables ($0 \leq x \leq 2\pi$) with no equality and inequality constraints. At the global optimum, 158 design variables are equal to 0 while 179 design variables are equal to 2π.

The objective function is

$$f(x) = \sum_{i=1}^{500} A_{(i)}\ sin(\omega_{(i)} x_{(i)} + b_{(i)}) \tag{7.6}$$

$$0 \leq x_{(i)} \leq 2\pi \quad \text{for all} \quad i = 1, \cdots, 500$$

The pseudo-objective function with the penalty terms is

$$\Psi(x) = f(x) + \sum_{i=1}^{500} p_{(i)}\ (\max(0, -x_{(i)})^2 + \max(0, x_{(i)} - 2\pi)^2) \tag{7.7}$$

where $p_{(i)}$ is the box-constraints penalty multiplier for $x_{(i)}$.

As seen in Fig. 7.1, the rate of convergence improves with the bounded search direction update method. It converges to lower $\Psi(x)$ than the unbounded method with fewer number of iteration. This improvement can be attributed to the fact that the optimal solution contains many variables on the bounds.

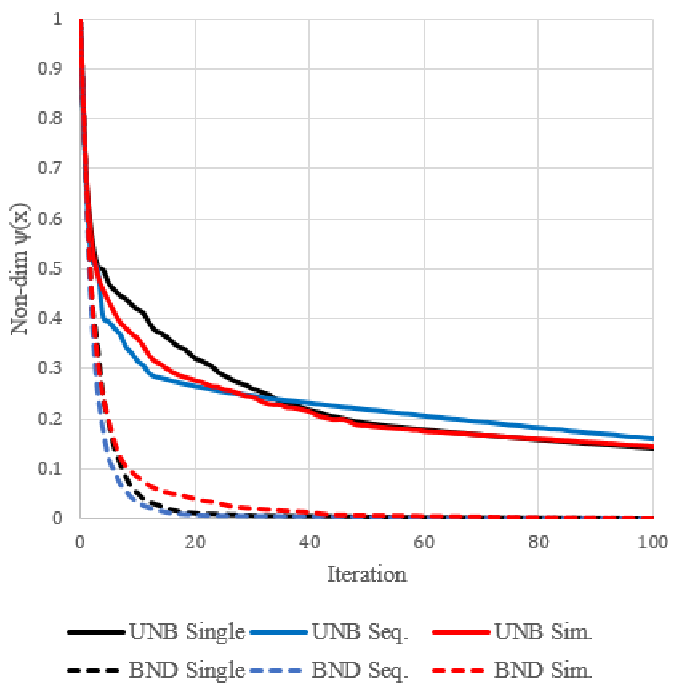

Figure 7.1: 500 Sine result $\Psi(x)$ vs. iteration

As seen in Table 7.1, the bounded method eliminates the box-constraints violation, and it reduces the number of function evaluations, the number of iterations, and the computational time.

Table 7.1: 500 sine optimization result

	unbounded			*bounded*		
	Single	Seq.	Sim.	Single	Seq.	Sim.
Non-dim $\Psi(x)$ value	0.142	0.159	0.144	$6.22 \cdot 10^{-5}$	0	$3.66 \cdot 10^{-4}$
Box constraints violation	0.461	0.913	0.413	0	0	0
Number of $\Psi(x)$ evaluation	1864	2796	2401	1836	2577	2338
Time (sec)	1.96	3.24	2.71	1.87	2.89	2.56
Number of Iteration where non-dim $\Psi(x) < 0.1$	> 100	> 100	> 100	8	6	9

The 'multi-variate method' does not have a significant effect on the 500 sine problem. The solutions obtained by *single* , *sequential*, and *simultaneous* methods are similar. The 500 sine problem is well-scaled for the single step length method. When a problem is well-scaled for a single step length method, the multi-variate step length methods show no merit.

7.4.2 Electrons on Sphere (EOS)

EOS problem finds the lowest potential energy configuration of n_p point charges on a conducting sphere [16] as seen in Fig. 7.2.

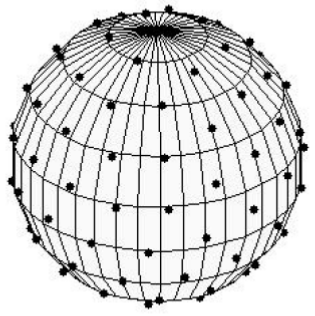

Figure 7.2: Optimal distribution of electrons, $n_p = 100$ [16]

The design variables are the location of each n_p point charge in rectilinear coordinate system $\bar{x} = [x_{(i)}\ y_{(i)}\ z_{(i)}]$. Given n_p point charges, $3n_p$ design variables exist. Every point charges are on the surface of a unit sphere (i.e, $x_{(i)}^2 + y_{(i)}^2 + z_{(i)}^2 = 1$), and the box constraints for each design variables is $-1 \leq x_{(i)},\ y_{(i)},\ z_{(i)} \leq 1$. This problem has been used in one of the benchmark study to evaluate the utility of various optimization algorithms [16].

The potential energy for n_p points $(x_{(i)}, y_{(i)}, z_{(i)})$ is defined as

$$f(x, y, z) = \sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} \left((x_{(i)} - x_{(j)})^2 + (y_{(i)} - y_{(j)})^2 + (z_{(i)} - z_{(j)})^2\right)^{-\frac{1}{2}} \tag{7.8}$$

The constraints on the n_p points are

$$\begin{aligned} x_{(i)}^2 + y_{(i)}^2 + z_{(i)}^2 &= 1 \qquad \text{for all} \quad i = 1, \cdots, n_p && \text{(7.9a)} \\ -1 \leq x_{(i)},\ y_{(i)},\ z_{(i)} &\leq 1 \qquad \text{for all} \quad i = 1, \cdots, n_p && \text{(7.9b)} \end{aligned}$$

The pseudo-objective function with the penalty terms is

$$\begin{aligned}\Psi(x,y,z) &= f(x,y,z)+\sum_{i=1}^{n_p} r_{(i)}(x_{(i)}^2+y_{(i)}^2+z_{(i)}^2-1)^2 \qquad (7.10)\\ &+ \sum_{i=1}^{n_p} p_{x_{(i)}}(\max(0,x_{(i)}-1)^2+\max(0,-x_{(i)}-1)^2)\\ &+ \sum_{i=1}^{n_p} p_{y_{(i)}}(\max(0,y_{(i)}-1)^2+\max(0,-y_{(i)}-1)^2)\\ &+ \sum_{i=1}^{n_p} p_{z_{(i)}}(\max(0,z_{(i)}-1)^2+\max(0,-z_{(i)}-1)^2)\end{aligned}$$

where $r_{(i)}$ is the equality constraint penalty multiplier, and $p_{x_{(i)}}, p_{y_{(i)}}, p_{z_{(i)}}$ are the box constraint penalty multipliers.

The unbound method and the bounded method produce *identical* sequences of x_k as seen in Fig. 7.3. The equality constraints ($x_{(i)}^2+y_{(i)}^2+z_{(i)}^2=1$) contains $x_{(i)}, y_{(i)}, z_{(i)}$ within the box constraints ($-1 \leq x_{(i)}, y_{(i)}, z_{(i)} \leq 1$). Thus, the bounded method has no effect on the result.

The ‘multi-variate step length method’ performs better than the baseline algorithm. The EOS problem is poorly scaled for the single-step length line search method. The magnitude of the search direction (d_k) is on the order of 10^5, and the line search method finds a small step length that is on order of 10^{-10}. As a result, the single step length method stalls at a sub-optimal point. Both multi-variate step length methods circumvent the scaling issue, and the optimization process continues beyond the sub-optimal point that the single step length method stalls at.

As seen in Table 7.2, both multi-variate step length methods require more numbers of function evaluation than the single step length method. Especially, the *simultaneous* method requires 6 times more function evaluations than the single step length method. However, the *simultaneous* method requires the fewest number of iterations

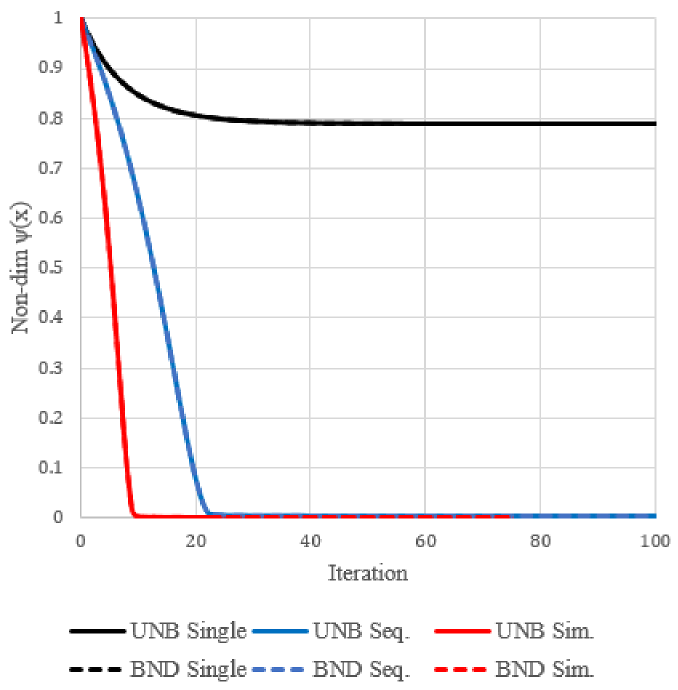

Figure 7.3: EOS result $\Psi(x)$ vs. iteration, $n_p = 1000$

to converge at the minimum. Explicit gradient equations for the pseudo-objective function is available for this problem. Thus, computing the gradient at every iteration is relatively inexpensive. However, when the gradient has to be computed using numerical methods, such as finite difference, reducing the number of iteration can reduce the computational cost.

Table 7.2: EOS optimization result

	Single	Seq.	Sim.
Non-dim $\Psi(x)$ value	0.789	$2.48 \cdot 10^{-03}$	0
Box constraints violation	0	0	0
Number of $\Psi(x)$ evaluation	302	342	1885
Time (sec)	76.3	80.1	187
Number of Iteration where non-dim $\Psi(x) < 0.1$	> 100	22	9

7.4.3 Cantilevered Beam

The cantilevered beam depicted in Fig. 7.4 and Fig. 7.5 is designed for minimum material volume. The design variables are the width, $b_{(i)}$, and height, $h_{(i)}$, of each of the N segments.

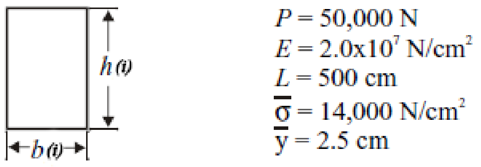

Figure 7.4: Cantilevered beam cross section [62]

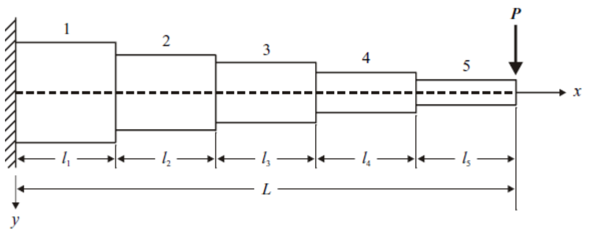

Figure 7.5: Cantilevered beam [62]

Given N segments, there are 2N design variables. The beam is subject to stress limits ($\bar{\sigma}$) at the left end of each segment. The problem solves for a minimal volume of the beam with 4 constraints.

1. The bending stress ($\sigma_{(i)}$) has to be less than or equal to the maximum bending stress ($\bar{\sigma} = 14,000$ N/cm^2).
2. The height of a beam section ($h_{(i)}$) cannot be greater than 20 times of the width ($b_{(i)}$) of the beam section.
3. The width of a beam section has to be greater than 3 cm.
4. The height of a beam section has to be greater than 10 cm.

The height of any segment does not exceed twenty times of the width ($h_{(i)} \leq 20b_{(i)}$) [62], and a complete optimization statement is defined as follows:

$$V(b,h) = \sum_{i=1}^{N} b_{(i)} h_{(i)} l_{(i)} \tag{7.11a}$$

subject to

$$\frac{\sigma_{(i)}}{\bar{\sigma}} - 1 \leq 0 \qquad i = 1, \cdots, N \tag{7.11b}$$

$$h_{(i)} - 20b_{(i)} \leq 0 \qquad i = 1, \cdots, N \tag{7.11c}$$

$$b_{(i)} \geq 3 \qquad i = 1, \cdots, N \tag{7.11d}$$

$$h_{(i)} \geq 10 \qquad i = 1, \cdots, N \tag{7.11e}$$

The pseudo-objective function with the penalty terms is

$$\begin{aligned} \Psi(b,h) &= V(b,h) \\ &+ \sum_{i=1}^{N} p_{b_{(i)}} (\max(0, -b_{(i)} + 3)^2) + \sum_{i=1}^{N} p_{h_{(i)}} (\max(0, -h_{(i)} + 10)^2) \\ &+ \sum_{i=1}^{N} q_{\alpha_{(i)}} (\max(0, \frac{\sigma_{(i)}}{\bar{\sigma}} - 1)^2) + \sum_{i=1}^{N} q_{bh_{(i)}} (\max(0, h_{(i)} - 20b_{(i)})^2) \end{aligned} \tag{7.12}$$

$p_{b_{(i)}}, p_{h_{(i)}}$ are the box constraint penalty multipliers, and $q_{\alpha_{(i)}}, q_{bh_{(i)}}$ are the inequality constraint penalty multipliers. The bending moment at the left end of segment ($M_{(i)}$), the corresponding maximum bending stress ($\sigma_{(i)}$), and the second moment of area ($I_{(i)}$) are computed as shown in Eqn. 7.13.

$$M_{(i)} = P\left[L + l_{(i)} - \sum_{j=1}^{i} l_{(j)}\right] \tag{7.13a}$$

$$\sigma_{(i)} = \frac{M_{(i)} h_{(i)}}{2I_{(i)}} \tag{7.13b}$$

$$I_{(i)} = \frac{b_{(i)} h_{(i)}^3}{12} \tag{7.13c}$$

The cantilevered problem is poorly scaled for the single steplength method. The magnitude of the search direction (d_k) is on order of 10^3, and the line search finds a small steplength (α_k) that is on order of 10^{-5}. As a result, single steplength method shows unsatisfactory rate of convergence as seen in Fig. 7.6.

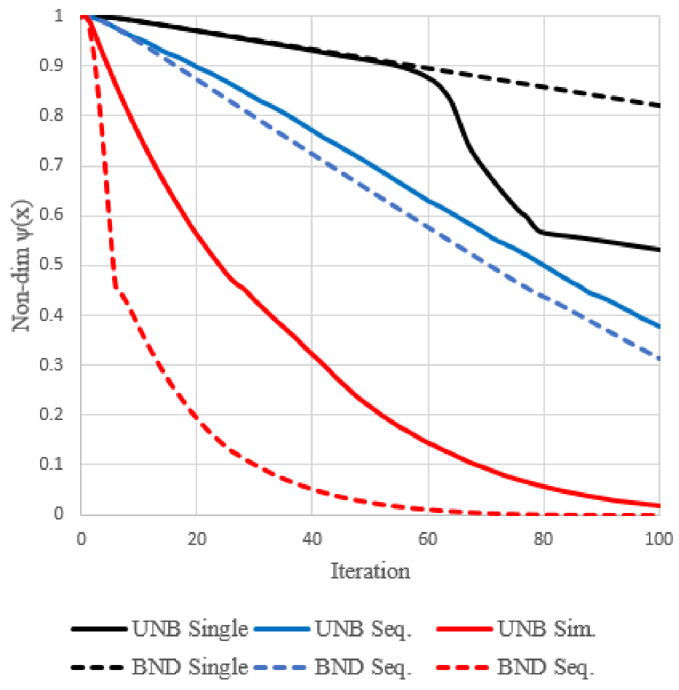

Figure 7.6: Cantilevered result $\Psi(x)$ vs. iteration, $N = 2000$

Although the 'multi-variate steplength method' increases the the number of function evaluation as shown in Table 7.3, it mitigates the scaling issue and converge at lower $\Psi(x)$ value with fewer number of iterations. Reducing the number of iteration is advantageous when computing the gradient of $\Psi(x)$ is expensive.

The 'bounded method' is shown useful in this problem. It eliminates the box constraints violation, and it reduces the number of function evaluations. When the bounded method is coupled with the *simultaneous* method, it converges to the minimum with the fewest number of iteration.

Table 7.3: Cantilevered beam optimization result

	unbounded			*bounded*		
	Single	Seq.	Sim.	Single	Seq.	Sim.
Non-dim $\Psi(x)$ value	0.532	0.379	$1.93 \cdot 10^{-2}$	0.819	0.313	0
Box constraints violation	$2.00 \cdot 10^{-8}$	$4.30 \cdot 10^{-4}$	$4.55 \cdot 10^{-5}$	0	0	0
Number of $\Psi(x)$ evaluation	1844	4889	5335	1882	2428	5347
Time (sec)	18.2	32.9	35.6	19.2	22.6	37.6
Number of Iteration where non-dim $\Psi(x) < 0.1$	> 100	> 100	69	> 100	> 100	31

7.5 Conclusion

The ‘bounded search direction update method’ and the ‘multi-variate step length method’ are introduced to improve the performance of CG method. Three constrained optimization problems are solved to evaluate effectiveness of the new methods. For each problem, algorithms with new methods are compared with the ‘baseline algorithm,’ which is Fletcher-Reeves CG method with exterior penalty terms.

The ‘bounded search direction update method’ inserts zeros to the search direction to contain the input variable within the box-constraints. It eliminates the box-constraints penalty terms and improves the rate of convergence when the optimal solutions contains many variables on the bounds.

The ‘multi-variate step length method’ partitions the input variables into multiple regions to mitigate scaling issues observed in CG method. The sequential method solves the line search algorithm for each region sequentially while the simultaneous method finds a vector of step length using the bounded Nelder-Mead simplex method. The multi-variate step length method outperforms the single step length method when the problem is poorly scaled for the single step length method. Although it requires more numbers of function evaluation, it converges at a lower $\Psi(x)$ with fewer numbers of iterations.

CHAPTER VIII

Solving a bi-level multi-objective optimization with a large number of design variables

In this chapter, the bi-level multi-objective algorithm is coupled with the bounded nonlinear CG methods for handling a large number of design variables. Two analytical problems are presented for demonstrating new capability. A simple multi-objective box-constrained optimization is solved first to demonstrate that the algorithm can handle large-scale box-constraints problems. Then, a multi-objective example is generated based on the static analysis of a beam. The detail problem description and the optimization results are presented and are discussed. All codes are written in MATLAB and are executed with a computer with the following specs: Intel i7-3770 CPU @ 3.40 GHz (8 CPUs) with 16GB of RAM. The convergence criteria that are used in all numerical optimization simulations are

1. maximum iteration $k_{max} = 100$
2. $||g_k|| \leq 10^{-6}$
3. $||d_k|| \leq 10^{-6}$
4. $|\psi_{k+1} - \psi_k| \leq 10^{-6}$

   where $||\cdot||$ is the maximum absolute component of a vector.

8.1 Multi-objective problem based on analytical functions

Three functions which have been utilized in the literature [70] for testing box-constrained optimization algorithms are combined for generating single multi-objective optimization problem. The three functions $(f_{\hat{1}}, f_{\hat{2}}, f_{\hat{3}})$ are defined in Eqn. 8.1. Each function is considered as the objective function of an optimization statement. Three disciplines are formed; they share global design variables $(x_{\hat{0}})$, and each has its own local design variables $(x_{\hat{1}}, x_{\hat{2}}, x_{\hat{3}})$. The numerical optimization is performed with $N = 500$, resulting in total 2000 design variables.

$$f_{\hat{1}}(x_{\hat{0}}, x_{\hat{1}}) = \sum_{j=1}^{N} 4(x_{\hat{0},(j)} - 5)^2 + (x_{\hat{1},(j)} - 6)^2 \tag{8.1a}$$

$$f_{\hat{2}}(x_{\hat{0}}, x_{\hat{2}}) = \sum_{j=1}^{N} (x_{\hat{2}.(j)} - x^2_{\hat{0}.(j)})^2 + 10\ (1 - x_{\hat{0}.(j)})^2 \tag{8.1b}$$

$$f_{\hat{3}}(x_{\hat{0}}, x_{\hat{3}}) = \sum_{j=1}^{N} 1000(x_{\hat{3},(j)} - x^2_{\hat{0},(j)})^2 + (2 - x_{\hat{0},(j)})^2 \tag{8.1c}$$

$$-1 \leq x_{\hat{0}}, x_{\hat{1}}, x_{\hat{2}}, x_{\hat{3}} \leq 11 \quad \forall \quad j = 1, \cdots, N \tag{8.1d}$$

8.1.1 Multi-objective formulation

The optimal solution of each objective is computed first to obtain $f^*_{\hat{i}}, Z_{\hat{i}}, prr_{\hat{i}}$, which are required to construct a system level objective function (f_{sys}). The numerical values of $f^*_{\hat{i}}, Z_{\hat{i}}, prr_{\hat{i}}$ are summarized in Table 8.1. The third objective $(f_{\hat{3}})$ is on the greatest order of magnitude among three objectives Without proper scaling of each objective, the algorithm would favor reduction in $f_{\hat{3}}$. $Z_{\hat{i}}$ and $prr_{\hat{i}}$ are identical in this problem because the optimal function values $(f_{\hat{i}})$ are zero. $(prr_{\hat{i}} = Z_{\hat{i}} - f_{\hat{i}})$.

$\hat{i}$	$f^*_{\hat{i}}$	$(x^{\hat{i}}_0)^*$	$(x^{\hat{i}}_{\hat{i}})^*$	$Z_{\hat{i}} = prr_{\hat{i}}$
1	0	$5 \,\forall\, j = 1 \cdots N$	$6 \,\forall\, j = 1 \cdots N$	32,000
2	0	$1 \,\forall\, j = 1 \cdots N$	$1 \,\forall\, j = 1 \cdots N$	192,500
3	0	$2 \,\forall\, j = 1 \cdots N$	$4 \,\forall\, j = 1 \cdots N$	112,504,500

Table 8.1: Single objective optimization result, $N = 500$

Once the numerical values of $f^*_{\hat{i}}, Z_{\hat{i}}, prr_{\hat{i}}$ are obtained, f_{sys} is constructed as shown in Eqn. 8.2.

$$f_{sys} = exp\left(\sum_{\hat{i}=\hat{1}}^{\hat{3}} w_{\hat{i}} \left(\frac{f_{\hat{i}}(x_0, x_{\hat{i}}) - f^*_{\hat{i}}}{prr_{\hat{i}}}\right)^2\right) \tag{8.2}$$

A complete system level optimization statement is defined as follows:

$$\min_{x_{\hat{0}}} \quad f_{sys}(x_{\hat{0}}, x_{\hat{1}}, x_{\hat{2}}, x_{\hat{3}}) \tag{8.3a}$$

$$\text{subject to}$$

$$-1 \leq x_{\hat{0}} \leq 11 \tag{8.3b}$$

$$x_{\hat{1}}, x_{\hat{2}}, x_{\hat{3}} = \text{constant}$$

The discipline level optimization statement for each $\hat{i} = \{1, 2, 3\}$ is defined as follows:

$$\min_{x_{\hat{i}} \in \mathbb{R}^N} \quad f_{\hat{i}}(x_{\hat{0}}, x_{\hat{i}}) \tag{8.4a}$$

$$\text{subject to}$$

$$-1 \leq x_{\hat{i}} \leq 11 \tag{8.4b}$$

$$x_{\hat{0}} = \text{constant}$$

Once the system level optimization is initialized, the algorithm iterates the system and the discipline level optimizations until system level converges.

8.1.2 Optimization result

The multi-objective function (f_{sys}) is solved multiple times with varying combinations of weights ($w_{\hat{1}}, w_{\hat{2}}, w_{\hat{3}}$). The weights assign relative importance among the objectives. Total 171 weights are used, and the algorithm finds 170 Pareto optimal points. The Pareto front, identified by the algorithms, is plotted in Fig. 8.1 through Fig. 8.4.

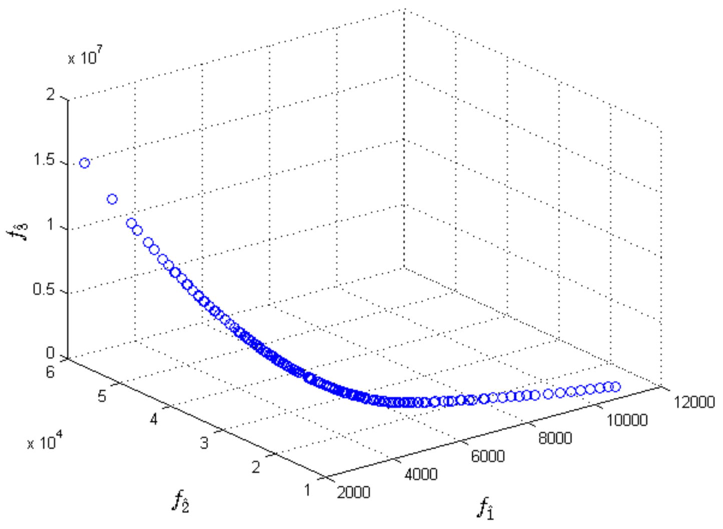

Figure 8.1: 3D scatter plot of the Pareto front

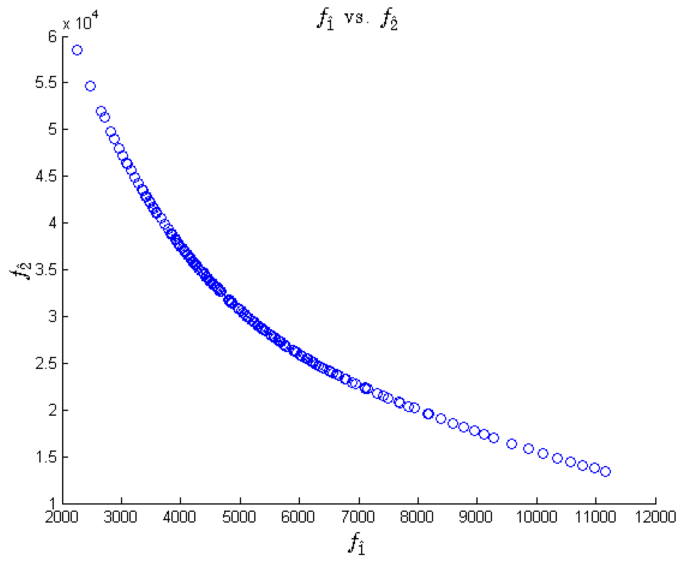

Figure 8.2: 2D projection of the Pareto front, $f_{\hat{1}}$ vs. $f_{\hat{2}}$

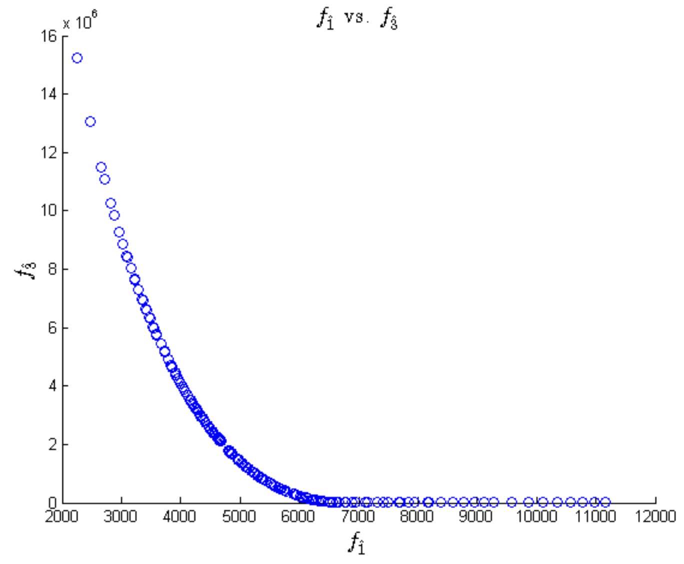

Figure 8.3: 2D projection of the Pareto front, $f_{\hat{1}}$ vs. $f_{\hat{3}}$

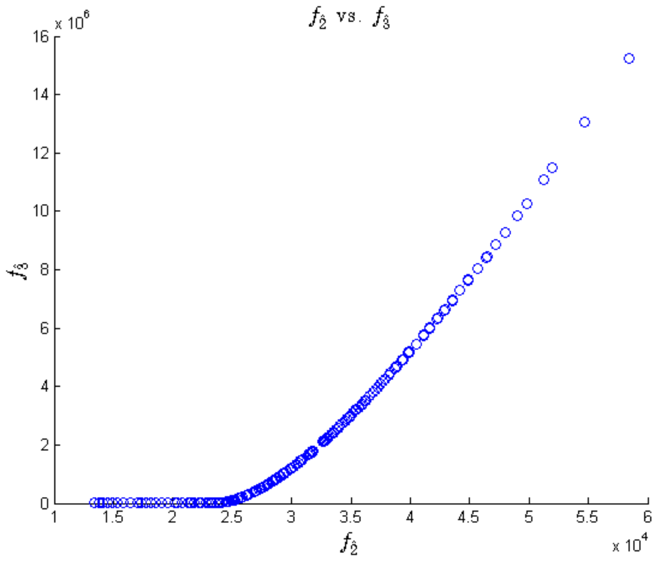

Figure 8.4: 2D projection of the Pareto front, $f_{\hat{2}}$ vs. $f_{\hat{3}}$

Even though three objectives are on different orders of magnitudes, the bi-level multi-objective algorithms constructs a Pareto front rapidly. This implies that the 'upper-lower bound' multiple objective transformation method is effective for this problem. 171 system level objective optimizations lasts 299.2 seconds with a mean computation time of 1.75 seconds and with a standard deviation of 5.07 seconds. More information about the computation time is summarized in Table 8.2.

Sum	Mean	Stdev	Min	-
2.99E+02	1.75E+00	5.07E+00	4.51E-01	sec.
1^{st} quartile	2^{nd} quartile	3^{rd} quartile	Max	-
6.43E-01	9.35E-01	1.23E+00	3.73E+01	sec.

Table 8.2: Computation time of three-objective problem

8.2 A multi-objective problem based on a beam static analysis

The cantilevered problem presented in Chapter VIII is modified for multiple loading conditions. Each loading represents a different operation requirements and comprises a separate discipline. The two loading conditions are: a cantilever beam with a point load in the middle and a simply supported beam with distributed load across the beam. Moreover, the constraints of the problem are more restrictive. The bending stress safety factor is increased to 3 as opposed to 1, and the height of the beam section cannot be greater than 4 times of the width. Thus, the constraints of the multi-objective beams are:

1. The bending stress ($\sigma_{(i)}$) has to be less than or equal to 1/3 of the maximum bending stress ($\bar{\sigma} = 14$ kN/cm^2).
2. The height of a beam section ($h_{(i)}$) cannot be greater than 4 times of width ($b_{(i)}$) of the beam section.
3. The width of a beam section has to be greater than 3 cm.
4. The height of a beam section has to be greater than 10 cm.

8.2.1 Multiple loading conditions

A multi-objective problem is composed for designing a beam that is optimized for multiple loading conditions such that a balanced solution is achieved. Two loading conditions are considered:

1. Load $P = 100$ kN applied in the middle of **cantilevered** beam at ($x = \frac{L}{2}$).

$$M_{\hat{1}}(x) = \begin{cases} P(\frac{1}{2}L - x) & \text{if } 0 \leq x \leq \frac{L}{2} \\ 0 & \text{if } \frac{L}{2} \leq x \leq L \end{cases} \tag{8.5}$$

The equation for the maximum stress at each cross section is:

$$\sigma_{\hat{1}(i)} = \frac{M_{\hat{1}}(x)h_{(i)}}{2I_{(i)}}, \quad \text{where} \quad I_{(i)} = \frac{b_{(i)}h_{(i)}^3}{12} \tag{8.6}$$

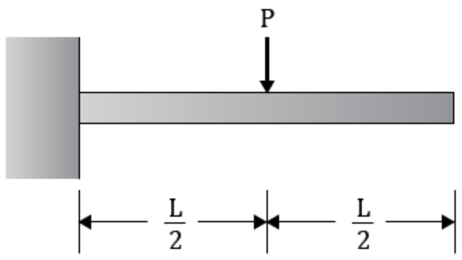

Figure 8.5: Loading Condition 1, Cantilevered Beam

2. Linearly increasing distributed load $w_0 = 0$ and $w_L = \frac{2P}{L}$, where $P = 350$ kN on **simply supported** beam.

$$M_{\hat{2}}(x) = \frac{P}{3L^2}(x^3 - L^2 x) \tag{8.7}$$

The equation for the maximum stress at each cross section is:

$$\sigma_{\hat{2}(i)} = \frac{M_{\hat{2}}(x) h_{(i)}}{2I_{(i)}}, \quad \text{where} \quad I_{(i)} = \frac{b_{(i)} h_{(i)}^3}{12} \tag{8.8}$$

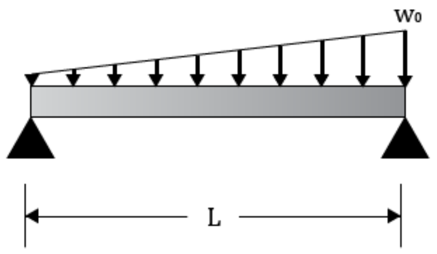

Figure 8.6: Loading Condition 2, Simply Supported Beam

The moment distribution of two loading conditions can be seen in Fig. 8.7. The first half of the beam design is subjected to bending moments from both loading conditions ($M_{\hat{1}}, M_{\hat{2}}$). The second half of the beam is subjected to bending moments from only one loading condition ($M_{\hat{2}}$).

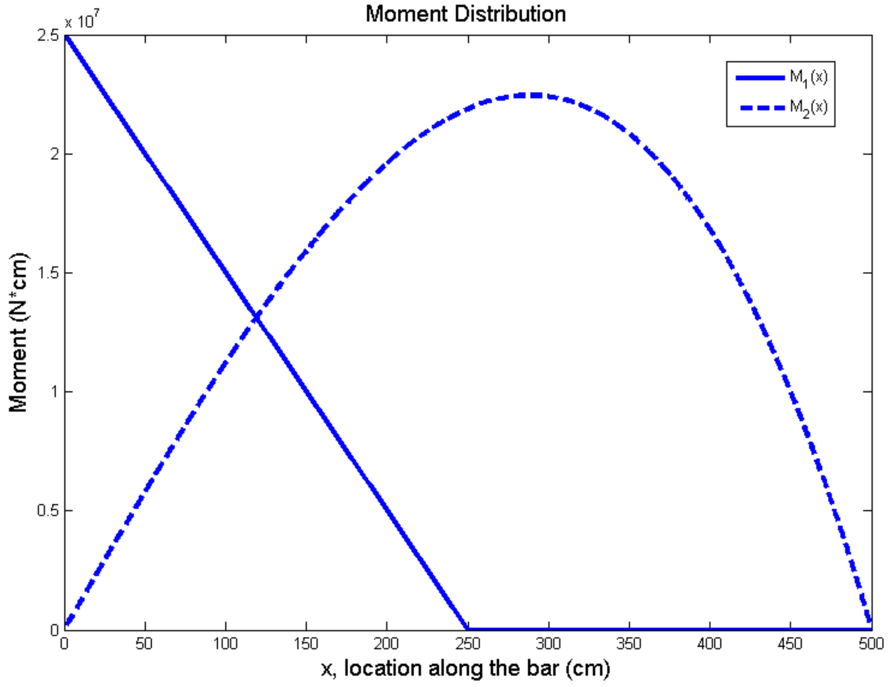

Figure 8.7: Bending moment distribution at different loading conditions

In a multi-objective formulation, the design variables which correspond to the first half of the beam are global design variables ($x_{\hat{0}} = \{b_{(i)} \ \cup \ h_{(i)} : x \leq 250 \text{ cm}\}$), and the design variables corresponding to the second of half of the beam are local design variables ($x_{\hat{1},\hat{2}} = \{b_{(i)} \ \cup \ h_{(i)} : x > 250 \text{ cm}\}$).

8.2.2 Multi-objective formulation

To construct a multi-objective objective function, the optimal solutions for each loading conditions $\left[(x_{\hat{0}}^1, x_{\hat{1}}^*)\ (x_{\hat{0}}^2, x_{\hat{2}}^*)\right]$ are computed first. The solution for loading condition 1 is obtained by solving the following optimization statements.

$$\begin{aligned}\min_{b,h} \Psi_{\hat{1}} &= V(b,h) \\ &+ \sum_{i=1}^{N} q_{\hat{1}\alpha_{(i)}} \cdot \max\ (0, \frac{3\sigma_{\hat{1},(i)}}{\bar{\sigma}} - 1)^2 \\ &+ \sum_{i=1}^{N} q_{bh_{(i)}} \cdot \max(0, h_{(i)} - 4b_{(i)})^2 \qquad (8.9a)\end{aligned}$$

subject to

$$3 \leq b_{(i)} \qquad i = 1, \cdots, N \qquad (8.9b)$$

$$10 \leq h_{(i)} \qquad i = 1, \cdots, N \qquad (8.9c)$$

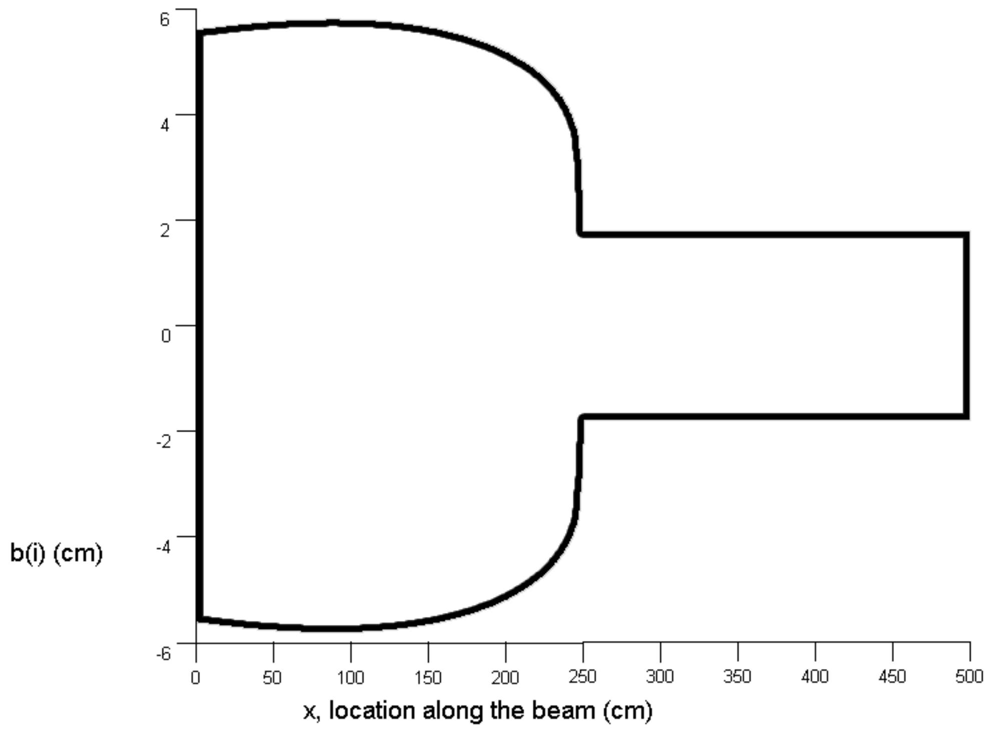

Figure 8.8: $b_{(i)}$ of optimal beam design under loading condition 1

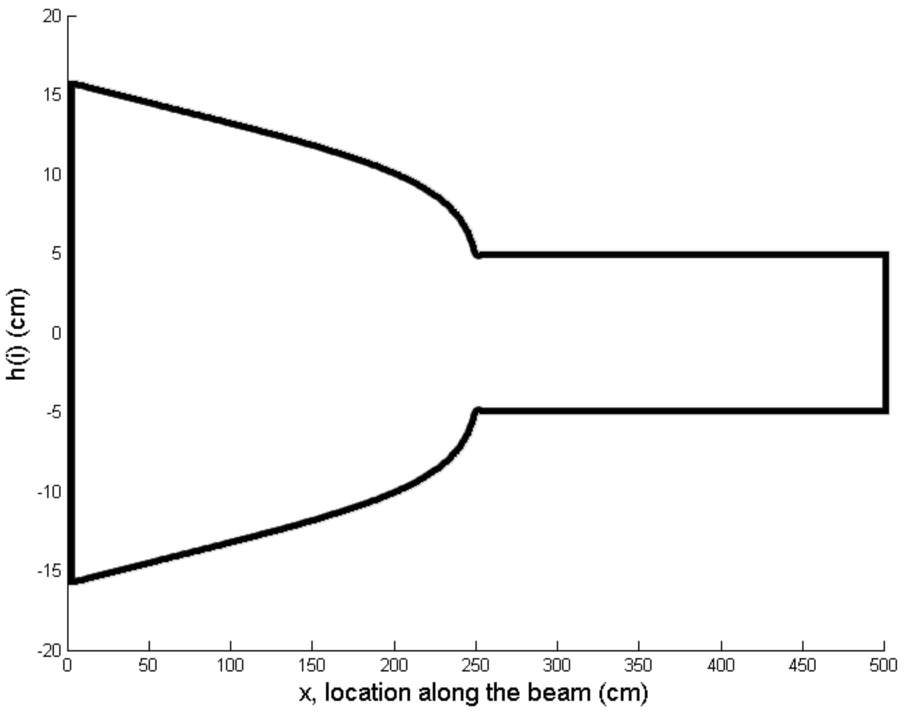

Figure 8.9: $h_{(i)}$ of optimal beam design under loading condition 1

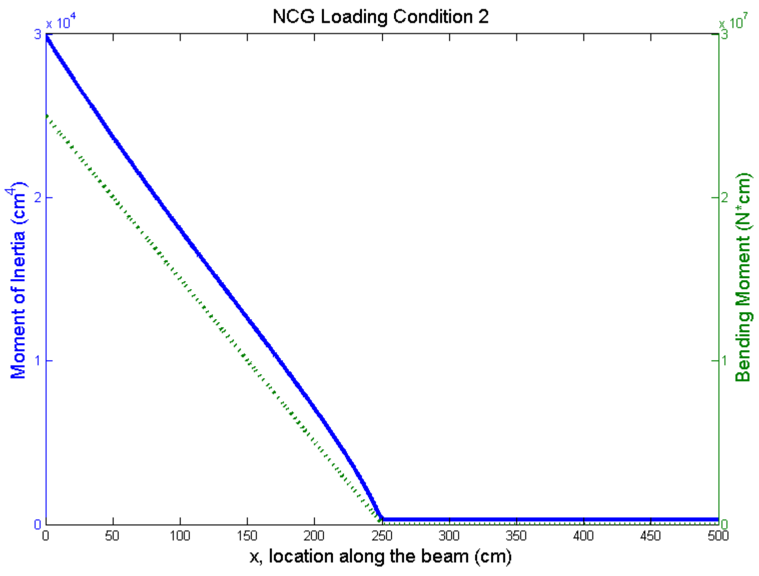

Figure 8.10: x vs. Moment ($M_{\hat{1},(i)}$) and sectional moment of inertia ($I_{(i)}$)

The solution for loading condition 2 is obtained by solving the following optimization statement.

$$\min_{b,h} \Psi_{\hat{2}} = V(b,h)$$

$$+ \sum_{i=1}^{N} q_{\hat{2}\alpha(i)}(\max(0, \frac{3\sigma_{\hat{2},(i)}}{\bar{\sigma}} - 1)^2) \tag{8.10a}$$

$$+ \sum_{i=1}^{N} q_{bh_{(i)}}(\max(0, h_{(i)} - 4b_{(i)})^2) \tag{8.10b}$$

subject to

$$3 \leq b_{(i)} \qquad i = 1, \cdots, N \tag{8.10c}$$

$$10 \leq h_{(i)} \qquad i = 1, \cdots, N \tag{8.10d}$$

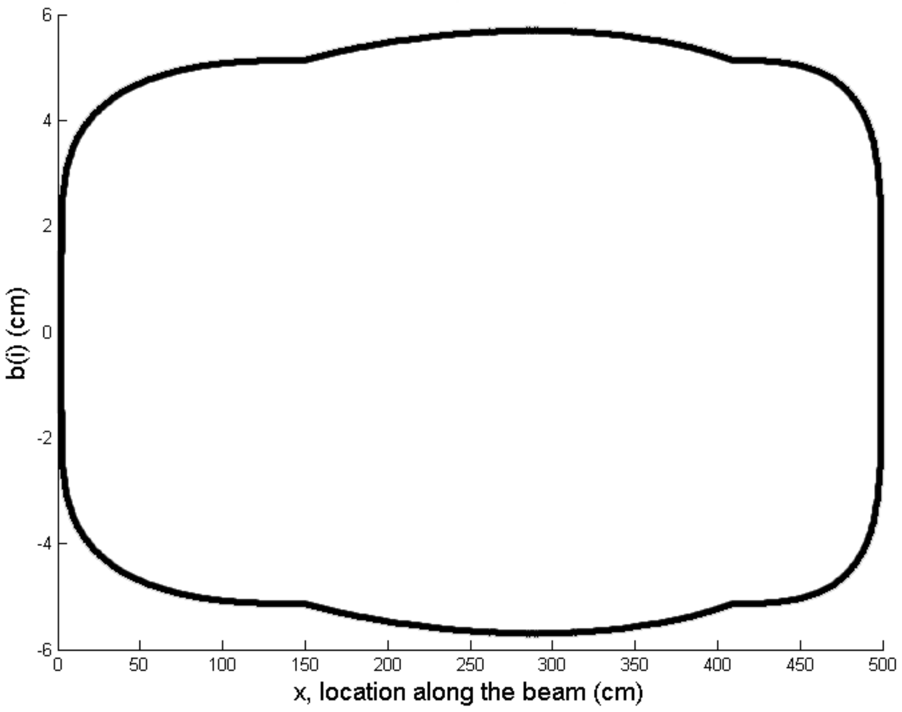

Figure 8.11: $b_{(i)}$ of optimal beam design under loading condition 2

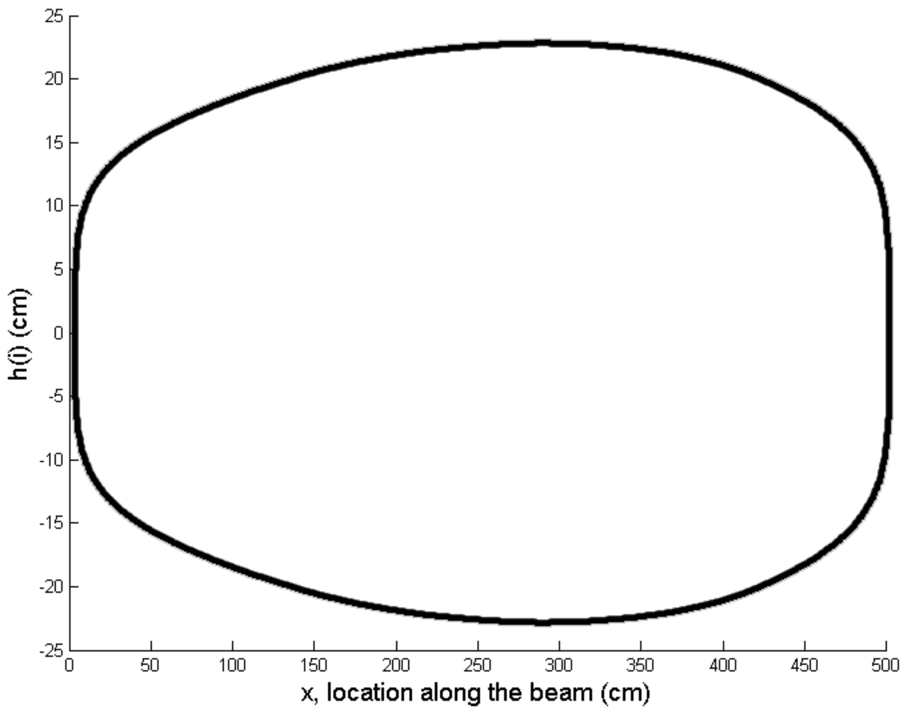

Figure 8.12: $h(i)$ of optimal beam design under loading condition 2

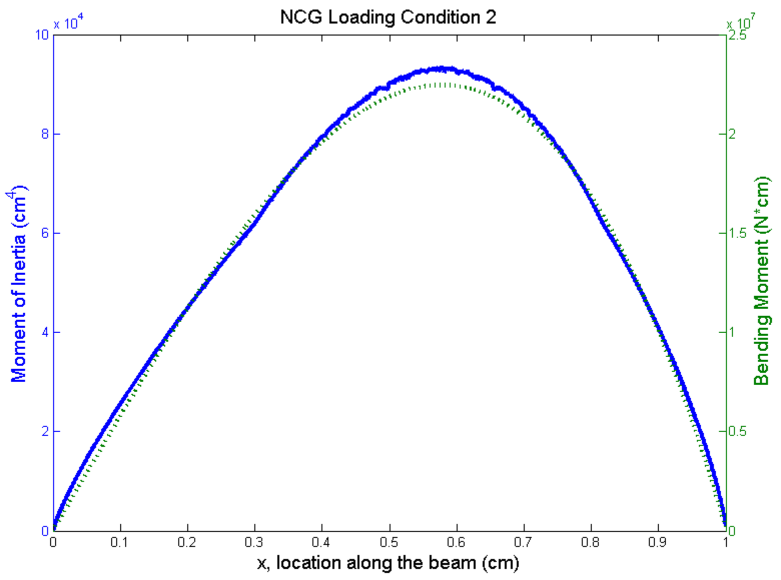

Figure 8.13: Plot of x vs. Moment, $M_{\hat{2},(i)}$, and sectional moment of inertia, $I_{(i)}$

The optimizer finds solutions that have sectional moment of inertia ($I_{(i)}$) correlating well with the bending moment distribution ($M_{(i)}$). When the beam is optimized for the loading condition 1, the height and the width in the second half of the beam go to the minimum value ($h_{(i)} = 10,\ b_{(i)} = 3$) since the beam does not support any bending moment. When the beam is optimized for the loading condition 2, it has the lowest sectional moment of inertia at two ends ($i = 0, 500$) of the beam. The sectional moment of inertia gradually increases as it approach mid-section where the bending moment increases. Using the optimization results above, following discipline objective functions are defined to measure the deviation from its respective optimal design ($x_{\hat{0}^{\hat{i}}}, x^*_{\hat{i}}$). The discipline level objective functions are defined as follows:

$$\begin{aligned} f_{\hat{1}}(x_{\hat{0}}, x_{\hat{1}}) &= \left(x_{\hat{0}} - x_{\hat{0}}^{\hat{1}}\right)^2 + \left(x_{\hat{1}} - x_{\hat{1}}^*\right)^2 \\ &+ \sum_{i=1}^{N} q_{\alpha_{\hat{1}(i)}} \cdot \max\left(0, \frac{3\sigma_{\hat{1}(i)}}{\bar{\sigma}} - 1\right)^2 \\ &+ \sum_{i=1}^{N} q_{bh_{(i)}} \cdot \max\left(0, h_{(i)} - 4b_{(i)}\right)^2 \end{aligned} \tag{8.11}$$

$$\begin{aligned} f_{\hat{2}}(x_{\hat{0}}, x_{\hat{2}}) &= \left(x_{\hat{0}} - x_{\hat{0}}^{\hat{2}}\right)^2 + \left(x_{\hat{2}} - x_{\hat{2}}^*\right)^2 \\ &+ \sum_{i=1}^{N} q_{\alpha_{\hat{2}(i)}} \cdot \max\left(0, \frac{3\sigma_{\hat{2}(i)}}{\bar{\sigma}} - 1\right)^2 \\ &+ \sum_{i=1}^{N} q_{bh(i)} \cdot \max\left(0, h_{(i)} - 4b_{(i)}\right)^2 \end{aligned} \tag{8.12}$$

where the penalty term multipliers ($q_{\alpha_{\hat{1}(i)}}, q_{\alpha_{\hat{2}(i)}}, q_{bh(i)}$) are fixed at 0.5% of the initial volume of the beam ($0.005 \cdot V(x_0) = 56.26$). To construct the system level objective function, $f^*_{\hat{i}}, prr_{\hat{i}}, Z_{\hat{i}}$ are computed first. The numerical value of the parameters are summarized in Table 8.3.

The system level objective function is defined as follows:

$$f_{sys}(x_{\hat{0}}, x_{\hat{1}}, x_{\hat{2}}) = exp\left(w_{\hat{1}} \left(\frac{f_{\hat{1}}(x_{\hat{0}}, x_{\hat{1}}) - f^*_{\hat{1}}}{prr_{\hat{1}}} \right)^2 + w_{\hat{2}} \left(\frac{f_{\hat{2}}(x_{\hat{0}}, x_{\hat{2}}) - f^*_{\hat{2}}}{prr_{\hat{2}}} \right)^2 \right) \tag{8.13}$$

$\hat{i}$	$f^*_{\hat{i}}$	$Z_{\hat{i}}$	$prr_{\hat{i}}$
$\hat{1}$	4.34E+03	1.61E+05	1.56E+05
$\hat{2}$	1.16E+02	1.56E+05	1.56E+05

Table 8.3: System level objective function parameters

A complete system level optimization statement is defined as follows:

$$\min_{x_{\hat{0}}=[b_{(i)},h_{(i)}]} \quad f_{sys}(x_{\hat{0}}, x_{\hat{1}}, x_{\hat{2}}) \tag{8.14a}$$

subject to

$$3 \leq b_{(i)} \qquad i = 1, \cdots, \frac{N}{2} \tag{8.14b}$$

$$10 \leq h_{(i)} \qquad i = 1, \cdots, \frac{N}{2} \tag{8.14c}$$

$$x_{\hat{1}},\ x_{\hat{2}} = \text{constant} \tag{8.14d}$$

Corresponding $\hat{j}^{\text{th}}$ discipline level optimization statements are defined as follows:

$$\min_{x_{\hat{j}}=[b(i),h(i)]} \quad f_{\hat{j}}(x_{\hat{0}}, x_{\hat{j}}) \tag{8.15a}$$

subject to

$$3 \le b_{(i)} \qquad i = \frac{N}{2} + 1, \cdots, N \tag{8.15b}$$

$$10 \le h_{(i)} \qquad i = \frac{N}{2} + 1, \cdots, N \tag{8.15c}$$

$$x_{\hat{0}} = \text{constant} \tag{8.15d}$$

8.2.2.1 Discussion and result

The system level objective function is optimized with varying weights $w_{\hat{1}} = [0,\ 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1]$. Converged solutions are Pareto optimal to one another as seen in Fig. 8.14. When there is a large number of design variables, it is often difficult to identify a Pareto front using brute force methods. Identifying a set of Pareto optimal provides valuable insights to the feasible design configuration of the beam, because the true Pareto front can be estimated by interpolating the converged solutions. A convex shape emerges when the solutions are interpolated.

When the system level objective function is optimized with varying weights, the system level optimization converges at different locations; the converged solutions are consistent with the physical interpretation of the weights distribution. When $w_{\hat{1}} > w_{\hat{2}}$, the system level solution is closer to the optimal solution under loading condition 1 and vice versa.

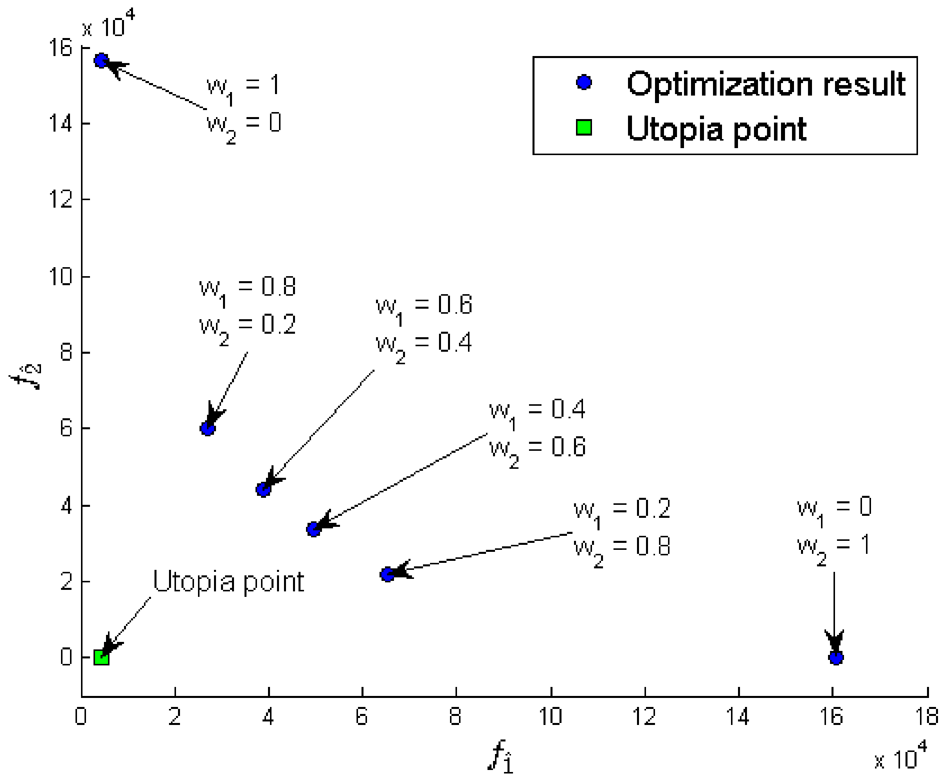

Figure 8.14: Converged solutions of multi-objective beam

On average, each simulation lasts $4.29 \cdot 10^3$ seconds. Four system level optimizations last total $1.71 \cdot 10^4$ seconds. The time required for each optimization is summarized in Table 8.4.

Sum	Mean	Stdev	$w_{\hat{1}} = 0.2$	$w_{\hat{1}} = 0.4$	$w_{\hat{1}} = 0.6$	$w_{\hat{1}} = 0.8$	-
1.71E+04	4.29E+03	4.06E+0.2	4.52E+03	4.52E+03	3.58E+03	4.52E+03	sec.

Table 8.4: Time required to complete system level optimizations

The sectional moment of inertia of the converged solutions are plotted in Fig. 8.15. Once again, the cross sectional moment of inertia of the beam correlates well with physical interpretation of the weights distribution.

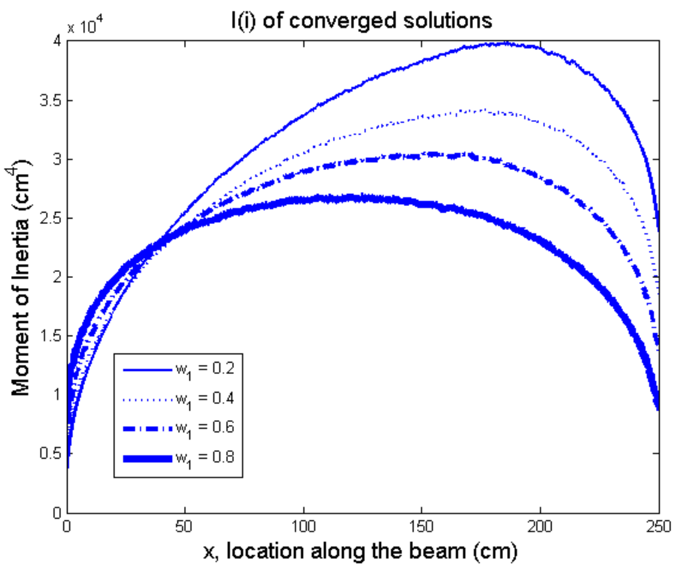

Figure 8.15: $I_{(i)}$ of the converged solutions

8.3 Summary

The bi-level multi-objective algorithm with the bounded nonlinear conjugate gradient method is presented for large-scale box-constrained optimization problems. The utility of the new algorithm is demonstrated by solving two numerical examples. The first problem is a large-scale box-constrained optimization problem with three objectives, which demonstrate different orders of magnitude. The system level optimization with varying weights identifies a Pareto front with evenly distributed Pareto optimal solutions. The second problem is a beam design optimization problem under two loading conditions. The algorithm also identifies a Pareto front with evenly-distributed Pareto optimal points. The successful identification of Pareto fronts in both problems implies that the proposed algorithms are effective in finding Pareto fronts of large-scale box-constrained optimization problems.

CHAPTER IX

Conclusion

9.1 Thesis contributions

In the first phase of this research, a new bi-level multi-objective algorithm is created by combining the strengths of known algorithms. It can provide solutions to complex multidisciplinary optimization problems. It decomposes an optimization problem into smaller problems by a global and local decomposition method, and a new multi-objective transformation method is proposed to accommodate the bi-level structure. The new algorithm is found to be effective when tested with two numerical optimization problems as it identified a Pareto front of each respective problem. The first problem is composed by combining two benchmark optimization problems: geometric programming problem and Golinski speed reducer problem; the second problem is a multi-objective version of the Sellar problem, which has extremely narrow Pareto front. In both cases, the new algorithm converges onto the neighborhood of a numerical Pareto front constructed by Monte Carlos simulation.

In the second phase of the research, two upgrades were completed in the CG method for solving box-constrained optimization problems with a large number of design variables. The first development implemented the gradient projected method, thus creating a bounded search direction update method. In this manner, the CG method no longer requires solving a constrained optimization with box constraints

Lagrange multipliers. The bounded search direction update method is of great utility especially when the optimal solution contains many design variables on upper and lower bounds because the bounded search direction update method significantly accelerates the rate of convergence.

A multi-variate step length method is the other modification method considered; it finds a vector of step length that performs betters than a uniform step length. The multi-variate step length method is found to be less sensitive to such scaling issues that cause CG method to stall at sub-optimal points. When the problem does not have a scaling issue; it does not show any merit while increasing computational cost significantly.

Lastly, the modified CG method is integrated in the bi-level multi-objective MDO algorithm for creating the ability to handle a large number of design variables during the MDO analysis. The latter capability is demonstrated by solving two numerical examples with up to 2000 design variables. One of the two problems conducts the structural design of a beam under multiple loading conditions. The correctness of the results is demonstrated through comparisons with the expected analytical solutions.

9.2 Future research

The following are recommendations for future research. In principle, the new bi-level algorithm can be extended to arbitrarily large number of levels. In fact, it is possible for an engineering optimization problem to exhibit several hierarchical levels. The performance of the bi-level algorithm with additional levels of optimization is not well understood. Learning how the algorithm behaves with larger number of levels will be useful. If the performance deteriorates too significantly, then engaging surrogate models can be considered for mitigating the increased computational burden.

The development implemented in the CG method were only tested against the Fletcher-Reeves CG method. Although computational advantages of the modified

CG method are identified, its computational advantage over alternative CG methods was not tested. A comparative performance study of alternative CG methods is recommended.

The partition scheme in the multi-variate step length method is heuristic. It is suspected that the partition scheme can be improved by insights from trust-region methods. The multi-variate step length method modifies the search direction and finds a step length simultaneously; this is similar to how the trust-region algorithm determines both the search direction and the step length simultaneously. A further investigation in the partition scheme is recommended.

The vector search algorithm used in the simultaneous multi-variate search algorithm is the bounded Nelder-Mead algorithm. Although the bounded Nelder-Mead algorithm performs reasonably well, alternative vector search methods can be considered.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] J. Agte, O. de Weck, S.-S. J., P. Arendsen, A. Morris, and M. Spieck, *Mdo: Assessment and direction for advancement an opinion of one international group*, Struct Multidiscip Optim, 40 (2010), pp. 17–33.

[2] N. Andrei, *An accelration of gradient descent algorithm with backtracking for unconstrained optimization*, Numerical Algorithms, 42 (2006), pp. 63–73.

[3] ——, *Another hybrid conjugate gradient algorithm for unconstrained optimization*, Numer Algor, 47 (2008), pp. 143–158.

[4] T. Athan and P. Papalambros, *A note on weighted criteria methods for compromise solutions in multi-objective optimization*, Eng Optim, 27 (1996), pp. 155–176.

[5] M. Balesdent, N. Berend, P. Depince, and A. Chriette, *A survey of multidisciplinary design optimization methods in launch vehicle design*, Struct Multidiscip Optim,, 45 (2012), pp. 619–642.

[6] H. Benson, *Existence of efficient solutions for vector maximization problems*, J. Optim. Theory Appl., 27 (1978), pp. 569–580.

[7] P. Bridgman, *Dimensional Analysis*, Yale University Press, New Haven.

[8] B. Brosowski and A. da Silva, *Simple tests for multicriteria optimality*, OR Spektrum, 16 (1994), pp. 243–247.

[9] P. H. Calamai and J. J. More, *Projected gradient methods for linearly constrained problems*, Mathematical Programming, 39 (1987), pp. 93–116.

[10] V. Chankong and Y. Haimes, *Multiobjective Decision Making Theory and Methodology*, Elsevier Science Publishing, New York.

[11] W. Chen, M. Wiecek, and J. Zhang, *Quality utility a compromise programming approach to robust design*, J. Mech. Des., 121 (1999), pp. 179–187.

[12] Y. Dai and Y. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM Journal on Optimization, 10 (1999), pp. 177–182.

[13] I. Das and J. Dennis, *A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems*, Struct. Optim., 14, pp. 63–69.

[14] K. Deb, *Genetic algorithms in multimodal function optimization*, tech. report, University of Alabama, Tuscaloosa, AL, 1989. TCGA Report No. 89002.

[15] J. D'Errico, *Bound constrained optimization using fminsearch*, Aug. 2005.

[16] E. D. Dolan and J. J. More, *Benchmarking optimization software with cops*, tech. report, Argonne National Laboratory, 2001. ANL/MCS-246.

[17] R. Eckenrode, *Weighting multiple criteria*, Manage. Sci., 12 (1965), pp. 180–192.

[18] F. Facchinei, S. Lucidi, and L. Palagi, *Truncated newton algorithm for largen scaled box constrained optimization*, Society for Industrial and Applied Mathematics, 12 (2002), pp. 1100–1125.

[19] R. Fletcher, *Practical Methods of Optimization vol.1: Unconstrained Optimization*, vol. 1, John Wiley & Sons, New York, 1987.

[20] R. Fletcher and C. Reeves, *Function minimization by conjugate gradients*, Comput. J., 7 (1964), p. 149154.

[21] E. Gerasimov and V. Repko, *Multicriterial optimizatin*, Sov. Appl. Mech., 14, pp. 1179–1184.

[22] J. C. Gilbert, *On the realization of the wolfe conditions in reduced quasi-newton methods for equality constrained optimization*, SIAM J. Optim., 7 (1997), pp. 780–813.

[23] J. C. Gilbert and J. Nocedal, *Global convergence properties of conjugate gradient methods for optimization*, SIAM J. Optim., (1992), pp. 21–42.

[24] J. Golinski, *Optimal synthesis problems solved by means of nonlinear programming and random methods*, J Mech Des-, 5 (1970), pp. 287–309.

[25] J. S. Gray, K. T. Moore, and B. A. Naylor, *OPENMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization*, in 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Fort Worth, TX, AIAA, AIAA-2010-9101, Fort Worth, Texas, August 2010, AIAA.

[26] W. W. Hager and H. Zhang, *A survey of nonlinear conjugate gradient methods*, Pacific Journal of Optimization, 2 (2006), pp. 35–58.

[27] J. He, S. Hannapel, and N. Vlahopoulos, *Multidisciplinary design optimization of ship hull forms*, In:Proceeding of ASME IDETC/CIE, 12 (2011), pp. 1100–1125.

[28] M. Hestenes and E. L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[29] B. Hobbs, *A comparison of weighting methods in power plant siting*, Decis. Sci., 11, pp. 725–737.

[30] Y. F. Hu and C. Storey, *Global convergence result for conjugate gradient method*, J. Optim. Theory Appl., 71 (1991), pp. 399–405.

[31] C. Hwang and K. Yoon, *Multiple attribute decision making: methods and applications*, Springer-Verlag, New York, 1981.

[32] D. Kim, S. Sra, and I. S. Dhillon, *Tackling box-constrained optimization via a new projected quasi-newton approach*, SIAM Journal on Scientific Computing, 32 (2010), pp. 3548–3563.

[33] H. Kim, *Target cascading in optimal system design*, PhD thesis, The University of Michigan, 2001.

[34] J. Koski, *Multicriterion optimization in structural design, In: Atrek, E.; Gallagher, R.H.; Ragsdell, K.M.; Zienkiewicz, O.C. (eds.) New Directions in Optimum Structural Design*, John Wiley and Sons, New York, 1984.

[35] J. Koski and R. Silvennoinen, *Norm methods and partial weighting in multicriterion optimization of structures*, Int. J. Numer. Methods Eng., 24 (1987), pp. 1101–1121.

[36] J. Lagarias, J. Reeds, M. Wright, and P. Wright, *Convergence properties of the nelder-mead simplex method in low dimensions*, SIAM J. Optim., 9 (1991), pp. 112–147.

[37] Y. Lai, T. L. T, and C. Hwang, *Topsis for modm*, European Journal of Operational Research, 76 (1994), pp. 486–500.

[38] J. Liu and S. Li, *New hybrid conjugate gradient method for unconstrained optimization*, Applied Mathematics and Computation, 245 (2014), pp. 36–43.

[39] Y. Liu and C. Storey, *Efficient generalized conjugate gradient algorithms, part 1: Theory*, J. Optim. Theory Appl., 69 (1991), pp. 129–137.

[40] D. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison Wesley, 1984.

[41] R. Marler and J. Arora, *Survey of multi-objective optimization methods for engineering*, Struct Multidiscip Optim, 26 (2004), pp. 369–395.

[42] ——, *Function-transformation methods for multi-objective optimization*, Eng Optim, 37 (2005), pp. 551–569.

[43] ——, *The weighted sum method for multi-objective optimization: new insights*, Struct Multidiscip Optim, 41 (2010), pp. 853–862.

[44] J. Martins and A. Lambe, *Multidisciplinary design optimization: A survey of architectures*, AIAA Journal, 51 (2013), pp. 2049–2075.

[45] A. Messac, *From dubious construction of objective functions to the application of physical programming*, AIAA J., 38, pp. 155–163.

[46] ——, *Physical programming: effective optimization for computational design*, AIAA J., 34, pp. 149–158.

[47] K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston, 1999.

[48] S. G. Nash, *A survey of truncated-newton methods*, Journal of Computational and Applied Mathematics, 124 (2000), pp. 45 – 59. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[49] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 2006.

[50] A. Osyczka, *An approach to multicriterion optimization problems for engineering design*, Comput. Methods Appl. Mech. Eng., 15 (1978), pp. 309–333.

[51] V. Pareto, *Manuale di Economica Politica*, Societa Editrice Libraria, Milan, 1906.

[52] E. Polak and G. Ribiere, *Note sur la convergence de directions conjugees, rev. francaise*, Rev. Francaise Informat Recherche Opertionelle, 3 (1969), pp. 35–43.

[53] B. Polyak, *The conjugate gradient method in extreme problems*, USSR Comp. Math., 9 (1969), pp. 94–112.

[54] K. Proos, G. Steven, O. Querin, and Y. Xie, *Multicriterion evolutionary structural optimization using the weighted and the global criterion methods*, AIAA J., 39 (2001), pp. 2006–2012.

[55] B. Roth and I. Kroo, *Enhanced collaborative optimization: a decomposition-based method for multidisciplinary design*, in ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 2008.

[56] M. Salukvadze, *Vector-Valued Optimization Problems in Control Theory*, Academic Press, New York, 1979.

[57] R. Sellar, S. Batill, and J. Renaud, *Response surface based, concurrent subspace optimization for multidisciplinary system design*, in 34th AIAA Aerospace Sciences Meeting and Exhibit, 1996.

[58] J. Sobieszczanski-Sobieski, J. Agte, and R. S. Jr., *Bi-level integrated system synthesis (bliss)*, tech. report, NASA, 1998. TM 1998-208715.

[59] J. Sobieszczanski-Sobieski, T. Altus, M. Phillips, and R. Sandusky, *Bi-level integrated system synthesis for concurrent and distributed processing*, AIAA Journal, 41 (2003), pp. 1996–2003.

[60] R. Tappeta and J. Renaud, *Multiobjective collaborative optimization*, J Mech Des, 119 (1997), pp. 403–411.

[61] D. Touati-Ahmed and C. Storey, *Efficient hybrid conjugate gradient techniques*, J. Optim. Theory Appl., 64, pp. 379–397.

[62] G. N. Vanderplaats, *Very large scale optimization*, tech. report, NASA, 2002. NASA/CR-2002-211768.

[63] ——, *Very large scale continuous and discrete variable optimization*, in 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA, 2004.

[64] T. Vincent, *Game theory as a design tool.*, ASME J. Mech. Transm. Autom. Des., 105 (1983), pp. 165–170.

[65] T. Vincent and W. Grantham, *Optimality in Parametric Systems*, John Wiley and Sons, New York, 1981.

[66] Q. Wang and Y. Che, *Sufficient descent polak-ribiere-polyak conjugate gradient algorithm for large-scale box-constrained optimization*, Abstract and Applied Analysis, (2004).

[67] P. Wolfe, *Convergence conditions for ascent method*, SIAM Review, 11 (1959), pp. 226–235.

[68] ——, *Convergence conditions for ascent method, ii: Some corrections*, SIAM Review, 13 (1971), pp. 185–188.

[69] C. Xu and J. Zhang, *A survey of quasi-newton equations and quasi-newton methods for optimization*, Annals of Operations Research, 103 (2001), pp. 213–234.

[70] J. Yu, M. Li, Y. Wang, and G. He, *A decomposition method for large-scale box constrained optimization*, Applied Mathematics and Computation, 231 (2014), pp. 9–15.

[71] P.-L. Yu and G. Leitmann, *Compromise solutions, domination structures, and salukvadzes solution*, J. Optim. Theory Appl., 13, pp. 362–378.

[72] M. Zeleny, *Multiple Criteria Decision Making*, McGraw Hill, New York.

[73] S. Zionts, *Multiple criteria mathematical programming: an updated overview and several approaches. In: Mitra, G. (ed.) Mathematical Models for Decision Support*, Springer-Verlag, Berlin.