Surrogate-Based Optimization using Multifidelity Models with Variable Parameterization

by

Theresa D Robinson

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007

Author . . . .
Department of Aeronautics and Astronautics
May 18, 2007

Certified by . . . .
Karen Willcox
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by . . . .
Robert Haimes
Principal Research Engineer
Thesis Supervisor

Certified by . . . .
Dimitri Bertsekas
McAfee Professor of Engineering
Thesis Committee Member

Accepted by . . . .
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Surrogate-Based Optimization using Multifidelity Models with Variable Parameterization

by

Theresa D Robinson

Submitted to the Department of Aeronautics and Astronautics
on May 18, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Engineers are increasingly using high-fidelity models for numerical optimization. However, the computational cost of these models, combined with the large number of objective function and constraint evaluations required by optimization methods, can render such optimization computationally intractable. Surrogate-based optimization (SBO) - optimization using a lower-fidelity model most of the time, with occasional recourse to the high-fidelity model - is a proven method for reducing the cost of optimization. One branch of SBO uses lower-fidelity physics models of the same system as the surrogate. Until now however, surrogates using a different set of design variables from that of the high-fidelity model have not been available to use in a provably convergent numerical optimization. New methods are herein developed and demonstrated to reduce the computational cost of numerical optimization of variable-parameterization problems, that is, problems for which the low-fidelity model uses a different set of design variables from the high-fidelity model. Four methods are presented to perform mapping between variable-parameterization spaces, the last three of which are new: space mapping, corrected space mapping, a mapping based on proper orthogonal decomposition (POD), and a hybrid between POD mapping and space mapping. These mapping methods provide links between different models of the same system and have further applications beyond formal optimization strategies. On an unconstrained airfoil design problem, it achieved up to 40% savings in high-fidelity function evaluations. On a constrained wing design problem it achieved 76% time savings, and on a bat flight design problem, it achieved 45% time savings. On a large-scale practical aerospace application, such time savings could represent weeks.

Thesis Supervisor: Karen Willcox
Title: Associate Professor of Aeronautics and Astronautics

Thesis Supervisor: Robert Haimes
Title: Principal Research Engineer

Acknowledgments

I would like to acknowledge my advisor, Professor Karen Willcox, for providing the right balance of high expectations and support during my PhD program. She is an excellent advisor and I learned much, both about technical matters and life in academia, from her. I would like to thank my co-advisor, Bob Haimes, for both his technical help, especially getting various pieces of software working, and his blunt honesty. Professor Dimitri Bertsekas, the third member of my thesis committee, has taught me much about optimization methods.

Thank you to Michael Eldred of Sandia National Laboratories for our collaboration over the last few years. Mike directly contributed to the development of corrected space mapping and the hybrid POD/space mapping. He provided many useful improvements to the text of this thesis. Thank you to Natalia Alexandrov of NASA Langley, whose work on trust region methods provided much of the foundation of this work, and who provided numerous detailed comments on this thesis. Dr. David Willis developed the aerodynamic codes used both for the wing design problem and for the flapping-flight problem. He spent many hours developing and debugging these programs, and helping me interpret the solutions found by my algorithms. Thank you to Dr. Chris Kuklewicz for helping me develop and debug some early test code.

I would like to acknowledge the financial support of the Computational Engineering Programme of the Singapore/MIT Alliance. Also, I would like to thank the Computer Science Research Institute at Sandia National Laboratories for sponsoring my research for one summer, providing me with an opportunity to interact with the excellent research staff in the Sandia Optimization and Uncertainty Quantification Group. Thank you to Krzysztof Fidkowski, Garrett Barter, and Todd Oliver for keeping the computers running, and running the needed software, in the ACDL. Many thanks to Jean Sofronas for all the work she does in support of the laboratory.

Finally, I would like to extend my deepest thanks to my husband, Andrew Lefoley Robinson. His help and support have been invaluable, and I appreciate that he continued to have confidence in me, even when I did not.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Simulation–based design optimization methods can be used in many settings within the conceptual and preliminary design of engineering systems. For example, in aircraft design, multidisciplinary design optimization (MDO) methods have been used to achieve systematic exploration of the design space, thus offering the potential of increased aircraft performance and profitability [48]. Examples of large multidisciplinary aerospace problems solved with MDO methods are described in [66], [40] and [38]. A broad class of design applications of interest can be described numerically by simulation models that depend on a set of design variables, $\mathbf{x}$, which represent the design decisions over which the designer has control, and a set of state variables, $\mathbf{y}$, which describe the physical state of the system. For this class of applications, the design problems can be formulated as a nonlinear programming problem

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) & & & (1.1)\\
\text{Subject to} \quad & R(\mathbf{x},\mathbf{y}) & &= 0, & (1.2)\\
& \mathbf{g}(\mathbf{x},\mathbf{y}) & &\leq 0, & (1.3)\\
& \mathbf{h}(\mathbf{x},\mathbf{y}) & &= 0, & (1.4)\\
& \mathbf{x}_{LB} \leq \mathbf{x} & &\leq \mathbf{x}_{UB}. & (1.5)
\end{aligned}$$

The design vector $\mathbf{x} \in \mathbb{R}^n$ contains design variables, or numerical quantities over which the designer has control. The objective function $f(\mathbf{x}, \mathbf{y})$ is the quantity the designer wishes to minimize. Examples of objective functions in aircraft design include drag, weight, cost, and combinations thereof. The set of state equations $R(\mathbf{x}, \mathbf{y})$ describe the physical governing equations that the system must satisfy and establish the interpendence of the state variables and design variables. They are normally a set of equations approximating physical phenomena. For instance, in the aerodynamic portion of an aircraft design problem, the state equations can use computational fluid dynamics to relate state variables such as air velocities and pressures to design variables defining the outer mold line of the aircraft. The constraint vectors $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ represent conditions that must be satisfied by the design. Examples of constraints include that lift is equal to weight or that stress in a structural member does not exceed some fraction of the material's strength. The bounds $\mathbf{x}_{LB}$ and $\mathbf{x}_{UB}$ on the design variables specify the range of allowable values of $\mathbf{x}$. For example, these can specify minimum thicknesses of structural members.

Here, the design problem is formulated using a reduced space optimization approach. That is, it is assumed that the state equation (1.2) can be solved explicitly for the state variables $\mathbf{y}$ as a function of the design variables $\mathbf{x}$. This permits the replacement of $\min_{\mathbf{x},\mathbf{y}}$ with $\min_{\mathbf{x}} f(\mathbf{y}(\mathbf{x})) = \min_{\mathbf{x}} f(\mathbf{x})$, where the dependence of $\mathbf{y}$ on $\mathbf{x}$ is implicit through the state equation (1.2). In addition, each equality constraint in Equation (1.4) can be replaced by two inequality constraints, and the bounds on the design variables in (1.5) are a special case of inequality constraints. The problem can then be written

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{Subject to} \quad & \mathbf{c}(\mathbf{x}) \leq 0. \end{aligned} \tag{1.6}$$

The computational expense of solving problem (1.6) is a result of potentially numerous factors. This work addresses two of those factors: firstly, the time taken to evaluate a given design, i.e. to evaluate $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$, may be large. Secondly, there may be a large number of design variables, causing n, the dimension of $\mathbf{x}$, to be large. This typically increases the number of function evaluations required to solve the problem. In addition, many engineering codes do not provide analytic gradients, and finite differences are required to compute the derivative information required for use in a gradient–based optimization solver. For example, a finite difference calculation of $\frac{df}{d\mathbf{x}}$ requires $O(n)$ function evaluations. Two AIAA white papers on MDO, [34] and [1], note the significant computational challenges presented by applying optimization to models with high fidelity and high computational costs.

1.2 Surrogate–Based Optimization

Surrogate–based optimization (SBO) has been proposed as one method to achieve high–fidelity design optimization at reduced computational cost. In SBO, a surrogate — or less expensive and lower–fidelity model — is used to perform the majority of the optimization, with recourse less frequently to the high–fidelity model. The determination of when to use the high–fidelity model is based on some rigorous procedure for deciding the appropriate level of fidelity.

Past work has focused on providing lower–fidelity models — $\hat{f}$ for f and $\hat{\mathbf{c}}$ for $\mathbf{c}$ — that are computationally efficient to evaluate. These models can be roughly divided into three categories: data fit surrogates, typically using interpolation or regression of the high–fidelity model evaluated at one or more sample points [70]; reduced–order models, derived using techniques such as proper orthogonal decomposition (POD) and modal analysis; and hierarchical models, also called multifidelity, variable–fidelity, or variable–complexity models. In this last case, a physics–based model of lower fidelity and reduced computational cost is used as the surrogate in place of the high–fidelity model. The multifidelity case can be further divided based on the means by which the fidelity is reduced in the lower–fidelity model. The low–fidelity model can be the same as the high–fidelity, but converged to a higher residual tolerance [33]; it can be the same model on a coarser grid [7]; or it can use a simpler engineering model that neglects some physics modeled by the high–fidelity method [5].

Much work has been performed on developing SBO methods that are provably convergent to an optimum of the high–fidelity problem. Ref. [58] reviews a broad spectrum of SBO work. One promising group of methods is based on trust–region model management (TRMM), which imposes limits on the amount of optimization performed using the low–fidelity model, based on a quantitative assessment of that model's predictive capability. TRMM evolved from classical trust–region algorithms [57], which use quadratic surrogates, and have more recently been used for surrogates of any type [50]. These TRMM methods are provably convergent to an optimum of the high–fidelity model [22, 32], provided the low–fidelity model is corrected to be at least first–order consistent with the high–fidelity model. Correcting to second–order or quasi–second–order consistency provides improved performance [27]. Ref [73] presents a survey of unconstrained trust–region methods.

A number of researchers have developed SBO methods for constrained problems. Booker et al. developed a direct–search SBO framework that converges to a minimum of an expensive objective function subject only to bounds on the design variables, and that does not require derivative evaluations [18]. Audet et al. extended that framework to handle general nonlinear constraints [11] using a filter method for step acceptance [10]. Rodriguez et al. developed a gradient–based TRMM augmented–Lagrangian strategy using response surfaces, and showed that using separate response surfaces for the objective and constraints provided faster convergence than using a single response surface for the augmented Lagrangian [60]. Alexandrov et al. developed the MAESTRO class of methods, which use gradient based optimization and trust region model management, and compared them to a sequential quadratic programming (SQP)–like TRMM method [7]. Under fairly mild conditions on the models, these methods are also convergent to a local minimum of the constrained high–fidelity problem [26, 22]. Ref. [62] reviews a broad spectrum of trust–region methods for constrained optimization and [22] has an extensive bibliography relating to both classical trust–region methods and to more recent TRMM methods, for both the unconstrained and the constrained case.

1.3 Present Limitations on SBO

The SBO methods developed to date achieve computational gain by performing most of the analysis on the low–fidelity model; however, they require that the high– and low–fidelity models operate with the same set of design variables. For practical design applications however, multifidelity models are often defined over different design spaces. That is, the low–fidelity model $\hat{f}(\hat{\mathbf{x}})$ takes $\hat{\mathbf{x}} \neq \mathbf{x}$ as an input. The dimension of $\hat{\mathbf{x}}$, in this work called $\hat{n}$, can be different from n, the dimension of $\mathbf{x}$. For example, the variable–fidelity supersonic business jet problem considered by Choi et al. [21] is to optimize the aerodynamic performance of a low–supersonic–boom aircraft. The low–fidelity model uses classical supersonic aerodynamics and vortex lattice–methods, and operates on an aircraft defined by 16 design variables: the wing area, aspect ratio, and sweep, the location of the wing root leading edge, the thickness to chord length at three locations on the wing, the minimum cockpit diameter, the minimum cabin diameter, and the fuselage radii at six locations. The geometric model of the aircraft used for the low–fidelity analysis is shown in Figure 1-1. The high–fidelity model uses 126 design variables: leading and trailing edge droop, twist, and 15 camber Hicks–Henne bumps at each of 7 locations on the wing. The geometric model of the aircraft used for the high–fidelity analysis is shown in Figure 1-2. The high–fidelity analysis uses the Euler equations and provides the gradient through the adjoint method.

However, because existing SBO methods cannot be applied to problems where the low– and high–fidelity models use different design variables, Choi et al. used the two models sequentially, optimizing first using the low–fidelity model, with Kriging corrections applied, and using the result of that optimization as a starting point for optimization using the high–fidelity model. This also required an additional step of mapping the low–fidelity optimum to the high–fidelity space, in order to provide a starting point for high–fidelity optimization.

New methodology is therefore required for expanding surrogate–based design optimization to the case where the low– and high–fidelity models use different design variables. Further, combining a low–fidelity model with a coarser parameterization of

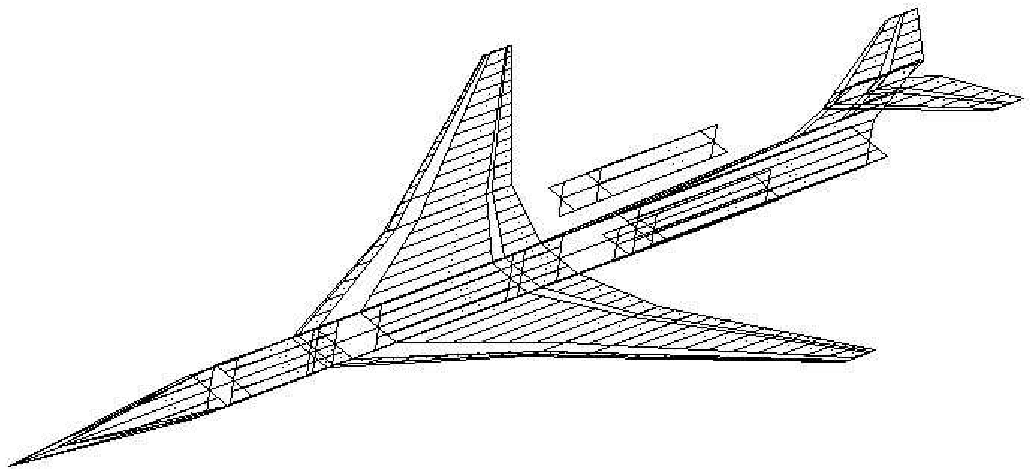

Figure 1-1: Visualization of the low–fidelity design variables for the supersonic business jet. Image courtesy I. Kroo.

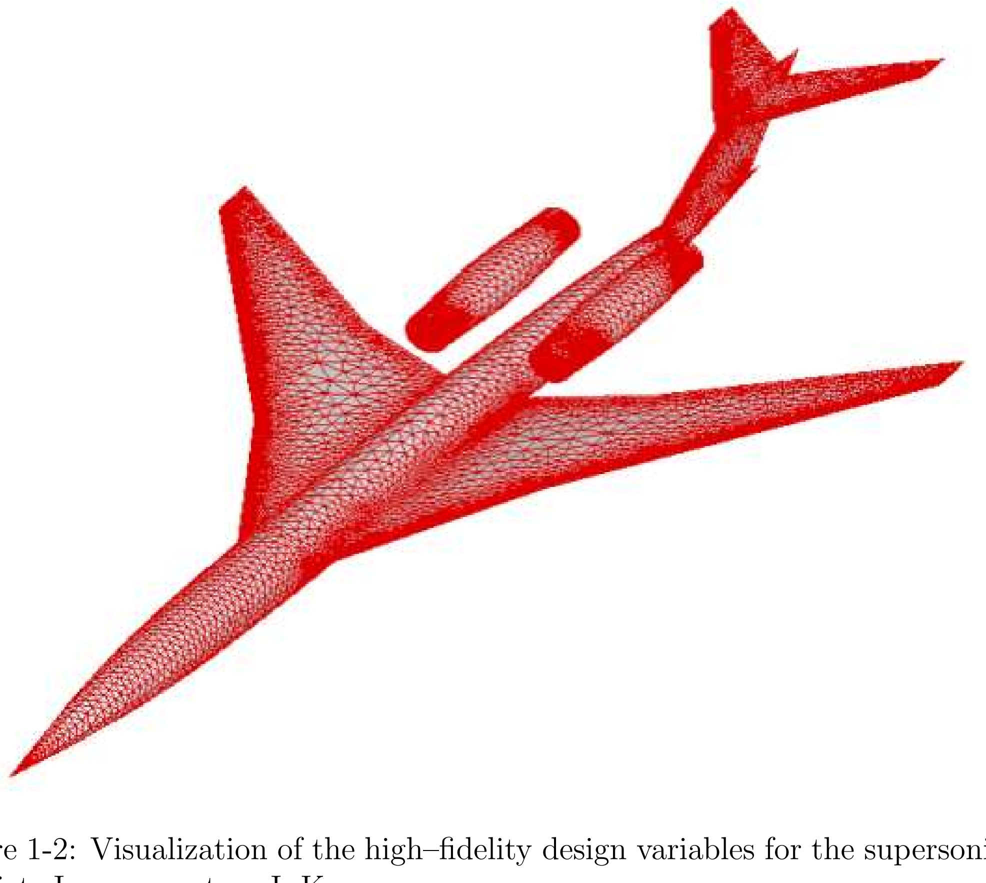

Figure 1-2: Visualization of the high–fidelity design variables for the supersonic business jet. Image courtesy I. Kroo.

the design offers the opportunity for additional reduction in computational complexity and cost beyond current SBO methods. To achieve this, new design methodology is required that incorporates variable–parameterization models into SBO methods.

1.4 Mapping

This thesis specifically addresses this need for new methodology by developing methods for mapping between design spaces. These mapping methods allow for SBO to be applied to the case where the low– and high–fidelity models use different design variables. These mapping methods eliminate the need to perform sequential optimization of each model: the low– and high–fidelity analyses are used in a single optimization. While optimization methods have predominantly been used in the preliminary and detailed design phases, there is an increasing drive to include higher–fidelity models, such as simulation–based physics models, in the conceptual phase of the design process. This drive stems from a desire to make better design decisions when those decisions can still be made: the more knowledge available earlier in the design process, the better the ultimate design. An additional benefit of these mapping methods is that even in the case where the low– and high–fidelity analyses are performed sequentially, the mapping can be applied after using the low–fidelity analysis, to convert the optimum in the low–fidelity space to a starting point in the high–fidelity space. Further, these mappings allow optimization paths in one space to be converted to paths in another space. If one space is more suitable for optimization, and another for human–computer interaction, this allows improved insight into the design process. Moreover, in the design of complex engineering systems, such as an aircraft, different levels of analysis are often used at different times by different groups. Mapping can provide a systematic way to translate results from one analysis setting to another, even outside the purview of formal optimization.

1.5 Terminology

Since the majority of the literature addresses single–parameterization problems, existing terminology may be inconsistent with new ideas presented in this thesis. The following is how terms are used in this work.

The *fidelity* of a mathematical or computational model is the degree to which the model accurately reflects reality. Higher–fidelity models correspond more closely to observed experimental results. Some models may be higher–fidelity in some regions of the design space than others. For the purposes of this thesis, it is assumed that it is trivial to list models in order of decreasing fidelity, and that this order stays constant over the design space of interest.

The *computational cost* of a model is the amount of time it takes a computer to find the outputs $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$, given a single set of design variables $\mathbf{x}$.

It is assumed in this thesis that higher fidelity models have higher computational cost than lower–fidelity models. This is not always the case. However, when the higher–fidelity model has less computational cost than the lower–fidelity model, it should be used. There is no reason to incorporate a model with both lower fidelity and higher computational cost.

A *variable–fidelity* design problem is a physical problem for which at least two mathematical or computational models exist. One model has both higher fidelity and higher computational cost than the other model.

A *single–fidelity* optimization method is a method that uses only one model of the physical system in order to minimize an objective function, possibly subject to some constraints.

A *multifidelity* optimization method is a method that uses more than one model of a variable–fidelity problem. Normally, the goal of a multifidelity optimization method is to find the optimum of the higher–fidelity model at reduced computational cost. It uses the low–fidelity model for some portion of the analysis as a means to that end.

The *parameterization* of a model is the set of design variables used as inputs to the model.

A *variable–parameterization* problem is a variable–fidelity problem in which each of the models has a different parameterization.

A *mapping* is a method for linking the design variables in a variable–parameterization problem. Given a set of design variables in one parameterization, it provides a set of design variables in another parameterization.

The *dimension* of a model is the number of design variables.

A *variable–dimensional* problem in which each of the models has a different dimension.

1.6 Research Contributions

The contributions of this thesis are to:

1. Develop techniques for mapping between analysis models with different design parameterizations.

2. Develop a provably–convergent optimization framework for multifidelity models with variable parameterizations for both unconstrained and constrained optimization problems.

3. Compare these new mapping methods and optimization techniques with one another and existing techniques.

4. Demonstrate these new mapping methods and optimization techniques on engineering design problems within the context of aerospace engineering.

1.7 Thesis Summary

Chapter 2 presents existing TRMM methods for optimization, and expands their applicability to variable–parameterization problems. It also presents a new TRMM method for optimization — including proof of convergence — that solves the trust–region subproblem in the low–fidelity space. Chapter 3 details the four methods used to perform mapping between variable–parameterization spaces: space mapping, corrected space mapping, a mapping based on the proper orthogonal decomposition (POD), and a hybrid space mapping/POD method. Chapter 4 contains the results of these new methods applied to two unconstrained problems — the Rosenbrock problem and an airfoil design problem — and compares the results to existing methods applied to the same problems. Chapter 5 presents detailed results of these new methods applied to the Barnes problem, including a comparison against existing methods. It also presents the results of varying the constraint–management method and a number of algorithm parameters. Chapter 6 shows the results of these new methods applied to two constrained design problems — the planform design of a wing and a flapping flight problem — and compares the results to existing methods applied to the same problems. Finally, Chapter 7 presents conclusions and some avenues for future work.

Chapter 2

Trust–Region Model Management

Surrogates can be incorporated into optimization by using a formal model–management strategy. One such strategy is a trust–region model–management (TRMM) framework [6]. TRMM imposes limits on the amount of optimization performed using the low–fidelity model, based on a quantitative assessment of that model's predictive capability. TRMM developed from the classical trust–region optimization method based on quadratic Taylor series models [23].

TRMM methods are provably convergent to an optimum of the high–fidelity model, provided that the models satisfy a small number of assumptions, described in Section 2.1.1 below. The general approach in TRMM is to solve a sequence of optimization subproblems using only the low–fidelity model, with an additional constraint that requires the solution of the subproblem to lie within a specified trust region. The radius of the trust region is adaptively managed on each subproblem iteration using a merit function to quantitatively assess the predictive capability of the low–fidelity model.

Based on test results to date with variable–resolution and variable–fidelity physics models in CFD, TRMM has achieved 2– to 7–fold reductions in high–fidelity function evaluations [7]. In general, the savings depend on the relative properties of models. The improvement factor appears to increase with the dimension of the underlying governing equations. The technique performs well even in cases in which the gradients of the objective function in the two models have opposite directions [2].

This chapter first addresses the unconstrained algorithm. It shows the general algorithm without specifying the nature of the surrogate, and lists the assumptions required for convergence. It then details how to construct appropriate surrogates in each of the single–parameterization and variable–parameterization unconstrained cases. It then moves on to the constrained algorithm, discussing constrained TRMM in general and each of four constrained TRMM algorithms specifically. Finally, it introduces an algorithm that takes steps in the lower–complexity space, including a proof of convergence.

2.1 Unconstrained Algorithm

This algorithm solves an unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \tag{2.1}$$

where the objective function $f(\mathbf{x})$ is a real–valued twice–continuously differentiable function, which is bounded below.

Algorithm 1: Unconstrained TRMM Method

- **Step 0: Initialization** Choose an initial point $\mathbf{x}^0$ and an initial trust–region radius $\Delta^0 > 0$. Set $k = 0$. Choose constants $\eta > 0$, $0 < c_1 < 1$, $c_2 > 1$, $0 < r_1 < r_2 < 1$, $\Delta^* \geq \Delta^0$.

- **Step 1: Surrogate definition** Choose a surrogate $\tilde{f}^k$ that satisfies $\tilde{f}^k(\mathbf{x}^k) = f(\mathbf{x}^k)$ and $\nabla_{\mathbf{x}}\tilde{f}^k(\mathbf{x}^k) = \nabla_{\mathbf{x}} f(\mathbf{x}^k)$. The specific means for generating this model will be discussed later in this chapter: in Section 2.1.2 for single–complexity problems and in Section 2.1.3 for variable–complexity problems.

- **Step 2: Step calculation** Compute a step $\mathbf{s}^k$ such that $\|\mathbf{s}^k\| \leq \Delta^k$ and that satisfies the fraction of Cauchy decrease condition (described in Equation (2.7) below). The computation of this step uses only the surrogates and is discussed in Section 2.1.4. It is the solution of the trust–region subproblem

$$\begin{aligned} \min_{\mathbf{s}} \quad & \tilde{f}(\mathbf{x}^k + \mathbf{s}) \\ \text{Subject to} \quad & \|\mathbf{s}\| \leq \Delta^k. \end{aligned} \tag{2.2}$$

- **Step 3: Acceptance or rejection of the trial point** Compute $f(\mathbf{x}^k + \mathbf{s}^k)$ and define the trust–region ratio

$$\rho^k = \frac{f(\mathbf{x}^k) - f(\mathbf{x}^k + \mathbf{s}^k)}{f(\mathbf{x}^k) - \tilde{f}^k(\mathbf{x}^k + \mathbf{s}^k)}. \tag{2.3}$$

  If $\rho^k \geq \eta$ accept the step and define $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$; otherwise reject the step and define $\mathbf{x}^{k+1} = \mathbf{x}^k$. Define a *successful* iterate as one in which the step is accepted; otherwise it is an unsuccessful iterate.

- **Step 4: Trust–region radius update** Set

$$\Delta^{k+1} = \begin{cases} c_1\|\mathbf{s}^k\| & \text{if } \rho^k < r_1, \\ \min(c_2\Delta^k, \Delta^*) & \text{if } \rho^k > r_2, \\ \|\mathbf{s}^k\| & \text{otherwise.} \end{cases} \tag{2.4}$$

  Increment k by 1 and go to Step 1.

Typical values for the algorithm parameters are $\eta = 0.01$, $r_1 = 0.2$, $r_2 = 0.9$, $c_1 = 0.25$ and $c_2 = 2.5$.

2.1.1 Assumptions for Provable Convergence

Chapter 6 of Conn, Gould, and Toint [22] proves that the sequence of iterates generated by the algorithm above converges to a local minimum of the high–fidelity function under a number of assumptions, which are enumerated there. Those on the high–fidelity problem are relatively mild and satisfied by the majority of functions. The two assumptions of most interest in this work are

1. On each iteration, the surrogate $\tilde{f}^k(\mathbf{x})$ is zeroth– and first–order consistent with the high fidelity function $f(\mathbf{x})$ at the center of the trust region. That is,

$$\begin{aligned} \tilde{f}^k(\mathbf{x}^k) &= f(\mathbf{x}^k), \qquad (2.5) \\ \nabla_{\mathbf{x}}\tilde{f}^k(\mathbf{x}^k) &= \nabla_{\mathbf{x}} f(\mathbf{x}^k). \qquad (2.6) \end{aligned}$$

2. Each trial step must satisfy the fraction of Cauchy decrease (FCD) condition on the surrogate. That is, there must exist numbers $\zeta > 0$ and $C > 0$ independent of k such that

$$\tilde{f}^k(\mathbf{x}^k) - \tilde{f}^k(\mathbf{x}^k + \mathbf{s}^k) \geq \zeta \|\nabla_{\mathbf{x}}\tilde{f}^k(\mathbf{x}^k)\| \min\left(\Delta^k, \frac{\|\nabla_{\mathbf{x}}\tilde{f}^k(\mathbf{x}^k)\|}{C}\right). \qquad (2.7)$$

2.1.2 Generating a Surrogate: Single Parameterization

When both the high–fidelity and low–fidelity models are defined over the same design variables, that is, when $\hat{\mathbf{x}} = \mathbf{x}$, the problem is single–parameterization. The TRMM framework is well–established and thoroughly tested for this case. This section will review the creation of a surrogate for a single–parameterization set of variable–fidelity models.

The standard method for creating a surrogate is to use the low–fidelity model along with a simple polynomial correction. One of the requirements for provable convergence of the TRMM algorithm is that surrogate is first–order consistent with the high–fidelity function at the center of the trust region. A surrogate that satisfies this condition can be created using a low–fidelity model and a correction function. The correction can be additive, multiplicative, or a convex combination of the two, and it can be of any order [27]. The corrections can be additive or multiplicative. In [27], based on the problem and correction sample, additive corrections have been shown to work better for a greater number of problems.

In the sections below, the high–fidelity function is denoted β, the low–fidelity function is $\hat{\beta}$, and the corrected surrogate is $\tilde{\beta}$. In the unconstrained case $\beta = f$,

$\hat{\beta} = \hat{f}$ and $\tilde{\beta} = \tilde{f}$; however, in future sections, other functions will be corrected, and therefore these corrections are presented generally.

Additive Correction

In an additive correction, the difference between the high–fidelity and the low–fidelity models is approximated by a Taylor series.

An exact additive correction is the difference between the high– and low–fidelity models

$$\mathcal{A}(\mathbf{x}) = \beta(\mathbf{x}) - \hat{\beta}(\mathbf{x}). \tag{2.8}$$

The surrogate in the additive case is then given by

$$\tilde{\beta}_A(\mathbf{x}) = \hat{\beta}(\mathbf{x}) + A(\mathbf{x}), \tag{2.9}$$

where $A(\mathbf{x})$ is a second–order Taylor series expansion of $\mathcal{A}(\mathbf{x})$ around $\mathbf{x}^k$:

$$A(\mathbf{x}) = \mathcal{A}(\mathbf{x}^k) + \nabla_{\mathbf{x}}\mathcal{A}(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^k)^T\nabla^2_{\mathbf{x}}\mathcal{A}(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k). \tag{2.10}$$

Each of the terms of the expansion is calculated using

$$\mathcal{A}(\mathbf{x}^k) = \beta(\mathbf{x}^k) - \hat{\beta}(\mathbf{x}^k), \tag{2.11}$$

$$\nabla_{\mathbf{x}}\mathcal{A}(\mathbf{x}^k) = \nabla_{\mathbf{x}}\beta(\mathbf{x}^k) - \nabla_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k), \tag{2.12}$$

$$\nabla^2_{\mathbf{x}}\mathcal{A}(\mathbf{x}^k) = \nabla^2_{\mathbf{x}}\beta(\mathbf{x}^k) - \nabla^2_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k). \tag{2.13}$$

In Equation (2.13), exact Hessian matrices can be used if they are available. Otherwise the BFGS approximation to the Hessian matrix, described below, or another quasi–Newton approximation can be used.

Multiplicative Correction

In a multiplicative correction, the quotient of the high–fidelity and the low–fidelity models is approximated by a Taylor series. An exact multiplicative correction function is defined as

$$\mathcal{B}(\mathbf{x}) = \frac{\beta(\mathbf{x})}{\hat{\beta}(\mathbf{x})}. \tag{2.14}$$

The surrogate in the multiplicative case is

$$\tilde{\beta}_B(\mathbf{x}) = \hat{\beta}(\mathbf{x})B(\mathbf{x}), \tag{2.15}$$

where $B(\mathbf{x})$ is the second–order Taylor series expansion of $\mathcal{B}(\mathbf{x})$ around $\mathbf{x}^k$:

$$B(\mathbf{x}) = \mathcal{B}(\mathbf{x}^k) + \nabla_{\mathbf{x}}\mathcal{B}(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T\nabla^2_{\mathbf{x}}\mathcal{B}(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k). \tag{2.16}$$

Each of these terms is calculated using

$$\mathcal{B}(\mathbf{x}^k) = \frac{\beta(\mathbf{x}^k)}{\hat{\beta}(\mathbf{x}^k)}, \tag{2.17}$$

$$\nabla_{\mathbf{x}}\mathcal{B}(\mathbf{x}^k) = \frac{\hat{\beta}(\mathbf{x}^k)\nabla_{\mathbf{x}}\beta(\mathbf{x})^k - \beta(\mathbf{x}^k)\nabla_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k)}{\hat{\beta}^2(\mathbf{x}^k)}, \tag{2.18}$$

$$\begin{aligned}
\nabla^2_{\mathbf{x}}\mathcal{B}(x^k) &= \Big[\hat{\beta}^2(\mathbf{x}^k)\nabla^2_{\mathbf{x}}\beta(\mathbf{x}^k) - \beta(\mathbf{x}^k)\hat{\beta}(\mathbf{x}^k)\nabla^2_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k) \\
&- \nabla_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k)\hat{\beta}(\mathbf{x}^k)(\nabla_{\mathbf{x}}\beta(\mathbf{x}^k))^T + \nabla_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k)\beta(\mathbf{x}^k)(\nabla_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k))^T\Big] \\
&/ \ \hat{\beta}^3(\mathbf{x}^k).
\end{aligned} \tag{2.19}$$

Combination Additive/Multiplicative Corrections

Since both the additive and the multiplicative corrections result in a surrogate which is consistent at the center of the trust region, a convex combination of the two is also be consistent. Such a combination is

$$\tilde{\beta}_C(\mathbf{x}) = \Gamma\tilde{\beta}_A(\mathbf{x}) + (1-\Gamma)\tilde{\beta}_B(\mathbf{x}), \tag{2.20}$$

where $0 \leq \Gamma \leq 1$. The parameter Γ can be chosen a number of ways. One is to choose a point, in addition to the center of the current trust region, at which to exactly match the surrogate to the high–fidelity model.

Approximate Hessian matrices

These derivations have shown second–order corrections. For engineering problems, exact Hessian matrices $\nabla^2_{\mathbf{x}}\beta(\mathbf{x}^k)$ and $\nabla^2_{\mathbf{x}}\hat{\beta}(\mathbf{x}^k)$ are often unavailable. They can be estimated using finite differences or approximated through quasi–Newton approximations , such as the Broyden–Fletcher–Goldfarb–Shanno [19, 31, 36, 64] (BFGS) update. The BFGS approximation to the Hessian matrix requires only function value and gradient information. It provides an improved estimate of the Hessian matrix as the gradient becomes known at more points. The initial estimate of the Hessian matrix is

$$\mathbf{H}^0 = \frac{(\mathbf{y}^0)^T\mathbf{y}^0}{(\mathbf{y}^0)^T\mathbf{s}^0}\mathbf{I}, \tag{2.21}$$

where $\mathbf{s}^0$ is the step on the first trust–region iteration

$$\mathbf{s}^0 = \mathbf{x}^1 - \mathbf{x}^0, \tag{2.22}$$

$\mathbf{y}^0$ is the yield of the first step,

$$\mathbf{y}^0 = \nabla_{\mathbf{x}}\beta(\mathbf{x}^1) - \nabla_{\mathbf{x}}\beta(\mathbf{x}^0), \tag{2.23}$$

and $\mathbf{I}$ is the identity matrix. The estimate is then updated, each time the gradient is evaluated at a new point, using

$$\mathbf{H}^{k+1} = \mathbf{H}^k - \frac{\mathbf{H}^k\mathbf{s}^k(\mathbf{s}^k)^T\mathbf{H}^k}{(\mathbf{s}^k)^T\mathbf{H}^k\mathbf{s}^k} + \frac{\mathbf{y}^k(\mathbf{y}^k)^T}{(\mathbf{y}^k)^T\mathbf{s}^k}, \tag{2.24}$$

where $\mathbf{s}^k$ is the step on the k^{th} trust–region iteration

$$\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k, \tag{2.25}$$

and $\mathbf{y}^k$ is the yield

$$\mathbf{y}^k = \nabla_{\mathbf{x}}\beta(\mathbf{x}^{k+1}) - \nabla_{\mathbf{x}}\beta(\mathbf{x}^k). \tag{2.26}$$

The BFGS update requires safeguarding against numerical failures due to small denominators in Equation (2.24). The update is therefore skipped if $|(\mathbf{y}^k)^T\mathbf{s}^k| < 10^{-6}(\mathbf{s}^k)^T\mathbf{H}^k\mathbf{s}^k$ or $(\mathbf{s}^k)^T\mathbf{H}^k\mathbf{s}^k < 10^{-10}$. For first–order corrections (sufficient to ensure provable convergence of the SBO method), $\nabla^2_{\mathbf{x}}A(\mathbf{x}^k)$ and $\nabla^2_{\mathbf{x}}B(\mathbf{x}^k)$ are neglected in Equations (2.10) and (2.16). However, quasi–second–order corrections improve the convergence speed of the algorithm [27] and are therefore used in the majority of the work in this thesis.

2.1.3 Generating a Surrogate: Variable Parameterization

In a variable–parameterization problem, the low–fidelity model is defined over a set of design variables different from those of the high–fidelity model. In this case, surrogate creation requires two components: correction and design variable mapping. Past work on the TRMM framework has focused on the single parameterization case. This section addresses correction, while design variable mapping is covered in Chapter 3. For this section, assume that a mapping exists of the form $\hat{\mathbf{x}} = P(\mathbf{x})$ and that first and second–order derivatives of the mapping are available. This section details the newly–derived corrections to the mapped low–fidelity model for use in variable–parameterization optimization.

Additive Corrections

For some low–fidelity function $\hat{\beta}(\hat{\mathbf{x}})$, the corresponding high–fidelity function $\beta(\mathbf{x})$, and a mapping $\hat{\mathbf{x}} = P(\mathbf{x})$, the k^{th} additive–corrected surrogate is defined as

$$\tilde{\beta}^k(\mathbf{x}) = \hat{\beta}\left(P(\mathbf{x})\right) + A^k(\mathbf{x}). \tag{2.27}$$

In order to obtain quasi–second–order consistency between $\tilde{\beta}^k(\mathbf{x}^k)$ and $\beta(\mathbf{x}^k)$, we define the correction function, $A^k(\mathbf{x})$, using a quadratic Taylor series expansion of the difference $\mathcal{A}(\mathbf{x})$ between the two functions β and $\hat{\beta}$, about the point $\mathbf{x}^k$:

$$A^k = \mathcal{A}(\mathbf{x}^k) + \nabla_{\mathbf{x}}\mathcal{A}(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T\nabla^2_{\mathbf{x}}\mathcal{A}(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k). \tag{2.28}$$

The elements in this expansion are calculated using

$$\mathcal{A}(\mathbf{x}^k) = \beta(\mathbf{x}^k) - \hat{\beta}\left(P(\mathbf{x}^k)\right), \tag{2.29}$$

$$\begin{aligned} \frac{\partial\mathcal{A}(\mathbf{x}^k)}{\partial x_p} &= \frac{\partial\beta}{\partial x_p}(\mathbf{x}^k) - \sum_{j=1}^{\hat{n}}\frac{\partial\hat{\beta}}{\partial\hat{x}_j}\left(P(\mathbf{x}^k)\right)\frac{\partial\hat{x}_j}{\partial x_p}, \qquad (2.30)\\ & p = 1\ldots n, \\ & \qquad (2.31) \end{aligned}$$

$$\begin{aligned} \frac{\partial^2\mathcal{A}(\mathbf{x}^k)}{\partial x_p\partial x_q} &= H^k_{pq} - \sum_{j=1}^{\hat{n}}\Bigg(\frac{\partial\hat{\beta}}{\partial\hat{x}_j}\left(P(\mathbf{x}^k)\right)\frac{\partial^2\hat{x}_j}{\partial x_p\partial x_q} \\ &+ \sum_{\ell=1}^{\hat{n}}\hat{H}^k_{j\ell}\frac{\partial\hat{x}_j}{\partial x_p}\frac{\partial\hat{x}_\ell}{\partial x_q}\Bigg), \qquad (2.32)\\ & p = 1\ldots n, \quad q = 1\ldots n, \end{aligned}$$

where x_p denotes the p^{th} element of the vector $\mathbf{x}$, $\mathbf{H}^k$ is the BFGS approximation to the Hessian matrix of the high–fidelity function β at $\mathbf{x}^k$, $\hat{\mathbf{H}}^k$ is the BFGS approximation to the Hessian matrix of the low–fidelity function $\hat{\beta}$ at $P(\mathbf{x}^k)$, and H^k_{pq} denotes the pq^{th} element of the matrix $\mathbf{H}^k$.

For each subproblem k, Equation (2.29) computes the difference between the value of the high–fidelity function and the low–fidelity function at the center of the trust region. Using the chain rule, Equation (2.30) computes the difference between the gradient of the high–fidelity function and the gradient of the low–fidelity function at the center of the trust region, where the gradients are computed with respect to the high–fidelity design vector $\mathbf{x}$. The second term in (2.30) therefore requires the Jacobian of the mapping, $\frac{\partial \hat{x}_j}{\partial x_p}$. Similarly, Equation (2.32) computes the difference between the BFGS approximation of the Hessian matrices of the high–fidelity and low–fidelity functions at the center of the trust region. Once again, derivatives are required with respect to $\mathbf{x}$ and are computed using the chain rule.

Multiplicative Corrections

For some low–fidelity function $\hat{\beta}(\hat{\mathbf{x}})$, the corresponding high–fidelity function $\beta(\mathbf{x})$, and a mapping $\hat{\mathbf{x}} = P(\mathbf{x})$, the k^{th} multiplicative–corrected surrogate is defined as

$$\tilde{\beta}^k(\mathbf{x}) = \hat{\beta}\left(P(\mathbf{x})\right) B^k(\mathbf{x}). \tag{2.33}$$

In order to obtain quasi–second–order consistency between $\tilde{\beta}^k(\mathbf{x}^k)$ and $\beta(\mathbf{x}^k)$, we define the correction function, $B^k(\mathbf{x})$, using a quadratic Taylor series expansion of the quotient $\mathcal{B}(\mathbf{x})$ of the two functions β and $\hat{\beta}$, about the point $\mathbf{x}^k$:

$$B^k(\mathbf{x}) = \mathcal{B}(\mathbf{x}^k) + \nabla_{\mathbf{x}}\mathcal{B}(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^k)^T \nabla^2_{\mathbf{x}}\mathcal{B}(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k). \tag{2.34}$$

The elements in this expansion are calculated using

$$\mathcal{B}(\mathbf{x}^k) = \frac{\beta(\mathbf{x}^k)}{\hat{\beta}(P(\mathbf{x}^k))}, \tag{2.35}$$

$$\frac{\partial \mathcal{B}(\mathbf{x}^k)}{\partial x_i} = \frac{\hat{\beta}(P(\mathbf{x}^k))\frac{\partial \beta(\mathbf{x}^k)}{\partial x_i} - \beta(\mathbf{x}^k)\sum_{p=1}^{\hat{n}} \frac{\partial \hat{\beta}(P(\mathbf{x}^k))}{\partial \hat{x}_p}\frac{\partial \hat{x}_p}{\partial x_i}}{\hat{\beta}(P(\mathbf{x}^k))^2}, \tag{2.36}$$

$$\begin{aligned}\frac{\partial^2 \mathcal{B}(\mathbf{x}^k)}{\partial x_i \partial x_j} &= \frac{1}{\hat{\beta}(P(\mathbf{x}^k))}\frac{\partial^2 \beta(\mathbf{x}^k)}{\partial x_i \partial x_j} \\ &- \frac{1}{\hat{\beta}^2(P(\mathbf{x}^k))}\sum_{p=1}^{\hat{n}}\left[\frac{\partial \hat{\beta}(P(\mathbf{x}^k))}{\partial \hat{x}_p}\left(\frac{\partial \hat{x}_p}{\partial x_j}\frac{\partial \beta(\mathbf{x}^k)}{\partial x_i} + \frac{\partial \hat{x}_p}{\partial x_i}\frac{\partial \beta(\mathbf{x}^k)}{\partial x_j}\right) + \beta(\mathbf{x}^k)\frac{\partial \hat{\beta}(P(\mathbf{x}^k))}{\partial \hat{x}_p}\frac{\partial^2 \hat{x}_p}{\partial x_i \partial x_j}\right] \\ &+ \frac{\beta(\mathbf{x}^k)}{\hat{\beta}^2(P(\mathbf{x}^k))}\sum_{p=1}^{\hat{n}}\sum_{q=1}^{\hat{n}}\left(2\frac{\partial \hat{\beta}(P(\mathbf{x}^k))}{\partial \hat{x}_p}\frac{\partial \hat{\beta}(P(\mathbf{x}^k))}{\partial \hat{x}_q}\frac{\partial \hat{x}_p}{\partial x_i}\frac{\partial \hat{x}_q}{\partial x_j} - \hat{\beta}(P(\mathbf{x}^k))\frac{\partial^2 \hat{\beta}(P(\mathbf{x}^k))}{\partial \hat{x}_p \partial \hat{x}_q}\frac{\partial \hat{x}_p}{\partial x_i}\frac{\partial \hat{x}_q}{\partial x_j}\right).\end{aligned} \tag{2.37}$$

2.1.4 Computing the trial step

After the generation of the surrogate, Step 2 in Algorithm 1 is the computation of the trial step. As stated in Section 2.1.1, this step must meet the fraction of Cauchy decrease condition on the surrogate. In order for the TRMM algorithm to meet the goal of a reduction in computational time, the step calculation must use only the surrogate, and not rely on the high–fidelity model.

While the algorithm will converge as long as each step satisfies the FCD condition, it achieves superior convergence rates if the surrogate reduction is larger. One approach is to decrease the surrogate as much as possible while remaining within the trust region. To this end, the algorithm generates the k^{th} trial step $\mathbf{s}^k$ by solving the trust–region subproblem

$$\begin{aligned} \min_{\mathbf{s}} \quad & \tilde{f}(\mathbf{x}^k + \mathbf{s}) \\ \text{Subject to} \quad & \|\mathbf{s}\| \leq \Delta^k. \end{aligned} \tag{2.38}$$

This finds a step that minimizes the surrogate over the trust region. The norm in Equation (2.2) can be any norm uniformly equivalent to the Euclidean 2–norm. Such norms include the 2–norm itself, the 1–norm and the infinity–norm. The 2–norm is preferred for computational efficiency of the solution of the trust region subproblem (2.38), and the infinity norm is trivial to apply when simple bounds on the design variables exist.

The trust–region subproblem can be solved using any standard constrained optimization method. In this work the algorithm uses a sequential quadratic programming method. Its solution is accelerated if the gradient of the surrogate is available.

2.2 Constrained Algorithms

To this point, the algorithm introduced has been for the solution of unconstrained optimization. The goal of constrained TRMM is to solve the design problem introduced in Chapter 1 as Problem (1.6),

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{Subject to} \quad & \mathbf{c}(\mathbf{x}) \leq 0, \end{aligned} \tag{2.39}$$

where $f \in \mathbb{R}$ represents the scalar objective to be minimized and $\mathbf{x} \in \mathbb{R}^n$ is the vector of n design variables that describe the design, and $\mathbf{c} \in \mathbb{R}^m$ is the vector of constraint functions.

There are a number of methods of solving problem (2.39) using TRMM. First, the general constrained TRMM algorithm will be explained, and then each of four methods will be described in more detail.

2.2.1 General Algorithm

A general constrained TRMM method consists of the following key steps on each iteration k. For subproblem k, we denote the center of the trust region by $\mathbf{x}^k$ and the radius of the trust region by Δ^k.

Algorithm 2: General Constrained TRMM method

- **Step 0: Initialization** Choose an initial point $\mathbf{x}^0$ and an initial trust–region radius $\Delta^0 > 0$. Set $k = 0$. Choose constants $\eta > 0$, $0 < c_1 < 1$, $c_2 > 1$, $0 < r_1 < r_2 < 1$, $\Delta^* \geq \Delta^0$. Choose any other initial values or constants required by the constraint–management method, such as initial values for Lagrange multipliers, penalty parameters, and the penalty parameter decrease constant.

- **Step 1: Surrogate definition** Choose an appropriate surrogate of the high–fidelity model. The constrained methods differ in the surrogates they create. Provable convergence of these methods requires first–order consistency between the surrogate and the high–fidelity model at $\mathbf{x}^k$, the center of the trust region.

- **Step 2: Problem creation** Create the k^{th} subproblem, which consists of an optimization problem using the surrogate and a constraint that the solution remain within the trust region, i.e. $\|\mathbf{x} - \mathbf{x}^k\|_2 \leq \Delta^k$. The constrained methods differ in the formulation of the subproblem, but all include the trust–region constraint.

- **Step 3: Step calculation** Approximately solve the k^{th} subproblem in order to generate a trial step, $\mathbf{s}^k$. It is required that the step satisfy some variant of the FCD condition.

- **Step 4: Acceptance or rejection of the trial point** Compute $f(\mathbf{x}^k + \mathbf{s}^k)$. Accept or reject the trial step $\mathbf{x}_*^k$, according to some step acceptance criterion. If the trial step is accepted, $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$. If the trial step is rejected, $\mathbf{x}^{k+1} = \mathbf{x}^k$. Two step acceptance criteria are described in Section 2.2.1.

- **Step 4: Trust–region radius update** Calculate the trust–region ratio

$$\rho^k = \frac{\xi(\mathbf{x}^k) - \xi(\mathbf{x}^k + \mathbf{s}^k)}{\xi(\mathbf{x}^k) - \tilde{\xi}^k(\mathbf{x}^k + \mathbf{s}^k)}, \tag{2.40}$$

  where $\xi(\mathbf{x})$ is a merit function and $\tilde{\xi}$ is a surrogate of that merit function. The definition of the merit function varies between methods. Set

$$\Delta^{k+1} = \begin{cases} c_1\|\mathbf{s}^k\| & \text{if } \rho^k < r_1, \\ \min(c_2\Delta^k, \Delta^*) & \text{if } \rho^k > r_2, \\ \|\mathbf{s}^k\| & \text{otherwise.} \end{cases} \tag{2.41}$$

- **Update other algorithm parameters** Calculate new values for any parameters needed by the constraint–management method, such as Lagrange multipliers, penalty parameters, and intermediate constraint tolerances. These parameters change somewhat from method to method. Increment k by 1 and go to Step 1.

Constrained trust–region methods differ from one another in three key aspects: the surrogate subproblem solved to generate the trial step, the step acceptance criterion, and the merit function used to size the trust region. Table 2.1 shows four major classes of methods — augmented Lagrangian methods, direct surrogate optimization, SQP–like methods, and MAESTRO — and highlights their key differences.

Method	Minimize	Subject to	Merit function	Acceptance
Augmented Lagrangian	Surrogate of augmented Lagrangian	Trust region constraint only	Augmented Lagrangian	Augmented Lagrangian
Direct surrogate optimization	Surrogate of objective function	Surrogate of constraints	Augmented Lagrangian	Filter
SQP–like	Surrogate of Lagrangian	Linearized high–fidelity constraints	Lagrangian	Filter
MAESTRO	1. Norm of constraints 2. Objective	1. Constraint trust region 2. Constraints constant	1. Norm of constraints 2. Objective	1. Norm of constraints 2. Objective

Table 2.1: Four major classes of constrained trust–region methods. Shown are the objective and constraints of the trust–region subproblem, the merit function used to manage the radius of the trust region, and the trial step acceptance criterion.

Step Acceptance Criteria

As shown in Table 2.1, two methods can be used to determine whether to accept a trial point $\mathbf{x}_*^k$ generated by the solution of the subproblem. The first is identical to the acceptance criterion of the unconstrained method, that is, to use the trust–region ratio ρ, calculated from the merit function ξ. In this case,

$$\mathbf{x}^{k+1} = \begin{cases} \mathbf{x}_*^k & \text{if } \rho^k \geq \eta \\ \mathbf{x}^k & \text{if } \rho^k < \eta, \end{cases} \tag{2.42}$$

where η is a small positive number.

The second method is a filter method. This uses dominance, a concept borrowed from multiobjective optimization [67]. A point $\mathbf{x}^1$ dominates a point $\mathbf{x}^2$ if both the following conditions are satisfied:

$$\begin{aligned} f(\mathbf{x}^1) &\leq f(\mathbf{x}^2), \\ \|\mathbf{c}^+(\mathbf{x}^1)\|_2 &\leq \|\mathbf{c}^+(\mathbf{x}^2)\|_2, \end{aligned} \tag{2.43}$$

where $\mathbf{c}^+(\mathbf{x})$ is a vector with elements defined by

$$c_i^+(\mathbf{x}) = \max(0, c_i(\mathbf{x})). \tag{2.44}$$

A filter is a set of points, none of which dominate any other. In a filter method, the initial filter is empty. A trial point $\mathbf{x}_*^k$ is accepted, and added to the filter, if it is not dominated by any point in the filter. If any element of the filter dominates the trial point, the trial point is rejected and not added to the filter. Significant detail on filter methods is available in Chapter 5 of [22].

2.2.2 Augmented Lagrangian Method

One method to solve constrained optimization problems is to formulate an augmented Lagrangian of the problem and solve the resulting unconstrained problem. The first constrained optimization method combines this approach with the unconstrained TRMM method of section 2.1. The algorithm is modified from Ref. [59], with the construction of the response surface in that work replaced by the correction of the low–fidelity model to be consistent with the high–fidelity model.

The high–fidelity augmented Lagrangian is given by

$$L(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) + \lambda^T \mathbf{c}(\mathbf{x}) + \frac{1}{2\mu} \sum_{i=1}^{m} \left(\Psi_i(\mathbf{x}, \lambda, \mu)\right)^2, \tag{2.45}$$

where

$$\Psi_i(\mathbf{x}, \lambda, \mu) = \max\left(c_i(\mathbf{x}), -\mu\lambda_i\right) \quad i = 1, \ldots, m, \tag{2.46}$$

and μ is a penalty parameter.

Algorithm 3: Augmented Lagrangian TRMM optimization method

The augmented Lagrangian algorithm is as follows

- **Step 0: Initialization.** Choose an initial guess $\mathbf{x}^0$, a vector of Lagrange multiplier estimates λ^0, an initial penalty parameter $\mu^0 < 1$, initial convergence tolerances $\omega^0 > 0$ and $\eta^0 > 0$, and the penalty parameter decrease coefficient $\tau < 1$. Set $k = 0$.

- **Step 1: Create a surrogate for the augmented Lagrangian.** There are two methods for creating a surrogate for an augmented Lagrangian, both described in Section 2.2.6. Each of them results in a surrogate which is zeroth– and first– order consistent with the high–fidelity function.

- **Step 2: Inner iteration.** Approximately solve the problem

$$\min_{\mathbf{x}} L(\mathbf{x}, \lambda^k, \mu^k) \tag{2.47}$$

  using Algorithm 1, the unconstrained variable–fidelity TRMM. The inner algorithm stops if

$$\|\nabla_{\mathbf{x}} L(\mathbf{x}^k, \lambda^k, \mu^k)\| \leq \omega^k. \tag{2.48}$$

  If $\|\mathbf{c}(\mathbf{x}^{k+1})\| \leq \eta^k$ proceed to Step 3. Otherwise, proceed to Step 4.

- **Step 3: Update Lagrange multiplier estimates.** Choose new Lagrange multiplier estimates λ^{k+1} by minimizing the residual in the Karush–Kuhn–Tucker optimality conditions. That is, we solve the problem

$$\begin{aligned} \lambda^{k+1} = \arg\min_{\lambda} \quad & \|\nabla_{\mathbf{x}} f(\mathbf{x}^k) + \textstyle\sum_{i \in S^k} \lambda_i \nabla_{\mathbf{x}} c_i(\mathbf{x}^k)\|_2^2 \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned} \tag{2.49}$$

  where S^k is the set of active constraints for subproblem k. The optimization problem (2.49) is a nonnegative least–squares constraint problem, which is solved using the method of Ref. [49]. Also set $\omega^{k+1} = \omega^k \mu^k$ and $\eta^{k+1} = \eta^k (\mu^k)^{0.9}$. Go to Step 1.

- **Step 4: Reduce the penalty parameter.** Set $\lambda^{k+1} = \lambda^k$ and

$$\mu^{k+1} = \tau \mu^k. \tag{2.50}$$

Also set $\omega^{k+1} = \omega^k \mu^k$ and $\eta^{k+1} = \eta^k (\mu^k)^{0.1}$. Go to Step 1.

2.2.3 Direct Surrogate Optimization

Direct Surrogate Optimization (DSO) solves a series of subproblems that minimize the surrogate objective subject to the surrogate constraints [28]. It implements the general algorithm, as described in Section 2.2.1 above. In Step 1, the surrogates generated are $\tilde{f}^k(\mathbf{x})$, a surrogate for $f(\mathbf{x})$, and $\tilde{\mathbf{c}}_i{}^k(\mathbf{x})$, $i = 1 \ldots m$, a surrogate for each constraint $\mathbf{c}_i(\mathbf{x})$. These follow the general correction method described in section 2.1.2 for single–parameterization problems or 2.1.3 for variable–parameterization problems.

The subproblem created in Step 2 and solved in Step 3 is

$$\begin{array}{rl} \min\limits_{\mathbf{x}} & \tilde{f}^k(\mathbf{x}) \\ \text{subject to} & \tilde{\mathbf{c}}^k(\mathbf{x}) \le 0 \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \le \Delta^k. \end{array} \tag{2.51}$$

The merit function, used in the step acceptance of Step 4, is the augmented Lagrangian, defined in Equation (2.45). The trust–region ratio is therefore defined by Equation (2.40), with $\xi = L$, and the size of the trust region is managed using the (2.41). The Lagrange multipliers and penalty parameters are also updated using Equations (2.49) and (2.50) in order to calculate the merit function.

2.2.4 SQP–like method

The SQP–like method is modified from Ref. [7]. It is similar to sequential quadratic programming (SQP) in that on each subproblem it minimizes a surrogate of the Lagrangian subject to linear approximations to the high–fidelity constraints. It follows the general algorithm in section 2.2.1. As for DSO, the required surrogates generated in Step 1 are $\tilde{f}^k(\mathbf{x})$, and $\tilde{\mathbf{c}}^k(\mathbf{x})$.

The subproblem generated in Step 2 and solved in Step 3 is

$$\begin{array}{rll} \min\limits_{\mathbf{x}} & \tilde{\mathcal{L}}^k(\mathbf{x}, \lambda^k) & \\ \text{subject to} & \mathbf{c}^k(\mathbf{x}_c^k) + \nabla_{\mathbf{x}} \mathbf{c}^k(\mathbf{x})^T(\mathbf{x} - \mathbf{x}_c^k) \leq 0 & \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty & \leq \Delta^k, \end{array} \tag{2.52}$$

The Lagrangian is defined as

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{c}(\mathbf{x}), \tag{2.53}$$

where λ is the vector of Lagrange multipliers, and $\tilde{\mathcal{L}}^k$ is a surrogate to the Lagrangian using

$$\tilde{\mathcal{L}}^k(\mathbf{x}, \lambda) = \tilde{f}^k(\mathbf{x}) + \lambda^T \tilde{\mathbf{c}}^k(\mathbf{x}). \tag{2.54}$$

The step acceptance criteria in Step 4 and the trust–region radius update in Step 5 use the Lagrangian as the merit function.

The Lagrange multipliers are updated by solving the nonnegative least–squares constraint problem

$$\begin{array}{rl} \min\limits_{\lambda} & \|\nabla_{\mathbf{x}} f(\mathbf{x}_0^k) + \sum_{i \in S} \lambda_i \nabla_{\mathbf{x}} c_i(\mathbf{x}_0^k)\|_2^2 \\ \text{subject to} & \lambda \geq 0, \end{array} \tag{2.55}$$

where S is the set of active constraints, using the method on page 161 of Ref. [49].

2.2.5 MAESTRO

The MAESTRO method was developed by Alexandrov in order to solve multidisciplinary design optimization (MDO) problems that include constraints from a number of disjoint disciplines, and that are possibly calculated on separate nodes of a distributed system. It is described without model management in Ref. [3] and with model management in Ref. [4]. The major difference between MAESTRO and the other trust–region methods is that MAESTRO solves two problems on each iteration: one to minimize the norm of the constraints and another to minimize the objective function. The step used is then the sum of the steps found from the solutions to those two problems. Separate trust regions are maintained for the objectives and for the constraints.

The MAESTRO method requires only equality constraints; therefore the inequality constraints are converted to equality constraints using squared slack variables. The i^{th} inequality constraint, c_i, is converted to an equality constraint, h_i, using

$$h_i(\mathbf{x}) = c_i(\mathbf{x}) + z_i^2 = 0, \tag{2.56}$$

where z_i is the corresponding slack variable. The vector of resulting equality constraints is denoted $\mathbf{h}(\mathbf{y})$. The new set of design variables is then the concatenated original variables and slack variables

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix}. \tag{2.57}$$

Each subproblem is broken down into two parts. The first part minimizes the Euclidean norm of the surrogates of the constraints, subject to the bounds on the solution provided by the trust region:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \| \tilde{\mathbf{h}}^k(\mathbf{y}) \|_2 \\ \text{subject to} \quad & \| \mathbf{y} - \mathbf{y}^k \| \quad \le \Delta^k_{con}, \end{aligned} \tag{2.58}$$

$$\tag{2.59}$$

where $\tilde{\mathbf{h}}^k$ is the surrogate model of $\mathbf{h}$ for the k^{th} subproblem, $\mathbf{y}^k$ is the vector of concatenated variable values at the center of the trust region, and Δ^k_{con} is the trust–region size for the k^{th} constraint subproblem. The solution to this problem is denoted $\mathbf{y}^k_1$. The constraint trust–region ratio is calculated as:

$$\rho_{con}^k = \frac{\| \mathbf{h}(\mathbf{y}^k) \|_2 - \| \mathbf{h}(\mathbf{y}_1^k) \|_2}{\| \tilde{\mathbf{h}}(\mathbf{y}^k) \|_2 - \| \tilde{\mathbf{h}}^k(\mathbf{y}_1^k) \|_2}. \tag{2.60}$$

The new solution $\mathbf{y}_1^k$ is accepted if $\rho_{con}^k > \eta$ and rejected otherwise. The constraint trust region radius Δ_{con}^k is updated using (2.41) with ρ_{con}^k as the trust–region ratio.

The second part of the subproblem minimizes the objective, subject to the conditions that the solution remains within the objective trust region and that improvement in the constraints is maintained:

$$\begin{aligned} \min_{\mathbf{y}} \quad & \tilde{\mathbf{f}}^k(\mathbf{y}) \\ \text{subject to} \quad & \| \mathbf{y} - \mathbf{y}_1^k \| \leq \Delta_{obj}^k, \\ & \tilde{\mathbf{h}}^k(\mathbf{y}) = \tilde{\mathbf{h}}^k(\mathbf{y}_1^k). \end{aligned} \tag{2.61}$$

The solution to the problem (2.61) is denoted $(\mathbf{y}_2^k)$. The objective trust–region ratio is calculated as

$$\rho_{obj}^k = \frac{f(\mathbf{y}_1^k) - f(\mathbf{y}_2^k)}{\tilde{f}(\mathbf{y}_1^k) - \tilde{f}^k(\mathbf{y}_2^k)}. \tag{2.62}$$

The new solution $\mathbf{y}_2^k$ is accepted if $\rho_{obj}^k > \eta$ and rejected otherwise, and the objective trust region size Δ_{obj}^k is updated using (2.41) with ρ_{obj}^k as the trust–region ratio.

2.2.6 Surrogates of Lagrangian functions

When using a method that requires surrogates of the Lagrangian or augmented Lagrangian function, there are two approaches to forming a corrected surrogate. The distinction between these approaches was suggested by Rodriguez et al. [59].

In the first approach, the entire augmented Lagrangian is approximated with a single surrogate. That is, an surrogate $\tilde{L}$ of L is created using a BFGS quasi–second–order additive correction to the augmented Lagrangian of the low–fidelity model, $\hat{L}$, defined by

$$\hat{L}(\hat{\mathbf{x}}, \lambda, \mu) = \hat{f}(\hat{\mathbf{x}}) + \lambda^T \hat{\mathbf{c}}(\hat{\mathbf{x}}) + \frac{1}{2\mu} \sum_{i=1}^{m} \left(\hat{\Psi}_i^k(\hat{\mathbf{x}}, \lambda, \mu) \right)^2, \tag{2.63}$$

where

$$\hat{\Psi}_i(\hat{\mathbf{x}}, \lambda, \mu) = \max\left(\hat{c}_i(\hat{\mathbf{x}}), -\mu\lambda_i\right), \; i = 1, \ldots, m. \tag{2.64}$$

The surrogate augmented Lagrangian for each subproblem is then obtained by correcting the low–fidelity augmented Lagrangian,

$$\tilde{L}^k(\mathbf{x}, \lambda, \mu) = \hat{L}(P(\mathbf{x}), \lambda, \mu) + A^k(\mathbf{x}, \lambda, \mu), \tag{2.65}$$

where A^k is defined in Equations (2.28) through (2.32), using $\beta(\mathbf{x}) = L(\mathbf{x}, \lambda^k, \mu^k)$ and $\hat{\beta}(\hat{\mathbf{x}}) = \hat{L}(\hat{\mathbf{x}}, \lambda^k, \mu^k)$.

The second approach to forming the surrogate augmented Lagrangian uses separate corrected surrogates of the objective and of the constraints. The surrogate augmented Lagrangian is therefore given by

$$\tilde{L}^k(\mathbf{x}, \lambda, \mu) = \tilde{f}^k(\mathbf{x}) + \lambda^T \tilde{\mathbf{c}}^k(\mathbf{x}) + \frac{1}{2\mu}\sum_{i=1}^{m}\left(\tilde{\Psi}_i^k(\mathbf{x}, \lambda, \mu)\right)^2, \tag{2.66}$$

where

$$\tilde{\Psi}_i^k(\mathbf{x}, \lambda, \mu) = \max\left(\tilde{c}_i^k(\mathbf{x}), -\mu\lambda_i\right), \quad i = 1, \ldots, m, \tag{2.67}$$

and $\tilde{f}^k$ and $\tilde{c}^k$ are obtained using additive corrections,

$$\tilde{f}^k(\mathbf{x}) = \hat{f}(P(\mathbf{x})) + A^k(\mathbf{x}), \tag{2.68}$$

and

$$\tilde{c}_i^k(\mathbf{x}) = \hat{c}_i(P(\mathbf{x})) + B_i^k(\mathbf{x}), \; i = 1 \ldots m. \tag{2.69}$$

The correction function A^k is defined in Equations (2.28) through (2.32), using $\beta(\mathbf{x}) = f(\mathbf{x})$ and $\hat{\beta}(\hat{\mathbf{x}}) = \hat{f}(P(\mathbf{x}))$. Each correction function B_i^k is defined similarly, with $\beta(\mathbf{x}) = c_i(\mathbf{x})$ and $\hat{\beta}(\hat{\mathbf{x}}) = \hat{c}_i(P(\mathbf{x}))$, ensuring quasi–second–order consistency between the surrogate constraints and the high–fidelity constraints at the center of the trust region.

Creating a surrogate $\tilde{\mathcal{L}}$ for the Lagrangian $\mathcal{L}$ can also be accomplished via two methods strictly analogous to those described above for the augmented Lagrangian.

2.3 Choice of Design Space

Solving the subproblems in the low–fidelity space has the advantage of reducing the dimension in which the algorithm is operating. This can be advantageous when, for instance, finite differences are required in the calculation of gradients. Because the gradient is in the low–fidelity space it requires a smaller number of high–fidelity function evaluations to approximate. In addition, the trust–region subproblem is in a lower–dimensional space and can be solved more efficiently.

An extension of the framework to variable design spaces requires a mapping from the low–fidelity design space to the high–fidelity design space, that is

$$\mathbf{x} = Q(\hat{\mathbf{x}}). \tag{2.70}$$

Define $\mathbf{J}$ as the Jacobian matrix of the mapping Q, or elementwise

$$J_{ij} = \frac{\partial x_i}{\partial \hat{x}_j}. \tag{2.71}$$

Define $\mathbf{T}(\hat{\mathbf{x}})$ elementwise as

$$T_{ij}(\hat{\mathbf{x}}) = \frac{d^2 f}{d\hat{x}_i d\hat{x}_j} = \sum_{p=1}^{n} \frac{\partial f}{\partial x_p} \frac{\partial^2 x_p}{\partial \hat{x}_i \partial \hat{x}_j} + \sum_{p=1}^{n} \sum_{q=1}^{n} \frac{\partial x_p}{\partial \hat{x}_i} \frac{\partial x_q}{\partial \hat{x}_j} \frac{\partial^2 f}{\partial x_p \partial x_q}, \tag{2.72}$$

which is expanded using the chain rule.

2.3.1 Algorithm in the low–fidelity space

This algorithm solves the unconstrained optimization problem 2.1. This differs from the original TRMM method described in Section 2.1 in that the trust–region subproblem is defined in the low–fidelity space. That is, the center of the trust region $\hat{\mathbf{x}}^k$ and the step $\hat{\mathbf{s}}^k$ are of dimension $\hat{n}$ rather than n.

Algorithm 4: TRMM optimization, with the trust–region subproblem solved in the low–fidelity space

Step 0: Initialization Choose an initial point $\hat{\mathbf{x}}^0$ and an initial trust–region radius $\hat{\Delta}^0 > 0$. Set $k = 0$. Choose constants $\eta > 0$, $0 < c_1 < 1$, $c_2 > 1$, $0 < r_1 < r_2 < 1$, $\hat{\Delta}^* \geq \hat{\Delta}^0$.

Step 1: Surrogate definition Choose a surrogate $\tilde{f}^k(\hat{\mathbf{x}})$ that satisfies $\tilde{f}^k(\hat{\mathbf{x}}^k) = f(Q(\mathbf{x}^k))$ and $\nabla_{\hat{\mathbf{x}}}\tilde{f}^k(\hat{\mathbf{x}}^k) = \mathbf{J}^T\nabla_{\mathbf{x}}f(Q(\hat{\mathbf{x}}^k))$. Please note that Chapter 3 will detail the construction of this surrogate.

Step 2: Step calculation Compute a step $\hat{\mathbf{s}}^k$ such that $\|\hat{\mathbf{s}}^k\| \leq \hat{\Delta}^k$ and that satisfies the fraction of Cauchy decrease condition (described in Equation (2.80) below).

Step 3: Acceptance or rejection of the trial point Compute $f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k))$ and define

$$\rho^k = \frac{f(Q(\hat{\mathbf{x}}^k)) - f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k))}{f(Q(\hat{\mathbf{x}}^k)) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)}. \tag{2.73}$$

If $\rho^k \geq \eta$ define $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k$; otherwise define $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k$. Define a *successful* iterate as one in which the step is accepted; otherwise it is an unsuccessful iterate.

Step 4: Trust–region radius update. Set

$$\hat{\Delta}^{k+1} = \begin{cases} c_1\|\hat{\mathbf{s}}^k\| & \text{if } \rho^k < r_1, \\ \min(c_2\hat{\Delta}^k, \hat{\Delta}^*) & \text{if } \rho^k > r_2, \\ \|\hat{\mathbf{s}}^k\| & \text{otherwise.} \end{cases} \tag{2.74}$$

Increment k by 1 and go to Step 1.

2.3.2 Proof of Convergence

We require the method to converge to a stationary point of the high–fidelity function. Achieving convergence when solving the subproblems in the low–fidelity space requires a number of assumptions, some of which are identical to some listed in [22], some of which are analogous to some in [22], and two of which, being assumptions on the mapping, are new.

Assumptions for Provable Convergence

The assumptions required for the algorithm to be provably convergent to a minimum of the high–fidelity function are:

1. The high–fidelity function function $f(\mathbf{x})$ is twice continuously differentiable on $\mathbb{R}^n$.

2. The function $f(\mathbf{x})$ is bounded below on $\mathbb{R}^n$, that is, there exists a constant κ_{lbf} such that, for all $\mathbf{x} \in \mathbb{R}^n$,
$$f(\mathbf{x}) \geq \kappa_{lbf}. \tag{2.75}$$

3. The norm of the Hessian matrix $\nabla^2_{\mathbf{x}} f(\mathbf{x})$ of the high–fidelity function must be bounded, independent of k, for all values that lie between any two iterates of the algorithm. This is satisfied, for instance, by the requirement that over the level set $\{\mathbf{x} \in \mathbb{R}^n | f(\mathbf{x}) \leq f(\mathbf{x}^0)\}$,
$$\|\nabla^2_{\mathbf{x}} f(\mathbf{x})\| \leq \kappa_{ufh}. \tag{2.76}$$

4. The mapping $\mathbf{x} = Q(\hat{\mathbf{x}})$ is twice continuously differentiable over $\mathbb{R}^{\hat{n}}$.

5. The matrix $\mathbf{T}$ defined in (2.72) is bounded on $\mathbb{R}^{\hat{n}}$. That is,
$$\|\mathbf{T}(\hat{\mathbf{x}})\| \leq \kappa_{ubT}. \tag{2.77}$$

6. On each iteration, the surrogate $\tilde{f}^k(\hat{\mathbf{x}}^k)$ is zeroth– and first–order consistent with the high–fidelity function, That is, $\tilde{f}^k(\hat{\mathbf{x}}^k) = f(Q(\hat{\mathbf{x}}^k))$ and $\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k) = \mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))$.

7. The low–fidelity function function $\tilde{f}(\hat{\mathbf{x}})$ is twice continuously differentiable on $\mathbb{R}^{\hat{n}}$.

8. On each iteration, the Hessian matrix $\nabla^2 \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}})$ of the surrogate is bounded

for all $\|\hat{\mathbf{s}}\| \leq \hat{\Delta}^k$. That is, over the trust region,

$$\|\nabla^2_{\mathbf{x}} \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}})\| \leq \kappa_{uah} - 1, \tag{2.78}$$

where κ_{uah} is a constant independent of k and 1 is subtracted for later notational convenience.

9. On each trust region, over the entire trust region, the mapping must be twice continuously differentiable.

10. The mapping must be defined in such a way that, for some number $\alpha > 0$, independent of k,

$$\|\mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\| \geq \alpha \|\nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\|. \tag{2.79}$$

The mapping method must be dynamic in order to satisfy this condition. Differences between static and dynamic mappings will be defined in Chapter 3.

11. Each trial step must satisfy the fraction of Cauchy decrease (FCD) condition. That is, there must exist numbers $\zeta > 0$ and $C > 0$ independent of k such that

$$\tilde{f}^k(\hat{\mathbf{x}}^k) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) \geq \zeta \|\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k)\| \min\left(\Delta^k, \frac{\|\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k)\|}{C}\right). \tag{2.80}$$

Gratton et al. have developed a similar algorithm for use in optimization of systems governed by discretized approximations to infinite–dimensional partial differential equations [37]. They also determine that Assumption 10 is required in order to ensure convergence. Algorithm 4 enforces Assumption 10 at every trust–region iteration through use of a dynamic mapping. The algorithm in [37] evaluates whether Assumption 10 is satisfied at each iteration. If it is satisfied the algorithm uses the multilevel algorithm; otherwise it uses a conventional second–order Taylor series surrogate defined in the high–fidelity space.

Bound on the Approximation Error

Define a point $\hat{\mathbf{x}}^k$ in the low–fidelity space and a map $\mathbf{x} = Q(\hat{\mathbf{x}})$. If the gradient of the high–fidelity function $\nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))$ is zero, then the algorithm has found a stationary point. If it is non–zero, then by Assumption 6, $\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k) = \mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))$. Then, by Assumption 10, $\|\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k)\| = \|\mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\| \geq \alpha \|\nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\| > 0$. Combining this with the FCD Assumption 11, we have that $\mathbf{s}^k \neq 0$ and $\tilde{f}^k(\mathbf{x}^k + \mathbf{s}^k) < \tilde{f}^k(\mathbf{x}^k)$. Therefore, as long as the iterate is not a stationary point, Equation (2.73) for the trust region ratio ρ^k is well–defined.

Using Assumptions 1 and 9, that is, that both the high–fidelity function $f(\mathbf{x})$ and the mapping $Q(\hat{\mathbf{x}})$ are continuously twice differentiable, the composition $f \circ Q$ is also continuously twice differentiable. Using this and Assumption 7, we apply the mean value theorem. There exists an $\hat{\xi}^k$ between $\hat{\mathbf{x}}^k$ and $\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k$ such that

$$f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)) = f(Q(\hat{\mathbf{x}}^k)) + (\hat{\mathbf{s}}^k)^T \mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k)) + \frac{1}{2} (\hat{\mathbf{s}}^k)^T \mathbf{T}(\hat{\xi}^k) \hat{\mathbf{s}}^k \tag{2.81}$$

and a $\hat{\chi}^k$ between $\hat{\mathbf{x}}^k$ and $\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k$ such that

$$\tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) = \tilde{f}^k(\hat{\mathbf{x}}^k) + (\hat{\mathbf{s}}^k)^T \nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k) + \frac{1}{2} (\hat{\mathbf{s}}^k)^T \nabla^2_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\chi}^k) \hat{\mathbf{s}}^k. \tag{2.82}$$

Subtracting Equation (2.82) from (2.81)

$$\begin{aligned} f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) &= f(Q(\hat{\mathbf{x}}^k)) - \tilde{f}^k(\hat{\mathbf{x}}^k) \\ &+ (\hat{\mathbf{s}}^k)^T \mathbf{J}^T \nabla_{\hat{\mathbf{x}}} f(Q(\hat{\mathbf{x}}^k)) - (\hat{\mathbf{s}}^k)^T \nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k) \\ &+ \frac{1}{2} \left((\hat{\mathbf{s}}^k)^T \mathbf{T}(\hat{\xi}^k) \hat{\mathbf{s}}^k - (\hat{\mathbf{s}}^k)^T \nabla^2 \tilde{f}^k(\hat{\chi}^k) \hat{\mathbf{s}}^k \right). \end{aligned} \tag{2.83}$$

Using Assumption 1,

$$f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) = \frac{1}{2} \left((\hat{\mathbf{s}}^k)^T \mathbf{T}(\hat{\xi}^k) \hat{\mathbf{s}}^k - (\hat{\mathbf{s}}^k)^T \nabla^2 \tilde{f}^k(\hat{\chi}^k) \hat{\mathbf{s}}^k \right). \tag{2.84}$$

Using the boundedness of the Hessian matrices of both the high–fidelity function and the surrogate (Assumptions 3 and 8), and the Cauchy–Schwarz inequality,

$$|f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)| \leq \frac{1}{2}(\kappa_{ufh} + \kappa_{uah} - 1)\|\hat{\mathbf{s}}^k\|^2. \tag{2.85}$$

Using the fact that the algorithm restricts the steps to within the trust region,

$$|f(Q(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)| \leq \frac{1}{2}(\kappa_{ufh} + \kappa_{uah} - 1)\|\Delta^k\|^2. \tag{2.86}$$

This shows that the approximation error decreases quadratically with the trust–region radius. Thus, we can show that as long as the current iterate is not a stationary point, there is a finite trust region size for which the step is accepted.

Trust–Region Size at which Trust Region is Expanded from a Non–Stationary Point

Suppose that the current iterate is not a stationary point. Therefore by Assumptions 6 and 10, $\|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\| > 0$. Suppose further that the trust region size

$$\hat{\Delta}^k \leq \|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\| \min\left(\frac{2(1-r_2)\zeta}{\kappa_{ubT} + \kappa_{uah}}, \frac{1}{C}\right), \tag{2.87}$$

where ζ and C are the parameters introduced in the FCD condition (2.80) and $0 < r_2 < 1$ is one of the algorithm parameters defining the trust region sizing decision process (2.74). Since

$$\hat{\Delta}^k \leq \frac{\|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\|}{C}, \tag{2.88}$$

the FCD condition (2.80) now becomes

$$\tilde{f}^k(\hat{\mathbf{x}}^k) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) \geq \zeta\|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\|\hat{\Delta}^k. \tag{2.89}$$

Subtracting 1 from Equation (2.73) and applying absolute values

$$|\rho - 1| = \frac{|\tilde{f}(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) - f^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)|}{|\tilde{f}^k(\hat{\mathbf{x}}^k) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)|}. \tag{2.90}$$

Using Equation (2.86) in the numerator and (2.89) in the denominator of Equation (2.90) results in

$$|\rho - 1| \leq \frac{|\frac{1}{2}(\hat{\Delta}^k)^2(\kappa_{ufh} + \kappa_{uah} - 1)|}{|\zeta\|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\|\hat{\Delta}^k|}. \tag{2.91}$$

All the terms in Equation (2.91) are positive. Simplifying,

$$|\rho - 1| \leq \frac{\hat{\Delta}^k(\kappa_{ufh} + \kappa_{uah} - 1)}{2\zeta\|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\|}. \tag{2.92}$$

Using Equation (2.74)

$$\hat{\Delta}^k \leq \frac{2(1 - r_2)\zeta\|\nabla_{\hat{\mathbf{x}}}\tilde{f}(\hat{\mathbf{x}}^k)\|}{\kappa_{ufh} + \kappa_{uah} - 1}, \tag{2.93}$$

we have

$$|\rho - 1| \leq 1 - r_2 \tag{2.94}$$

or

$$\rho \geq r_2. \tag{2.95}$$

Therefore, using the rules of equation (2.74), $\hat{\Delta}^{k+1} \geq \hat{\Delta}^k$. Note also that since $r_2 > \eta$, this shows that, given a small enough trust region, there will always be a successful step.

Lower Bound on Trust–Region Size

Now we will prove that the trust–region radius has a lower bound as long as the algorithm has not approached a stationary point too closely. To this end, we assume that we are not "too near" a stationary point, that is, there is a constant $\kappa_{lbg} > 0$ such that

$$\|\nabla_{\hat{\mathbf{x}}} f(\hat{\mathbf{x}}^k)\| \geq \kappa_{lbg} \tag{2.96}$$

for all k. The proof proceeds by contradiction. Assume that iteration k is the first iteration for which

$$\hat{\Delta}^{k+1} \leq r_1 \kappa_{lbg} \min\left(\frac{2(1 - r_2)\zeta}{\kappa_{ufT} + \kappa_{uah}}, \frac{1}{C}\right). \tag{2.97}$$

We know from equation (2.74) and Assumption (2.96) that

$$\hat{\Delta}^k \le \frac{\hat{\Delta}^{k+1}}{r_1} \le \|\nabla_{\hat{\mathbf{x}}} f(\hat{\mathbf{x}}^k)\| \min\left(\frac{2(1-r_2)\zeta}{\kappa_{ufT}+\kappa_{uah}}, \frac{1}{C}\right), \tag{2.98}$$

which we have just shown implies that $\hat{\Delta}^{k+1} \ge \hat{\Delta}^k$, contradicting the assumption that iteration k is the first to satisfy (2.97). Therefore, we know that for all k,

$$\hat{\Delta}^k \ge r_1 \kappa_{lbg} \min\left(2\frac{(1-r_2)\zeta}{\kappa_{ufT}+\kappa_{uah}}, \frac{1}{C}\right) = \kappa_{lbd}, \tag{2.99}$$

establishing a lower bound κ_{lbd} on the trust region size away from a stationary point.

Stationarity of Limit Point for Finitely Many Successful Iterations

If there is a finite sequence of successful iterates, there is a last successful iterate k_{last} and all iterates after that one are unsuccessful. Since all iterations after iteration k_{last} are unsuccessful, the trust–region update rules (2.74) imply that the sequence $\{\hat{\Delta}^k\}$ must therefore converge to zero. However, we showed in section 2.3.2 that as long as $\|\nabla_{\mathbf{x}} f(x_{k_{last}+1})\| > 0$, below a certain trust region size there will be a successful iteration. As $\{\hat{\Delta}^k\}$ converges towards zero, it will fall below that limit, resulting in an eventual successful iteration with $k > k_{last}+1$. This implies that $\|\nabla_{\mathbf{x}} f(x_{k_{last}+1})\| = 0$ and the method converges to a stationary point.

Existence of Stationary Point in Set of Limit Points

Now we will prove that at least one point of an infinite sequence of successful iterates must be a stationary point. This proof is by contradiction. Therefore, we assume that, for all k and some $\epsilon > 0$,

$$\|\nabla_{\mathbf{x}} f(\mathbf{x}^k)\| \ge \epsilon. \tag{2.100}$$

Consider a successful iteration with index k. Since the iteration was successful, we know that

$$f(Q(\hat{\mathbf{x}}^k)) - f(Q(\hat{\mathbf{x}}^{k+1})) \ge \eta(\tilde{f}^k(\hat{\mathbf{x}}^k) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k)) \tag{2.101}$$

and by the FCD condition (2.80)

$$\tilde{f}^k(\hat{\mathbf{x}}^k) - \tilde{f}^k(\hat{\mathbf{x}}^k + \hat{\mathbf{s}}^k) \geq \zeta \|\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k)\| \min\left(\hat{\Delta}^k, \frac{\nabla_{\hat{\mathbf{x}}} \tilde{f}^k(\hat{\mathbf{x}}^k)}{C}\right), \tag{2.102}$$

leading, along with Assumption 6 to

$$f(Q(\hat{\mathbf{x}}^k)) - f(Q(\hat{\mathbf{x}}^{k+1})) \geq \zeta\eta \|\mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\| \min\left(\hat{\Delta}^k, \frac{\|\mathbf{J}^T \nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\|}{C}\right). \tag{2.103}$$

Using Assumption 10

$$f(Q(\hat{\mathbf{x}}^k)) - f(Q(\hat{\mathbf{x}}^{k+1})) \geq \zeta\eta\alpha \|\nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\| \min\left(\hat{\Delta}^k, \frac{\alpha\|\nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\|}{C}\right), \tag{2.104}$$

which, using Equations (2.100) and (2.99) results in

$$f(Q(\hat{\mathbf{x}}^k)) - f(Q(\hat{\mathbf{x}}^{k+1})) \geq \zeta\eta\epsilon\alpha \min\left(\kappa_{lbd}, \frac{\alpha\epsilon}{C}\right) > 0. \tag{2.105}$$

If σ_k is the number of successful iterations up to iteration k, then

$$f(Q(\hat{\mathbf{x}}^0)) - f(Q(\hat{\mathbf{x}}^{k+1})) \geq \sigma_k \zeta\eta\epsilon\alpha \min\left(\kappa_{lbd}, \frac{\alpha\epsilon}{C}\right) \tag{2.106}$$

but, since in this case we have assumed that there are an infinite number of successful iterations, we have that

$$\lim_{k\to\infty} \sigma_k = \infty \tag{2.107}$$

leading to

$$\lim_{k\to\infty} \left(f(Q(\hat{\mathbf{x}}^0)) - f(Q(\hat{\mathbf{x}}^{k+1}))\right) = \infty, \tag{2.108}$$

which contradicts Assumption 2, that f is bounded below. Thus we have found a contradiction, and have shown that

$$\liminf_{k\to\infty} \|\nabla_{\mathbf{x}} f(Q(\hat{\mathbf{x}}^k))\| = 0. \tag{2.109}$$

2.4 Conclusion

This chapter described TRMM as applied to both unconstrained and constrained optimization, with the subproblem defined in the high–fidelity space. It also showed a proof of convergence of unconstrained optimization with the subproblem defined in the low–fidelity space. The constrained methods can also be combined with the lower–dimensional subproblems, though that has not been detailed here.

The algorithms described in this chapter show two levels of fidelity. It could be extended to use several analyses with a hierarchical set of fidelity models by nesting the algorithm. That is, when solving the surrogate problem, a model–management scheme could be used, with yet a lower level of geometric fidelity to speed up the optimization. This could be extended to any number of analysis models.

The variable–parameterization portions of this chapter assumed the availability of a mapping between the high–fidelity and low–fidelity design variables. The next chapter presents four mapping methods to use in conjunction with TRMM: space mapping, corrected space mapping, POD mapping, and a hybrid POD/space mapping.

Chapter 3

Mapping Methods

SBO methods have until now been applicable only to models in which both the high–fidelity model $f(\mathbf{x})$, $\mathbf{c}(\mathbf{x})$ and the low–fidelity model $\hat{f}(\hat{\mathbf{x}})$, $\hat{\mathbf{c}}(\hat{\mathbf{x}})$ are defined over the same space $\mathbf{x} = \hat{\mathbf{x}}$. In order to use a low–fidelity model with a different number of design variables as the high–fidelity function to be optimized, it is necessary to find a relationship between the two sets of design vectors, that is, $\hat{\mathbf{x}} = P(\mathbf{x})$ or $\mathbf{x} = Q(\hat{\mathbf{x}})$. In the first case, in which the trust–region subproblem is solved in the high–fidelity space, $\hat{f}(P(\mathbf{x}))$ is corrected to a surrogate for $f(\mathbf{x})$ and $\hat{\mathbf{c}}(P(\mathbf{x}))$ is corrected to a surrogate for $\mathbf{c}(\mathbf{x})$. In the second case, the subproblem is solved in the low–fidelity space, and $\hat{f}(\hat{\mathbf{x}})$ is corrected to a surrogate for $\mathbf{f}(Q(\hat{\mathbf{x}}))$ and $\hat{\mathbf{c}}(\hat{\mathbf{x}})$ is corrected to a surrogate for $c(Q(\hat{\mathbf{x}}))$.

In some cases, this design space mapping can be obvious and problem–specific. For instance, if the high– and low–fidelity models are the same set of physical equations, but on a fine and coarse grid, and the design vectors in each case are geometric parameters defined on that grid, the low–fidelity design vector can be a subset of the high–fidelity design vector, or the high–fidelity design vector can be an interpolation of the low–fidelity design vector. However, in other problems, there is no obvious mathematical relationship between the design vectors. In this case, an empirical mapping is needed. One example of a problem without an obvious mapping is the airfoil design problem described in Section 4.3. Another is the multifidelity supersonic business jet problem used by Choi et al. [21], described in Chapter 1.

This chapter presents four mapping methods, the last three of which are new: space mapping (SM), corrected space mapping, a mapping based on the proper orthogonal decomposition (POD), and a hybrid POD/SM mapping. The use of mapping to construct surrogates for TRMM algorithms is detailed, showing that the resulting surrogates are at least first–order consistent with the high–fidelity problem at the center of the trust region. A preliminary comparison of the mapping methodologies is then presented. Later chapters provide experimental comparisons.

3.1 Space mapping

Space mapping, first introduced by Bandler [14], links the high– and low–fidelity models through their input parameters. The goal of space mapping is to vary the input parameters to the low–fidelity model in order to match the output of the high–fidelity model. In microwave circuit design, where space mapping was first developed, it is often appropriate to make corrections to the input of a model, rather than to its output.

The first space–mapping–based optimization algorithm used a linear mapping between the high– and low–fidelity design spaces. It used a least–squares solution of the linear equations resulting from associating corresponding data points in the two spaces. Space mapping optimization consists of optimizing in the low–fidelity space and inverting the mapping to find a trial point in the high–fidelity space. New data points near the trial point are then used to construct the mapping for the next iteration. This process is repeated until no further progress is made. While this method can result in substantial improvement, as demonstrated by several design problems, most in circuit design [12], but some in other disciplines [13], it is not provably convergent to even a local minimum of the high–fidelity space. In fact, while improvement in the high–fidelity model is often possible when the low–fidelity model is accurate, it is not guaranteed.

Space mapping was further improved with the introduction of aggressive space mapping [15]. Aggressive space mapping descends more quickly towards the optimum than space mapping, but requires the assumptions that the mapping between the spaces is bijective and that it is always possible to find a set of low–fidelity design vectors that, when fed into the low–fidelity model, provide an output almost identical to the high–fidelity model evaluated at any given high–fidelity design vector. It also requires that the design variables are the same dimension in both spaces.

The space–mapping examples available in the literature consider only the case where the design vectors have the same length. Therefore, this work expands it to include the variable–parameterization case, including when the design vectors are not the same length.

3.1.1 Standard space–mapping methods

In space mapping, a particular form is assumed for the relationship P between the high– and low–fidelity design vectors. This form is described by some set of space–mapping parameters, contained here in a vector $\mathbf{p}$, that are found by solving an optimization problem

$$\mathbf{p} = \arg\min_{\mathbf{p}} \sum_{i=1}^{q} \left(||\beta(\mathbf{x}^i) - \hat{\beta}\left(P(\mathbf{x}^i, \mathbf{p})\right)||_2 \right). \tag{3.1}$$

This optimization problem seeks to minimize the difference between some high–fidelity function $\beta(\mathbf{x})$ and the corresponding low–fidelity function $\hat{\beta}\left(\hat{\mathbf{x}}\right) = \hat{\beta}\left(P(\mathbf{x}, \mathbf{p})\right)$ over a set of q sample points $\mathbf{x}^i$, where $\mathbf{x}^i$ denotes the i^{th} sample point. Both the choice of sample points and the particular form of the mapping P are left to the implementation. Because the method does not ensure first–order accuracy, the proofs of convergence of trust–region methods do not extend to those methods using space mapping. However, Madsen and Søndergaard have developed a provably convergent algorithm by using a hybrid method in which the surrogate is a convex combination of the space–mapped low–fidelity function and a Taylor series approximation to the high–fidelity function [51].

In the implementation employed in this work, the sample points used in Equation (3.1) are the previous q accepted steps in the TRMM algorithm, $\mathbf{x}^{k-q+1} \ldots \mathbf{x}^k$, at which high–fidelity function values are already available. A linear relationship is chosen for the mapping P:

$$\hat{\mathbf{x}} = P(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{b}, \tag{3.2}$$

where $\mathbf{M}$ is a matrix with $n \times \hat{n}$ elements and $\mathbf{b}$ is a vector of length $\hat{n}$ for a total of $\hat{n} \times (n+1)$ space–mapping parameters. It should be noted that other forms of the mapping could also be used. The space mapping parameters must be determined at each iteration of the TRMM method by solving the optimization problem (3.1).

3.1.2 Corrected space mapping

Because space mapping does not provide provable convergence within a TRMM framework, but any surrogate that is first–order accurate does, one approach is to correct the space–mapping framework to at least first order. This can be done with the corrections in Section 2.1.3. However, if the input parameters are first selected in order to match the output function at some number of control points, and a correction (either additive or multiplicative) is subsequently applied, it is likely that the correction will unnecessarily distort the match performed in the space–mapping step. This can be resolved by performing the space mapping and correction steps simultaneously, which is achieved by embedding the correction within the space mapping.

This concept is illustrated in Figure 3-1. In this figure, the available data points are marked with black circles, and the center of the trust region with a red 'x'. The dotted magenta curve is a cubic function found with a least–squares fit to the available data. It provides no consistency at the trust region center. The dashed cyan curve shows the result of adding a linear additive correction to that fit, in order to enforce first–order accuracy at the center of the trust region. The local correction distorts the global data fitting. The solid blue curve is also a cubic function, generated by first enforcing first–order accuracy at the center, and then performing a least–squares fit with the remaining degrees of freedom. This last curve is more globally accurate than the sequential fitting and correction steps.

Using this concept, corrected space–mapping performs the space mapping and

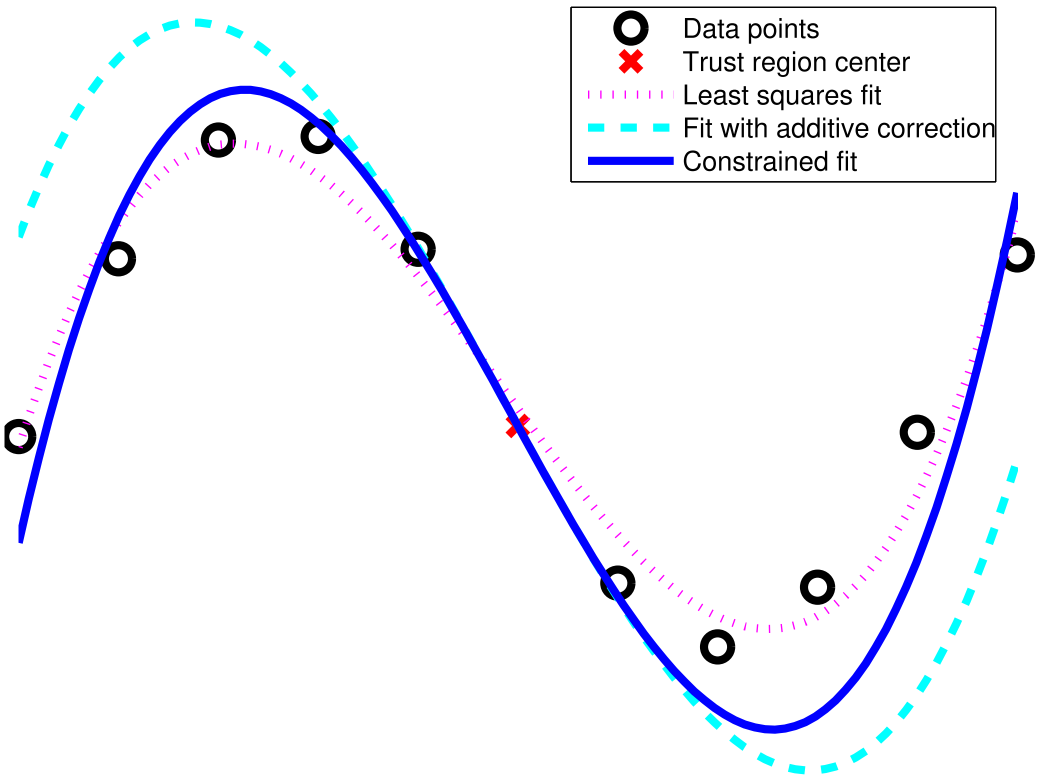

Figure 3-1: Demonstration of simultaneous vs. sequential data fitting and enforcement of first–order accuracy

correction steps simultaneously. That is, it incorporates a correction, and with the remaining degrees of freedom, performs the best match possible to the control points by varying the input mapping.

The corrected space mapping optimization problem is

$$\mathbf{p}^k = \arg\min_{\mathbf{p}} \sum_{i=1}^{q} ||\beta(\mathbf{x}^i) - \tilde{\beta}^k \left(P(\mathbf{x}^i, \mathbf{p}) \right) ||_2. \tag{3.3}$$

Equation (3.3) is the same as Equation (3.1), with $\hat{\beta}$, the uncorrected low–fidelity function, replaced by $\tilde{\beta}^k$, the corrected surrogate to the high–fidelity function on the k^{th} subproblem. The optimization problem (3.3) seeks to minimize the difference between the high–fidelity and surrogate objective functions over a set of k sample points $\mathbf{x}^i$, where $\mathbf{x}^i$ denotes the i^{th} sample (or control) point. Both the choice of sample points and the particular form of the mapping P is left to the implementation.

It should be noted that when using corrected space mapping, the correction function must be re–evaluated with each new value of $\mathbf{p}$, since the low–fidelity function values, the low–fidelity gradient, and the first– and second–order derivatives of the mapping needed in Equations (2.29)–(2.32) change with the space–mapping parameters. Since the resulting corrected function is at least first–order accurate at the center of the trust region, the resulting trust–region optimization is provably convergent to a local minimum of the high–fidelity problem.

3.1.3 Solving the subproblem in the low–fidelity space

Since space mapping is a dynamic mapping, changing from point to point over the course of the optimization, it is an appropriate selection for working in the low–fidelity space. It works as follows. As described in Section 2.3.2, one of the requirements of the mapping for working in the low–fidelity space is

$$\|\mathbf{J}^T \nabla \beta(Q(\hat{\mathbf{x}^k}))\| \geq \alpha \|\nabla \beta(Q(\hat{\mathbf{x}^k}))\|, \tag{3.4}$$

where Q is the map from the low– to the high–fidelity space, $\mathbf{J}$ is the Jacobian of the mapping, $\hat{\mathbf{x}}^k$ is the center of the trust–region (defined in the low–fidelity space) on the k^{th} trust–region iteration, and α is a small positive number.

Equation (3.4) specifies that the gradient of the high–fidelity function is not orthogonal to the mapping. It is expected that larger values of α will cause the optimization method to converge more quickly; that is, the closer the high–fidelity gradient is to being orthogonal to the mapped space, the slower the convergence of the method. This optimization algorithm — like all methods discussed in this work — is designed for general non–linear programming, where the gradient is expected to change from point to point during optimization. Therefore, in order to satisfy Equation (3.4), the mapping is required to be dynamic, that is $Q(x^k)$ must change as x^k changes.

The corrected space mapping optimization problem to ensure condition (3.4) is given by

$$\begin{array}{rl} \min\limits_{\mathbf{p}} & \sum_{i=1}^{k} ||\beta(Q(\hat{\mathbf{x}}^i,\mathbf{p})) - \tilde{\beta}^k(\hat{\mathbf{x}}^i,\mathbf{p})||_2, \\ \text{Subject to} & \|\mathbf{J}^T\nabla\beta(Q(\hat{\mathbf{x}}^k))\| \geq \alpha\|\nabla\beta(Q(\hat{\mathbf{x}}^k),\mathbf{p}))\|. \end{array} \tag{3.5}$$

The corrected surrogate, $\tilde{\beta}$, must satisfy the consistency conditions 2.5 and 2.6, and can for example be computed using an additive correction

$$\tilde{\beta}^k(\hat{\mathbf{x}}) = \hat{\beta}\left(P(\hat{\mathbf{x}})\right) + A^k(\mathbf{x}), \tag{3.6}$$

where the correction function, A^k, is a quadratic Taylor series centered at $\mathbf{x}^k$ of the difference between the two functions $\beta(Q(\hat{\mathbf{x}}))$ and $\hat{\beta}(\hat{\mathbf{x}})$.

3.2 POD mapping

The third mapping methodology is based on the gappy POD method of reconstructing data sets. This, in turn, is based on the POD method [45, 65], also known as principal components analysis and Karhunen–Loève expansions, which yields a set of basis vectors that provides the least–squares optimal representation of a given data set.

3.2.1 Proper Orthogonal Decomposition

The POD method of snapshots, developed by Sirovich [65], finds the basis vectors empirically. In this method, a set of q snapshots $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^q$, or column vectors describing different states of a system, is computed. The POD basis vectors, $\phi^j,\ j = 1, 2, \ldots, q$, can then be computed as the left singular vectors of the matrix $\mathbf{X}$, defined as

$$\mathbf{X} = \Big[[\mathbf{x}^1 - \bar{\mathbf{x}}] \quad [\mathbf{x}^2 - \bar{\mathbf{x}}] \quad \cdots \quad [\mathbf{x}^q - \bar{\mathbf{x}}] \Big], \tag{3.7}$$

where $\bar{\mathbf{x}}$ is the mean of the snapshots and the i^{th} column of $\mathbf{X}$ contains the i^{th} snapshot minus the mean.

The singular values of $\mathbf{X}$ indicate the relative importance of the corresponding basis vector in representing the snapshots. Therefore, only the d basis vectors corresponding to the largest singular values are used. A low–dimensional representation of a solution $\mathbf{x}$ is thus given by

$$\mathbf{x} \approx \bar{\mathbf{x}} + \sum_{i=1}^{d} \gamma_i \phi^i, \tag{3.8}$$

where γ_i is the coefficient describing the contribution of the i^{th} POD mode ϕ^i to the solution $\mathbf{x}$.

Omitting the remaining $d - q$ vectors from the basis leads to an error, given by the sum of the squares of the singular values corresponding to those neglected modes, in the reduced–basis representation of the original snapshot set.

3.2.2 Gappy POD

The gappy POD method allows one to reconstruct data from a "gappy" data set, that is, a set in which some of the data are unknown or missing [30, 20]. The first step is to define a mask vector, which describes for a particular solution vector where data are available and where data are missing. For the solution $\mathbf{x}$, the corresponding mask vector $\mathbf{n}$ is defined as follows:

$$n_i = \begin{cases} 0 & \text{if } x_i \text{ is missing,} \\ 1 & \text{if } x_i \text{ is known.} \end{cases} \tag{3.9}$$

Point–wise multiplication is defined as $(\mathbf{n}, \mathbf{x})_i = n_i x_i$. Then the gappy inner product is defined as $(\mathbf{u}, \mathbf{v})_n = ((\mathbf{n}, \mathbf{u}), (\mathbf{n}, \mathbf{v}))$, and the induced norm is $(\|\mathbf{v}\|_n)^2 = (\mathbf{v}, \mathbf{v})_n$.

Let there also exist a set of snapshots in which all the components are known. Perform the POD basis generation on that set to obtain a set of basis vectors ϕ^i, $i = 1 \dots q$. For a vector $\mathbf{x}$ that has some unknown components, it is assumed that the repaired vector $\check{\mathbf{x}}$ can be represented by the expansion

$$\check{\mathbf{x}} \approx \sum_{i=1}^{d} \gamma_i \phi^i. \tag{3.10}$$

The POD coefficients γ_i are chosen to minimize the error between the available and reconstructed data. This error can be defined as

$$e = \|\mathbf{x} - \check{\mathbf{x}}\|_n^2, \tag{3.11}$$

using the gappy norm so that only the known data elements in $\mathbf{x}$ are compared. The coefficients γ_i that minimize the error e can be found by differentiating (3.11) with respect to each of the γ_i in turn. This leads to the linear system of equations

$$\mathbf{E}\gamma = \mathbf{g}, \tag{3.12}$$

where the ij^{th} component of $\mathbf{E}$ is given by

$$E_{ij} = \left(\phi^i, \phi^j\right)_n \tag{3.13}$$

and the i^{th} component of $\mathbf{g}$ is given by

$$g_i = \left([\mathbf{x} - \bar{\mathbf{x}}], \phi^i\right)_n. \tag{3.14}$$

Solving Equation (3.12) for γ, and then using (3.10) gives the repaired vector $\check{\mathbf{x}}$. Finally, the missing elements of $\mathbf{x}$ are replaced with the corresponding elements of the repaired vector $\check{\mathbf{x}}$. That is,

$$x_i^{repaired} = \begin{cases} \check{x}_i & \text{if } n_i = 0, \\ x_i & \text{if } n_i = 1. \end{cases} \tag{3.15}$$

3.2.3 POD mapping

The gappy POD method provides a way to map between high– and low–fidelity design space data: the high–fidelity vector is treated as the known data, and the low–fidelity as the unknown data, or vice versa. In the mapping application, the POD basis vectors must span both low– and high–fidelity design space. This is achieved by generating a set of q training pairs, for which the low– and the high–fidelity vectors describe the same physical system. These training pairs are combined in the following way to form the snapshot matrix:

$$\mathbf{X} = \begin{bmatrix} [\hat{\mathbf{x}}^1 - \bar{\hat{\mathbf{x}}}] & [\hat{\mathbf{x}}^2 - \bar{\hat{\mathbf{x}}}] & \cdots & [\hat{\mathbf{x}}^q - \bar{\hat{\mathbf{x}}}] \\ - & - & & - \\ [\mathbf{x}^1 - \bar{\mathbf{x}}] & [\mathbf{x}^2 - \bar{\mathbf{x}}] & \cdots & [\mathbf{x}^q - \bar{\mathbf{x}}] \end{bmatrix}, \tag{3.16}$$

where now the i^{th} column of $\mathbf{X}$ contains both the i^{th} low– and the i^{th} high–fidelity snapshots, and $\bar{\hat{\mathbf{x}}}$ denotes the mean of the low–fidelity snapshot set.

The left singular vectors of this snapshot matrix provide the corresponding POD basis vectors, which are partitioned in the same way as the snapshot vectors. That is,

$$\mathbf{\Phi} = \begin{bmatrix} \hat{\phi}^1 & \hat{\phi}^2 & \cdots & \hat{\phi}^q \\ - & - & & - \\ \phi^1 & \phi^2 & \cdots & \phi^q \end{bmatrix}, \tag{3.17}$$

where ϕ^i is the portion of the i^{th} POD basis vector corresponding to $\mathbf{x}$ and $\hat{\phi}^i$ is the portion corresponding to $\hat{\mathbf{x}}$. Equation (3.8) can then be decomposed into two equations,

$$\hat{\mathbf{x}} = \bar{\hat{\mathbf{x}}} + \sum_{i=1}^{d} \gamma_i \hat{\phi}^i, \tag{3.18}$$

$$\mathbf{x} = \bar{\mathbf{x}} + \sum_{i=1}^{d} \gamma_i \phi^i. \tag{3.19}$$

Using the gappy POD formulation, Equation (3.19) can be solved in a least squares sense in order to find the coefficients γ that best represent a given high–fidelity vector $\mathbf{x}$. Those coefficients can then be used in Equation (3.18) to calculate the low–fidelity vector. Alternatively, if a mapping is desired from the low–fidelity space to the high–fidelity space, the coefficients are found from Equation (3.18) and used in (3.19).

Unlike for space mapping, a single POD mapping is used for the objective and all constraints in all constraint–management methods. When incorporating this method into the TRMM framework, an additive or multiplicative correction must be used to ensure at least first–order consistency. These corrections are applied as shown in Section 2.1.3.

POD mapping is static in that the mapping stays the same over the course of the optimization. Because the mapping is not recalculated on each step, this leads to less computational overhead. As described in Section 3.1.3, space mapping and corrected space mapping are dynamic. Therefore, when the sub–problem is solved in the low–fidelity space, corrected space mapping is chosen.

3.3 Hybrid POD/Space Mapping Method

Performing space mapping or corrected space mapping in the high–fidelity space requires the solution of the $O(\hat{n}n)$ optimization problem (3.3) on each trust–region iteration. If n, $\hat{n}$, or both are large, this can contribute to significant overhead on each optimization step. POD mapping, on the other hand, has significantly less overhead. However, POD mapping is static, and can therefore not be used for optimization in the low–fidelity space. A method was therefore developed to combine the advantages of corrected space mapping with the lower overhead of POD mapping. This method is a hybrid between POD mapping and corrected space mapping.

Hybrid POD/space mapping can be interpreted as space mapping in the POD space. The POD basis vectors are constructed, but instead of using the POD coefficients calculated from Equation (3.18) in Equation (3.19), separate coefficients are linked through space mapping. That is, Equation (3.19) remains the same,

$$\mathbf{x} = \bar{\mathbf{x}} + \sum_{i=1}^{d} \gamma_i \phi^i, \tag{3.20}$$

but Equation (3.18) becomes

$$\hat{\mathbf{x}} = \bar{\hat{\mathbf{x}}} + \sum_{i=1}^{d} \hat{\gamma}_i \hat{\phi}^i. \tag{3.21}$$

The new high–fidelity POD coefficients γ are calculated by solving Equation (3.20) in a least–squares sense. Then, the low–fidelity coefficients $\hat{\gamma}$ are found using space–mapping.

The matrix $\mathbf{E}$ and vector $\mathbf{g}$ are defined as in the POD mapping method; that is

$$E_{ij} = \left(\phi^i, \phi^j\right)_n, \tag{3.22}$$

$$g_i = \left([\mathbf{x} - \bar{\mathbf{x}}], \phi^i\right)_n. \tag{3.23}$$

The high–fidelity POD coefficients γ can then be calculated using:

$$\gamma = \mathbf{E}^{-1}\mathbf{g}. \tag{3.24}$$

Then, the low–fidelity POD coefficients are calculated using space mapping

$$\hat{\gamma} = \mathbf{M}\gamma + \mathbf{b} \tag{3.25}$$

and the low–fidelity design parameters are reconstructed using the POD–mapping rules with the new low–fidelity POD coefficients

$$\hat{\mathbf{x}} = \bar{\hat{\mathbf{x}}} + \hat{\mathbf{\Phi}}\hat{\gamma}. \tag{3.26}$$

The mapping $\hat{\mathbf{x}} = P(\mathbf{x}, \mathbf{M}, \mathbf{b})$ is now defined, except for the definition of the space mapping parameters $\mathbf{M}$ and $\mathbf{b}$. The corrected surrogate can now be defined, using the mapping, as

$$\tilde{\beta}(\mathbf{x}, \mathbf{M}.\mathbf{b}) = \hat{\beta}(P(\mathbf{x}, \mathbf{M}, \mathbf{b})) + A(\mathbf{x}, \mathbf{M}, \mathbf{b}), \tag{3.27}$$

where A is the standard additive correction function of some order, defined such that $\tilde{\beta}(\mathbf{x}, \mathbf{M}, \mathbf{b})$ is at least first–order consistent with $\beta(\mathbf{x})$ at the center of the trust region. The space–mapping parameters $\mathbf{M}, \mathbf{b}$ in Equation (3.25) are chosen by solving the optimization problem

$$\mathbf{M}, \mathbf{b} = \arg\min_{\mathbf{M},\mathbf{b}} \sum_{i=1}^{k} ||\beta(\mathbf{x}^i) - \tilde{\beta}(\mathbf{x}^i, \mathbf{M}, \mathbf{b})||_2. \tag{3.28}$$

Each variable–complexity correction scheme described in 2.1.3 requires the Jacobian, or first derivative, of the mapping. The Jacobian of this mapping is

$$\mathbf{J} = \hat{\mathbf{\Phi}}\mathbf{M}\mathbf{E}^{-1}\mathbf{\Phi}^T. \tag{3.29}$$

In the case where the dimension of $\mathbf{x}$ or $\hat{\mathbf{x}}$ is very large, space mapping involves calculating the elements of $\mathbf{M}$ and $\mathbf{b}$ by solving an optimization problem in $O(n\hat{n})$ dimensions, which can be prohibitive. The hybrid space mapping optimization problem has $O(r^2)$ dimensions. The reduced space mapping problem size corresponds to reduced flexibility in the mapping. This in turn may correspond to more trust–region sub–problems, each with a high–fidelity function call. Therefore, on large problems, the computational savings realized from reducing the dimension of the space–mapping problem may outweigh the cost of the increased number of high–fidelity function calls resulting from there being more trust–region sub–problems. In addition, space mapping requires enough data points to uniquely determine the elements of $\mathbf{M}$ and $\mathbf{b}$, which can be a very large number in the case of large problems. Hybrid POD/space mapping reduces the number of data points required. Hybrid POD mapping therefore represents another option in the designer's toolbox: appropriate for use when the problem is very large and where the high–fidelity function calls are inexpensive relative to the time it takes to solve the space–mapping optimization problem.

3.4 Application of Mapping to TRMM

Each of the TRMM algorithms in Chapter 2 requires the construction of a surrogate $\tilde{\beta}^k(\mathbf{x})$ to some high fidelity function $\beta(\mathbf{x})$ with specified properties. This surrogate requires a low–fidelity model $\hat{\beta}(\hat{\mathbf{x}})$, a mapping $\hat{\mathbf{x}} = P(\mathbf{x})$, and a correction function. The surrogate construction and evaluation below show additive corrections. Modifications for multiplicative and combination corrections are straightforward.

3.4.1 Construction of the surrogate

This process is performed once per trust–region iteration and results in a surrogate $\tilde{\beta}^k(\mathbf{x})$ that is zeroth– and first–order consistent with the high fidelity function $\beta(\mathbf{x})$.

1. Calculate the high–fidelity function value $\beta(\mathbf{x}^k)$ and gradient $\nabla_{\mathbf{x}}\beta(\mathbf{x}^k)$ at the center $\mathbf{x}^k$ of the k^{th} trust region. The value $\beta(\mathbf{x}^k)$ is already available from evaluating the center of the trust region as a trial point in the calculation of the trust–region ratio, in Equation 2.3 or 2.40.

2. In the case of a dynamic mapping, such as corrected space mapping, calculate the mapping $P(\mathbf{x})$. In the case of a static mapping, such as POD mapping, use the existing pre–calculated mapping.

3. Calculate the value $\hat{\beta}(P(\mathbf{x}^k))$ and the gradient $\nabla_{\hat{\mathbf{x}}}\hat{\beta}(P(\mathbf{x}^k))$ of the low–fidelity function at the mapped center of the trust region.

4. If using a quasi–second–order correction function, update the estimate of the Hessian matrix of both the high– and low–fidelity functions, using the equations in Section 2.1.2.

5. Calculate the correction function $A^k(\mathbf{x})$ using the equations of Section 2.1.3. $A^k(\mathbf{x})$ is a Taylor series of at least first order of $\mathcal{A}(\mathbf{x}) = \beta(\mathbf{x}) - \hat{\beta}(P(\mathbf{x}))$ around the center $\mathbf{x}^k$ of the trust region.

3.4.2 Evaluation of the surrogate

Recall that the surrogate is minimized using a standard constrained minimization algorithm. These algorithms evaluate the surrogate and its gradient at a number of points. Below is the series of steps required to evaluate the surrogate $\tilde{\beta}^k(\cdot)$ at a particular point $\mathbf{x}$.

1. Calculate the point $\hat{\mathbf{x}}$ in the low fidelity space by applying the mapping $\hat{\mathbf{x}} = P(\mathbf{x})$
2. Evaluate the low–fidelity function at that point, to obtain $\hat{\beta}(\hat{\mathbf{x}})$
3. Apply the correction. For example, in the additive case, the value of the surrogate is $\tilde{\beta}^k(\mathbf{x}) = \hat{\beta}(P(\mathbf{x})) + A^k(\mathbf{x})$ where $A^k(\mathbf{x})$ is the additive correction function described in Section 2.1.3.

This provides the surrogate $\tilde{\beta}^k(\mathbf{x})$, which, since the correction function is at least a first–order Taylor series of the difference between the mapped low–fidelity function and the high–fidelity function, is at least first–order consistent with the high–fidelity function at the center of the trust region. Therefore, as long as the high–fidelity function meets the other mild requirements for convergence, and the inner optimization algorithm generates a trial step that satisfies the FCD condition, the algorithms in Chapter 2 are provably convergent to a local minimum of the high–fidelity problem.

3.5 Comparison of Mapping Methodology

Before presenting results, some observations about the differences between the mapping methods above can be made.

Space mapping is dynamic, changing from point to point over the course of the optimization. It requires no initial setup work. However, it requires a solution of a least–squares problem on every trust–region iteration. This problem is over a parameter space of size $O(\hat{n}n)$. Depending on the number of design variables in the high– and low–fidelity models, this can be a very large number. It can be concluded, therefore, that the space–mapping and corrected–space–mapping procedures have a high overhead, and this overhead increases with the square of the size of the underlying problem. However, in the case of high–fidelity analyses of large engineering systems of interest, a single high–fidelity function evaluation is normally much more costly than the overhead required by the optimization algorithm or the mapping method. The metric of interest is therefore the number of high–fidelity function evaluations required to find the optimum.

POD mapping is static, remaining constant during an optimization. Therefore, only simple algebraic manipulation is required on each trust–region iteration, resulting in low overhead. However, it requires significantly more setup work before solving the optimization problem. Training pairs must be generated and POD basis functions computed. An in–depth knowledge of the problem and significant human intervention may be required to generate the training pairs.

The hybrid mapping reduces the overhead of space mapping on large problems, while allowing a dynamic mapping. The tradeoff, however, is that it, like POD mapping, requires training pairs and some upfront algebraic work.

Table 3.1 presents these advantages and disadvantages of each method compactly.

Mapping	**Provable Convergence**	**Initial Work**	**Overhead**
Space mapping	No	Low	High
Corrected space mapping	Yes	Low	High
POD mapping	Yes	High	Low
Hybrid mapping	Yes	High	Medium

Table 3.1: Advantages and disadvantages of mapping methods

Further chapters will apply these mapping methods, in combination with TRMM optimization methods, to both analytic models and engineering models, for both constrained and unconstrained problems. Chapter 4 presents the unconstrained problems, Chapter 5 presents a constrained analytic problem, and Chapter 6 presents two constrained design problems.

Chapter 4

Unconstrained Problems

The methods were first tested on three unconstrained problems: two analytic problems using the Rosenbrock function and an unconstrained airfoil design problem.

4.1 Two–Dimensional Rosenbrock Problem

4.1.1 Problem Description

For the first example, the dimension of the problem is the same in both the high–fidelity and low–fidelity models. The high–fidelity problem is the minimization of the Rosenbrock function

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 1)^2 + 100(x_2 - x_1^2)^2. \tag{4.1}$$

Figure 4-1 show the contours of the Rosenbrock function. The optimal solution of the problem is $x_1 = 1$, $x_2 = 1$, with an optimal objective function value of zero. The Rosenbrock problem is a well–known test function for numerical optimization. It was first used by Rosenbrock in 1960 [61] and is widely used as a benchmark for both evolutionary algorithms and gradient–based optimization algorithms [72, 47, 39].

The minimum is in a long, narrow valley, and methods based on unscaled steepest descent tend to work poorly. Due to its shape, the Rosenbrock function often serves as a test case for premature convergence [47, 39, 35]. While this problem is not

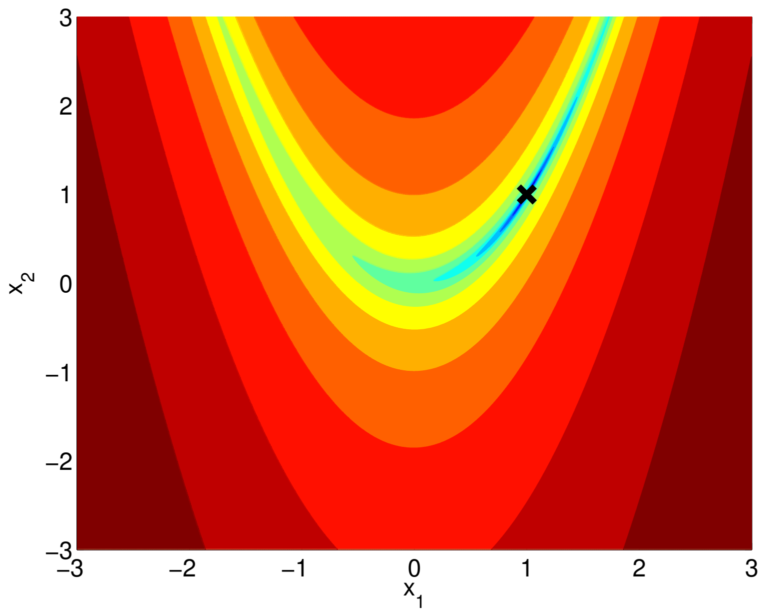

Figure 4-1: Contours of the Rosenbrock function. The minimum, at $\mathbf{x} = (1, 1)$, is marked with an 'x'.

representative of real–world design problems, it serves as a good first test problem to evaluate the method for the expected superlinear convergence.

For this simple test, a quadratic function was used as the low–fidelity model:

$$\hat{f}(\hat{\mathbf{x}}) = \hat{x}_1^2 + \hat{x}_2^2. \tag{4.2}$$

Figure 4-2 shows contours of this function.

4.1.2 Results

The benchmark method is the unconstrained medium–scale unconstrained optimization method implemented by MATLAB©. This is a quasi–Newton method, using the BFGS formula for updating the approximation to the Hessian matrix. The line search algorithm is a safeguarded mixed quadratic and cubic polynomial interpolation and extrapolation method.

The initial point is $\mathbf{x} = (-2, -2)$. The multifidelity method was then run, using both corrected space mapping and POD mapping. Corrected space mapping used

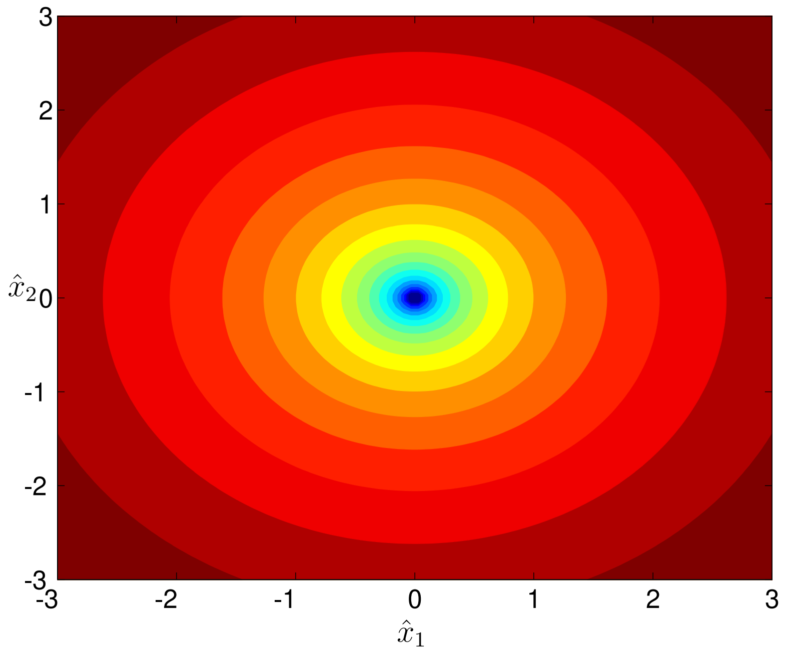

Figure 4-2: Contours of a simple quadratic function

eight sample points. POD mapping used 100 snapshots distributed in a grid from -3 to 3 in both design variables. Two POD basis vectors were used.

Both the low–fidelity and high–fidelity models used analytic gradients. When counting the number of high–fidelity function calls, one call includes the calculation of both the function and the gradient.

Since, in this case, the sum of the quadratic low–fidelity function and a quasi–second–order additive correction still provides a quadratic model, the expected convergence rate of the multifidelity TRMM framework is the same as that of the direct high–fidelity optimization, which uses approximations based on a quasi–second–order Taylor series. The multifidelity methods are therefore not expected to provide significant computational savings for this case, nor is a mapping from high– to low–fidelity space required in this simple case where $\mathbf{x} = \tilde{\mathbf{x}}$; however, super–linear convergence on this problem verifies that the combination of the multifidelity method and the mapping behaves as expected.

Figure 4-3 shows the paths each of the methods took in the design–variable space. The benchmark quasi–Newton method and the multifidelity method with POD map-

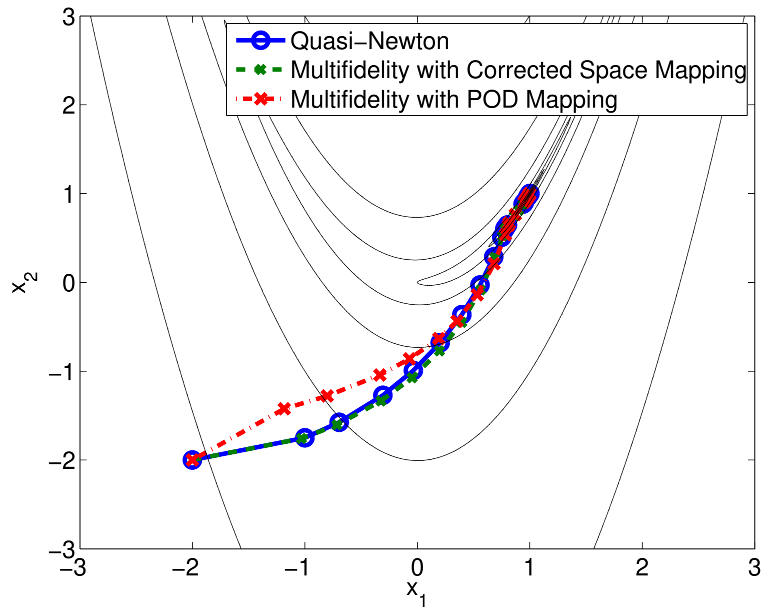

Figure 4-3: Path of the benchmark method, the multifidelity method with corrected space mapping, and the multifidelity method with POD mapping, on the two–dimensional Rosenbrock problem.

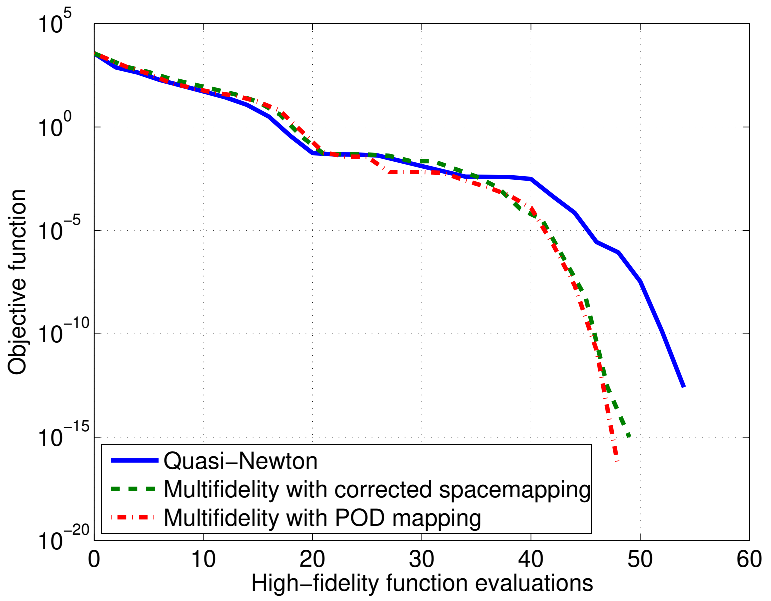

Figure 4-4: Objective function of the benchmark quasi–Newton method, the multifidelity method with corrected space mapping, and the multifidelity method with POD mapping, on the two–dimensional Rosenbrock problem.

ping took very similar paths, while the multifidelity method with corrected space mapping took a slightly different path initially before rejoining the path of the other two methods. Figure 4-4 shows the objective function versus the number of high–fidelity function evaluations. All three methods converge super–linearly. The multifidelity method, with either choice of mapping, provides modest computational gains over the benchmark. For example, it takes the quasi–Newton method 52 high–fidelity function calls to get to an objective function below 10^{-10}, while it takes the multifidelity method, with either mapping, 46 high–fidelity function calls. This is a computational savings of 11.5%. While it is promising that the method provides computational savings even on a simple problem, it is more reassuring that the method converges super–linearly, as expected.

4.2 Rosenbrock Problem: Variable parameterization

The second example is a simple analytic problem for which the high– and low–fidelity design vectors have different dimension. The high–fidelity problem is the minimization of a six–dimensional extended Rosenbrock function

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 1)^2 + 100 \sum_{i=2}^{6} (x_i - x_{i-1}^2)^2. \tag{4.3}$$

This problem has a global minimum at $x_i = 1$, $i = 1 \ldots 6$ [63]. The low–fidelity model is a two–dimensional Rosenbrock function

$$\hat{f}(\hat{\mathbf{x}}) = (\hat{x}_1 - 1)^2 + 100(\hat{x}_2 - \hat{x}_1^2)^2. \tag{4.4}$$

For all presented results, an initial guess of $x_i = -2,\ i = 1, 2, \ldots, 6$ was used.

This problem was chosen to confirm that the methods are convergent when the low–fidelity and high–fidelity models are of different dimension. As in the previous section, the benchmark was MATLAB's implementation of a quasi–Newton method with mixed quadratic and cubic line searches. The multifidelity method with corrected space mapping and POD mapping were compared to this benchmark. For corrected space mapping, eight sample points were used. For POD mapping, 100 snapshots were generated. The low–fidelity portion of the snapshots were generated using a grid extending from -3 to 3 in each variable. The high–fidelity portion of each snapshot was created by setting $x_1 = \hat{x}_1$, $x_2 = \hat{x}_2$, $x_3 = \hat{x}_1$, $x_4 = \hat{x}_2$, $x_5 = \hat{x}_1$, and $x_6 = \hat{x}_2$. Two POD basis functions were used.

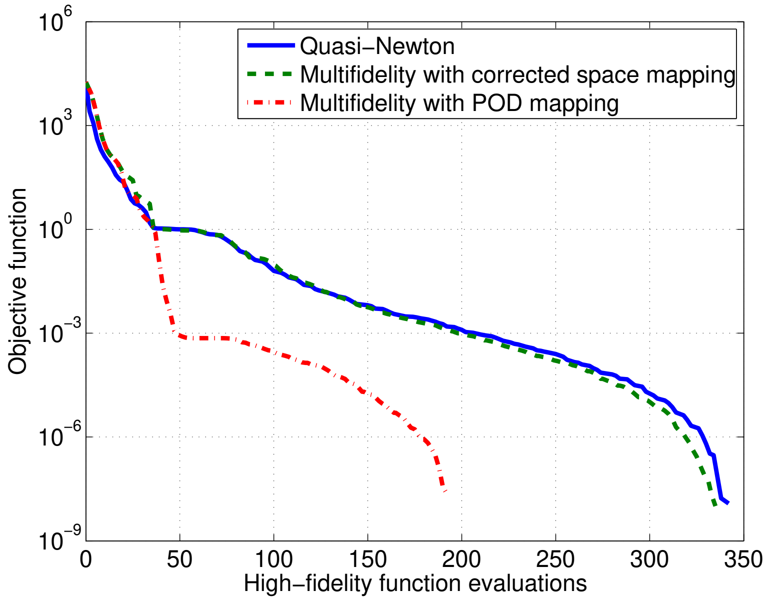

Figure 4-5: Objective function of the benchmark quasi–Newton method, the multifidelity method with corrected space mapping, and the multifidelity method with POD mapping, on the six–dimensional Rosenbrock problem.

Figure 4-5 shows the objective function of each of the methods as a function of the number of high–fidelity function evaluations. All three methods converge super–linearly. The multifidelity method with corrected space mapping converged at nearly an identical rate to the benchmark method, providing small computational gains only late in the optimization. The multifidelity method with POD mapping performs exceptionally well in the first 50 steps, before entering the super–linear phase. It maintains its lead, and, using a convergence tolerance of 10^{-8}, achieves approximately 50% savings in high–fidelity function evaluations. These results show that the method is super–linearly convergent, even in the case where the high–fidelity problem and the low–fidelity problem are of different dimension. Furthermore, it shows that the multifidelity method has the potential for computational savings, particularly using the POD mapping method.

4.3 Airfoil Design Problem

The next problem is an airfoil design problem. It was chosen to represent a realistic engineering design problem with a low–fidelity model that neglects physics contained in the high–fidelity model. The two models have different numbers of design variables due to differences in the geometric parameterization of the problem. The difference in the number of design variables is significant: the low–fidelity model has two, and the high–fidelity model has thirty–six.

The goal of this problem is to design an airfoil that matches a target pressure distribution. The pressure distribution of the NACA 2412 airfoil was chosen as the target. The objective function is the difference between the pressure distribution on an airfoil and the target pressure distribution

$$f = \int_0^1 \left(C_P - C_{P_{target}}\right)^2 ds, \tag{4.5}$$

where C_P is the coefficient of pressure and $C_{P_{target}}$ is the coefficient of pressure of the goal airfoil. The integral is over the unit chord and is approximated using trapezoidal integration. The gradients necessary to solve the optimization problem were calculated using finite differences. When tallying high–fidelity function calls, those required to generate finite–difference gradients are also included. The initial and target airfoils, and the corresponding coefficient–of–pressure distributions, are shown in Figure 4-6.

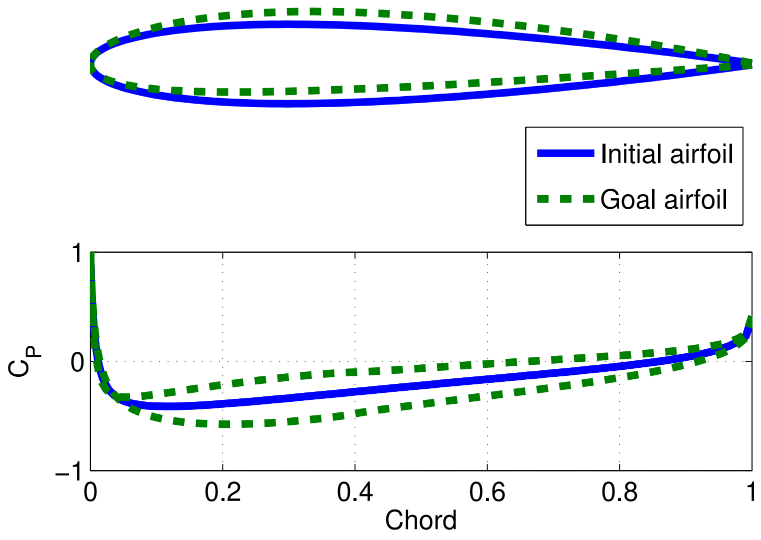

Figure 4-6: Initial and goal airfoils, along with their coefficient of pressure distributions.

4.3.1 High–fidelity analysis

The high–fidelity analysis uses XFOIL [24] in inviscid mode. The inviscid formulation of XFOIL uses a linear–vorticity stream–function panel method. The equations are closed with an explicit Kutta condition. The high–fidelity geometry vector consists of the magnitudes of 36 Hicks–Henne bump parameters [43], 18 of which perturb the thickness of the airfoil and 18 of which perturb the camber, as shown in Figure 4-7. They are evenly distributed across the airfoil at 18 control points.

4.3.2 Low–fidelity analysis

The low–fidelity analysis is an analytic solution to a Joukowski transform [8]. The Joukowski transform is a conformal map that maps the points on a unit circle to points on the surface of an airfoil. Only two variables are needed to describe a Joukowski airfoil: μ_x and μ_y, the x and y coordinates of the center of the circle used in the Joukowski transform.

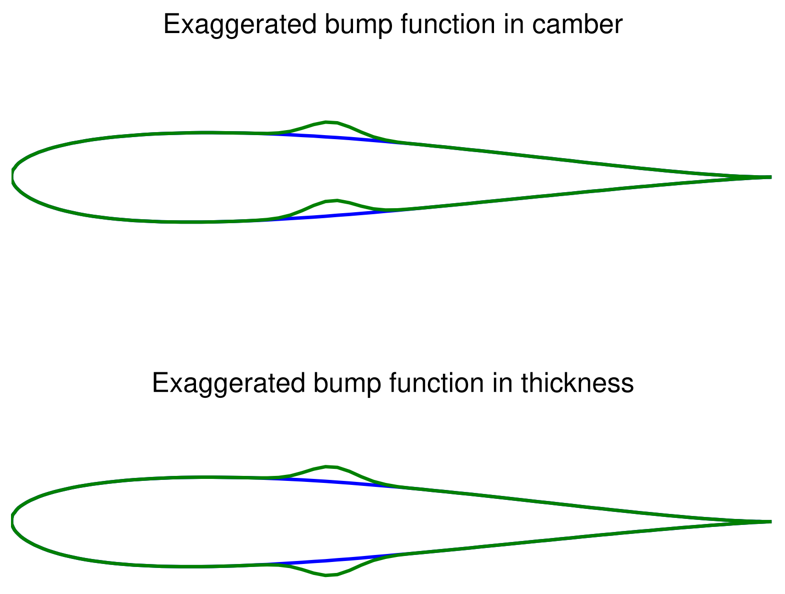
Exaggerated bump function in camber
Exaggerated bump function in thickness

Figure 4-7: For the high–fidelity model, Hicks–Henne bump functions are used to parameterize the airfoil geometry.

The transform is

$$z = \zeta + \frac{1}{\zeta}, \tag{4.6}$$

where $z = x + iy$ is the complex number representing the coordinates of a point on the airfoil in the new space and $\zeta = \chi + i\eta$ is the complex number representing a point on the circle in the original space. Figure 4-8 shows one Joukowski airfoil and the corresponding circle. The circle encloses the origin, where the conformal map has a singularity, and intersects the point $z = 1$.

The transform is used to solve for the two–dimensional potential flow around the Joukowski airfoil. The complex velocity $\tilde{W}$ around the circle in the ζ plane is

$$\tilde{W} = V_\infty e^{i\alpha} + \frac{i\Gamma}{2\pi(\zeta - \mu)} - \frac{V_\infty R^2 e^{i\alpha}}{(\zeta - \mu)^2}, \tag{4.7}$$

where $\mu = \mu_x + i\mu_y$ is the complex coordinate of the center of the circle, V_∞ is the freestream velocity of the fluid, α is the angle of attack of the airfoil with respect to the freestream flow, R is the radius of the circle, calculated using $R = \sqrt{(1-\mu_x)^2 + \mu_y^2}$, and Γ is the circulation, found using the Kutta condition, which reduces in this case to

$$\Gamma = 4\pi V_\infty R \sin\left(\alpha + \sin^{-1}\left(\frac{\mu_y}{R}\right)\right). \tag{4.8}$$

The complex velocity W around the airfoil in the z plane is, according the rules of conformal mapping,

$$\begin{aligned} W &= \frac{\tilde{W}}{\frac{dz}{d\zeta}} && (4.9) \\ &= \frac{\tilde{W}}{1 - \frac{1}{\zeta^2}}. && (4.10) \end{aligned}$$

The coefficient of pressure is then calculated using

$$C_P(\zeta) = 1 - \frac{\|W(\zeta)\|^2}{V_\infty^2} \tag{4.11}$$

at each point on the surface of the airfoil.

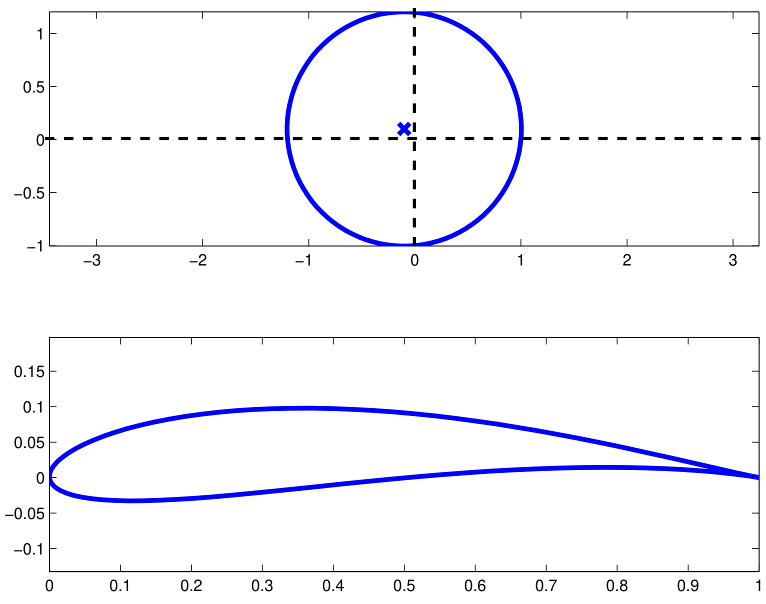

Figure 4-8: The unit circle in the upper plot is transformed to the airfoil in the lower plot using the Joukowski transform.

4.3.3 Generation of Results

100 snapshots were created using a grid in the Joukowski parameter space, varying μ_x from 0.01 to 0.3, and μ_y from 0 to 0.5. The corresponding high–fidelity design vectors were produced by solving an optimization problem to determine the magnitudes of the 36 Hicks–Henne bump functions that best matched the desired Joukowski airfoil. Specifically, the bump functions were chosen so as to minimize the integral over the chord length of the square of the difference between the airfoils defined in each space.

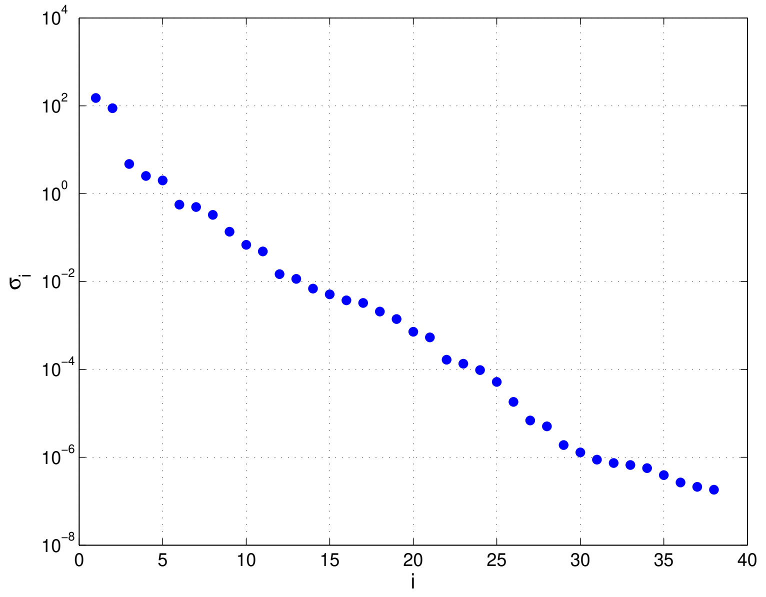

Figure 4-9: Singular values of the snapshot matrix for the airfoil design problem, showing that the first two are much more significant than the others.

Figure 4-9 shows the singular values of the POD snapshot matrix. The relative magnitudes of the singular values indicate the importance of the corresponding POD basis vectors in representing the snapshot data. The first two singular values are more than an order of magnitude larger than the third; therefore, only the first two basis vectors were used in the POD mapping.

Figures 4-10 to 4-12 show the behavior of the objective function for some selected cuts in the high–fidelity design space. These cuts correspond to an iterate near the end of the optimization process, that is, for an airfoil that is close to the goal airfoil. Figures 4-10 and 4-11 show the variation of the objective function with the

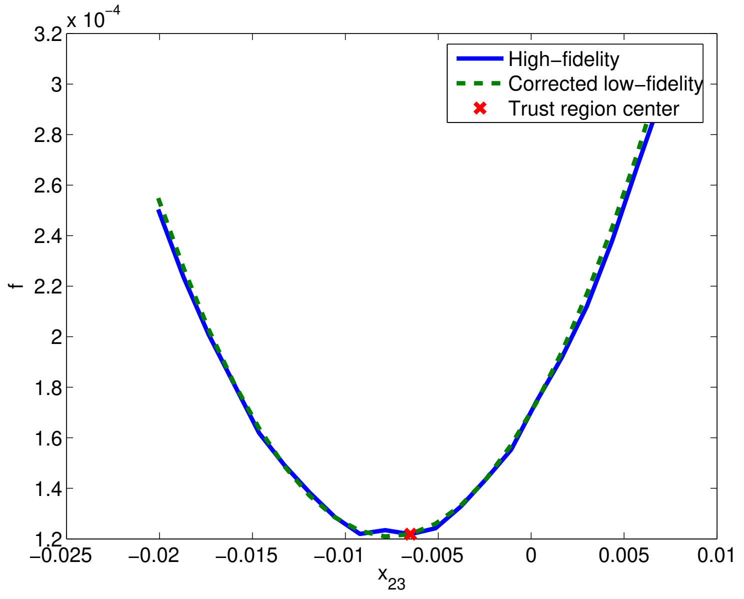

Figure 4-10: The high-fidelity and corrected low-fidelity functions as a function of variable x_{23} for an airfoil close to the goal airfoil.

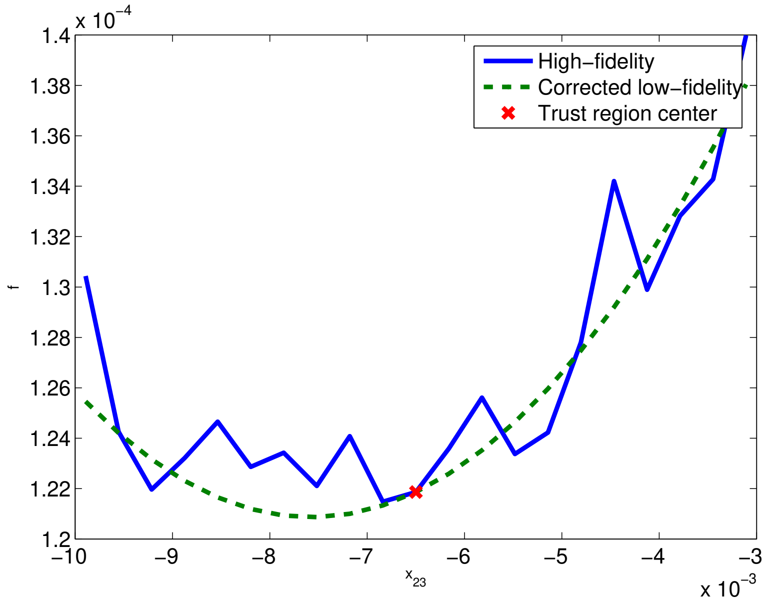

Figure 4-11: The high-fidelity and corrected low-fidelity functions as a function of variable x_{23} for an airfoil close to the goal airfoil, over a smaller range than Figure 4-10.

variable x_{23}, which corresponds to the magnitude of a thickness bump function located approximately a third of the chord behind the leading edge, near the thickest part of the airfoil. Figure 4-10 shows that the corrected low–fidelity function matches the high–fidelity function well over a range of values. Figure 4-11 shows a more detailed section of the same plot — it can be seen that the high–fidelity function is noisy, while the corrected low–fidelity function is smooth. The corrected low–fidelity model is able to capture the second–order trend of the data even when noise is significant.

Further interrogation of the design space shows that the high–fidelity function is even noisier along variables defining the leading and trailing edges than those near the center of the airfoil. Figure 4-12 shows the variations in the corrected low–fidelity and high–fidelity functions with the variable x_1, which corresponds to a bump function in camber near the leading edge. Compared with Figure 4-11, the oscillations in the high–fidelity function are much larger.

Figures 4-10 to 4-12 indicate that it is unrealistic to expect an optimization method to result in objective function values of lower than $O(10^{-4})$ for this problem. At this point, the noise in the high–fidelity function becomes significant and the progress of the optimization method is likely to be impaired due to an inability to compute gradient information accurately with finite differences. A convergence criteria of 10^{-4} was therefore applied to optimization results.

4.3.4 Results

As in previous problems, a quasi–Newton method was used as the benchmark. Figure 4-13 shows the multifidelity method with each of corrected space mapping and POD mapping, along with the benchmark quasi–Newton method. The benchmark method takes 924 high–fidelity function evaluations, including those required for finite–difference gradients, to achieve an objective function value of 10^{-4}. The multifidelity method with corrected space mapping takes 814 function calls to get to the same value, which is a savings of 12%, and the multifidelity method with POD mapping takes 554 function calls, a savings of 40%.

Figure 4-14 shows the varying rates of convergence of the POD method as the

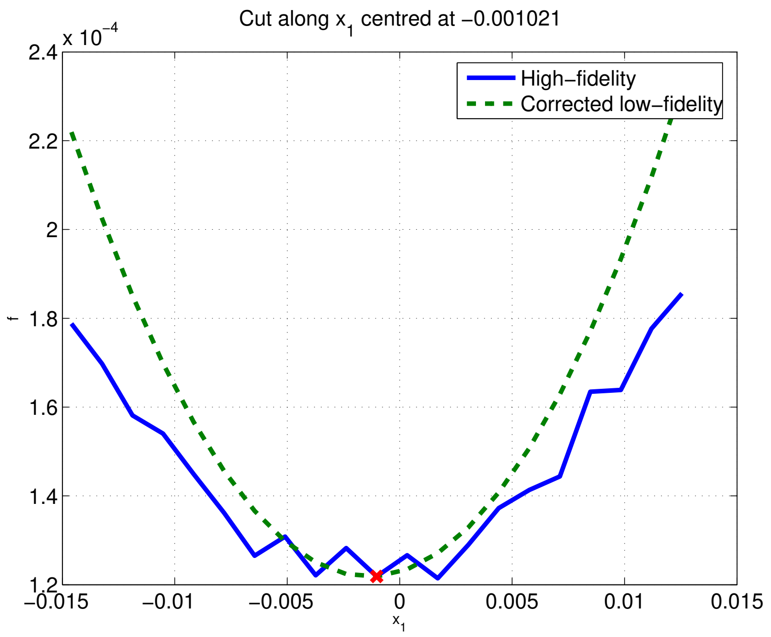

Figure 4-12: The high–fidelity and corrected low–fidelity functions as a function of variable x_1 for an airfoil close to the goal airfoil.

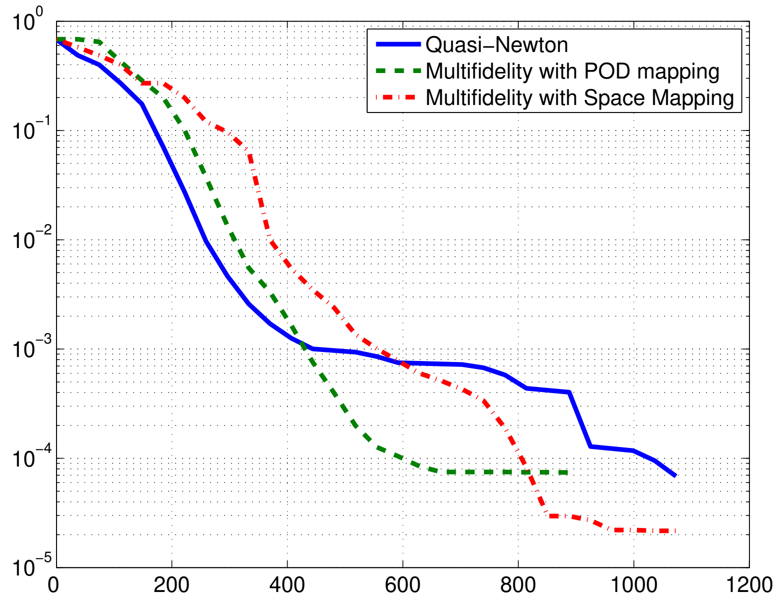

Figure 4-13: Convergence of quasi–Newton method, multifidelity with POD mapping, and multifidelity with corrected space mapping on the airfoil design problem.

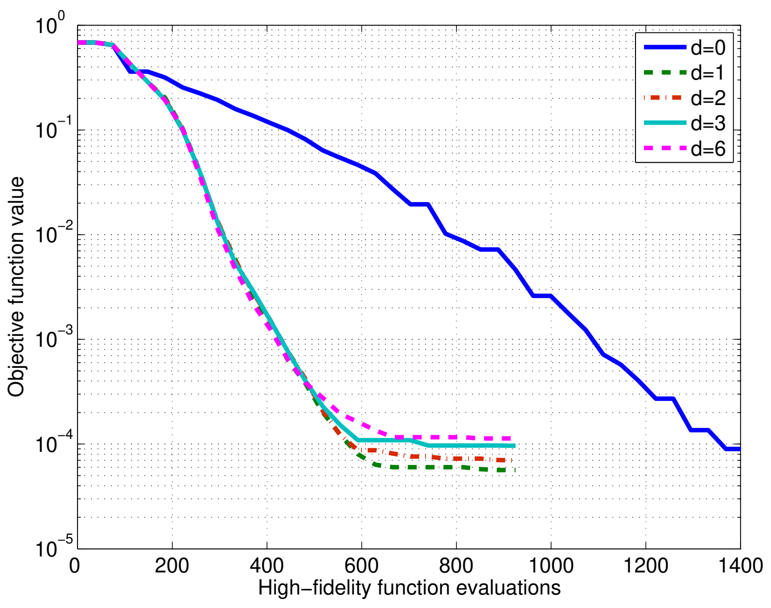

Figure 4-14: Convergence of multifidelity with POD mapping on airfoil design problem varying the number of basis vectors

number of basis vectors is varied. The curve labeled $d = 0$ effectively uses a constant for the low–fidelity analysis. With zero basis vectors, the low–fidelity analysis always evaluates the mean airfoil. Thus, the resulting corrected surrogate function is a second–order Taylor series about the center of the trust region using the BFGS approximation to the Hessian matrix. The remaining curves show convergence with increasing numbers of basis vectors. This shows that nearly the entire mapping relationship is captured by the first basis vector and the remaining basis vectors add very little information and do not significantly improve the convergence rate.

As is generally true for optimization of non–convex functions, the locally–optimal solution to which the algorithm converges depends on the initial choice of design variables. The convergence rates of the multifidelity methods presented here also depend on the initial solution. In most cases, the multifidelity methods were found to yield substantial computational savings when compared with direct high–fidelity optimization; however, for some choices of initial design the improvement was not significant. The multifidelity method was applied to a number of different initial designs. Figure 4-15 shows three representative convergence plots. It can be seen

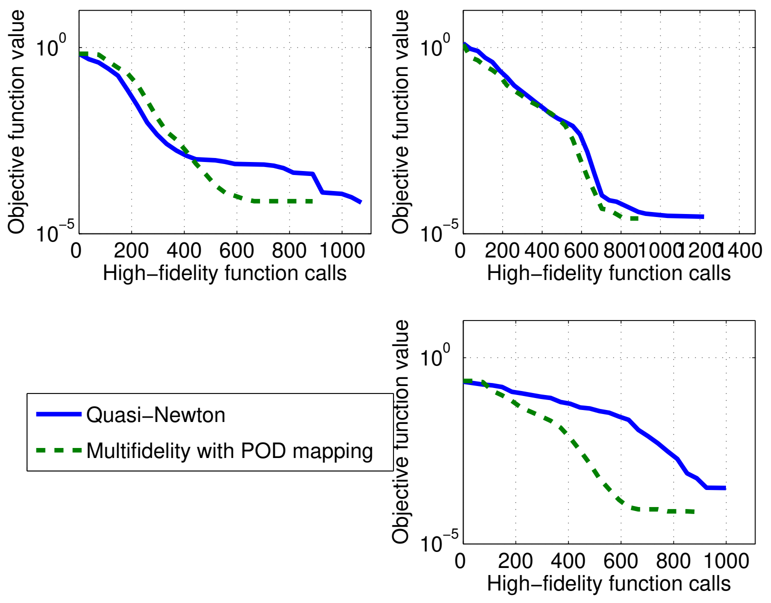

Figure 4-15: Convergence of multifidelity with POD mapping on airfoil design problem varying the initial airfoil.

that the POD multifidelity method is more efficient than the quasi–Newton method for these cases, although for the second case there is very little improvement.

4.3.5 Chapter Summary

This chapter applied the unconstrained multifidelity methods, using both POD mapping and corrected space mapping, to three unconstrained optimization problems. The first problem was a two–dimensional Rosenbrock problem for which the dimension of the design vector was the same in both the high–fidelity and low–fidelity models. The second was a six–dimensional Rosenbrock problem for which the low–fidelity model had fewer design variables than the high–fidelity model. The third problem was a design of an airfoil, for which the low–fidelity model had two design variables and the high–fidelity model had 36 design variables.

In all cases, the multifidelity method achieved the optimum with satisfactory convergence. The computational savings varied from nearly none to approximately 40%. On average, the multifidelity method using POD mapping achieved greater computational savings than that using corrected space mapping.

Chapter 5

Barnes Problem

The previous chapter applied and confirmed the new multifidelity methods on two constrained problems: a simple analytic problem and a real–world design problem. They have shown varying amounts of computational savings, from small to significant. Next the methods will be demonstrated and verified on constrained problems: first, in this chapter, on a simple analytic problem; then, in the next chapter, on real–world design problems. The goal is to verify that the methods converge to a local minimum of the high–fidelity model and determine which methods provide reduced computational costs, relative to one another and to a benchmark single–fidelity method.

The Barnes problem was chosen as the simple analytic problem because it is two–dimensional and therefore easy to visualize. It also has a number of interesting starting points, thereby allowing a demonstration of tradeoffs between minimizing the objective and satisfying the constraints. The Barnes problem has also been used in previous studies.

5.1 Problem Description

This problem was originally developed by G. Barnes [16] as part of an M.S. thesis. It was then used as a demonstration problem in a textbook [44], and has since been used a number of times to demonstrate optimization approaches [60, 55, 56, 54].

The objective function is

$$
\begin{aligned}
f &= -75.196 + 3.81x_1 - 0.126x_1^2 + 2.5056 \times 10^{-3}x_1^3 \\
&- 1.034 \times 10^{-5}x_1^4 6.83x_2 - 0.0302x_1x_2 \\
&+ 1.281 \times 10^{-3}x_2x_1^2 - 3.525 \times 10^{-5}x_2x_1^3 \\
&+ 2.266 \times 10^{-7}x_2x_1^4 - 0.256x_2^2 3.46 \times 10^{-3}x_2^3 \\
&- 1.35 \times 10^{-5}x_2^4 + \frac{28.106}{x_2+1} + 5.237 \times 10^{-6}x_1^2x_2^2 \\
&+ 6.3 \times 10^{-8}x_1^3x_2^2 + 1.663 \times 10^{-6}x_1x_2^3 + 2.867e^{0.0005x_1x_2}.
\end{aligned} \tag{5.1}
$$

The constraints and variable bounds are

$$c_1 = -(x_1x_2/700 - 1) \le 0, \tag{5.2}$$

$$c_2 = -(x_2/5 - x_1^2/625) \le 0, \tag{5.3}$$

$$c_3 = -(x_2/50 - 1)^2 - (x_1/500 - 0.11) \le 0, \tag{5.4}$$

$$0 \le x_1 \le 80, \tag{5.5}$$

$$0 \le x_2 \le 80. \tag{5.6}$$

The problem has a global minimum at $\mathbf{x} = (80, 80)$ with an objective value of $f = -132.876$ and a local minimum at $\mathbf{x} = (49.526, 19.622)$ with an objective value of $f = -31.6372$. At that local minimum only the second constraint is active. Contours of the objective and lines showing the constraints are shown in Figure 5-1.

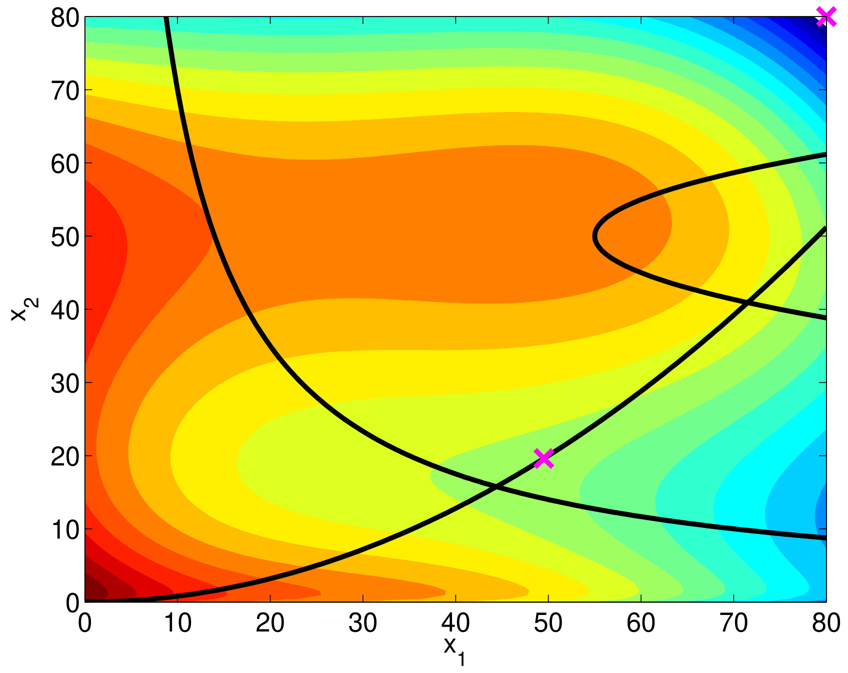

Figure 5-1: Objective function contours and constraints of the Barnes problem. Local and global optima are indicated with an 'x'.

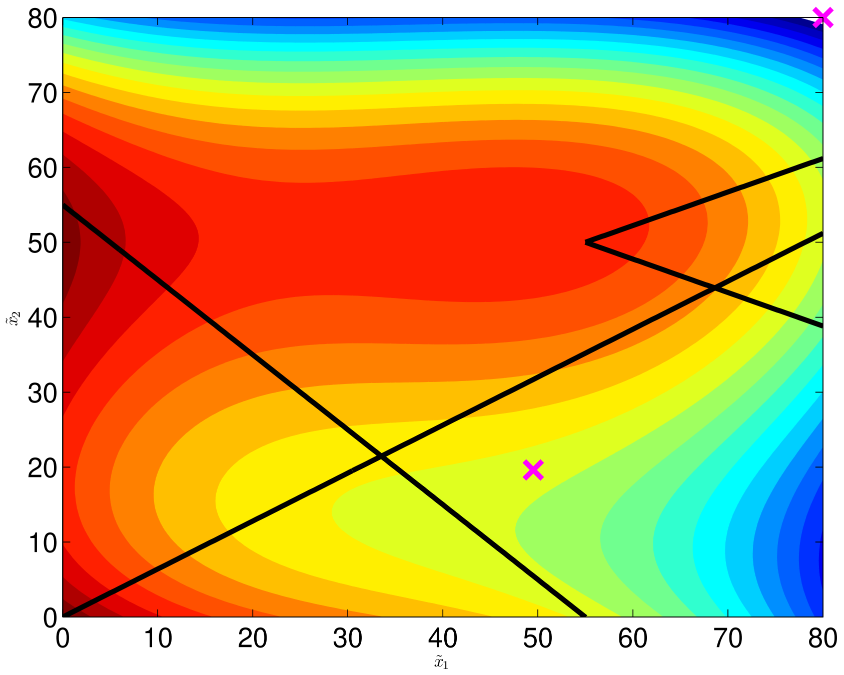

Figure 5-2: Objective function contours and constraints of the low–fidelity approximation to the Barnes problem. Optima of the high–fidelity problem are indicated with an 'x'.

5.2 Low–fidelity Model

The chosen low–fidelity objective function is a third–order Taylor series expansion of the high–fidelity function about the point $\mathbf{x} = (30, 40)$

$$\begin{aligned}\hat{f} &= -2.744 + 0.01214(\hat{x}_1 - 30) + 0.009957(\hat{x}_1 - 30)^2 - 5.557 \times 10^{-4}(\hat{x}_1 - 30)^3 \\ &+ \left(1.1508 + 0.009473(\hat{x}_1 - 30) + 2.9948 \times 10^{-5}(\hat{x}_1 - 30)^2\right)(\hat{x}_2 - 40) \qquad (5.7) \\ &+ \left(-0.02997 - 1.676 \times 10^4(\hat{x}_1 - 30)\right)(\hat{x}_2 - 40)^2 - 0.001322(\hat{x}_2 - 40)^3.\end{aligned}$$

The low–fidelity constraints are linear functions chosen to be in approximately the same location as the quadratic curves of the exact constraints over the global space. The equations for these low–fidelity constraints are

$$\hat{c}_1 = (-\hat{x}_1 - \hat{x}_2 + 50)/10 \leq 0, \qquad (5.8)$$

$$\hat{c}_2 = (0.64\hat{x}_1 - \hat{x}_2)/6 \leq 0, \qquad (5.9)$$

$$\hat{c}_3 = \begin{cases} 0.006\hat{x}_1 - 0.0134\hat{x}_2 + 0.34 & \leq 0 \quad \text{if } \hat{x}_2 > 50, \\ 0.006\hat{x}_1 + 0.0134\hat{x}_2 - 1 & \leq 0 \quad \text{if } \hat{x}_2 \leq 50. \end{cases} \qquad (5.10)$$

The objective function contours and constraints used for the low-fidelity model are shown in Figure 5-2.

5.3 Constraint–management Methods

The Barnes problem was run with each of the constraint–management methods described in Section 2.2 combined with each of corrected space mapping and POD mapping.

Three starting points were used. The point $\mathbf{x}^0 = (30, 40)$ is a feasible starting point, while $\mathbf{x}^0 = (65, 5)$ and $\mathbf{x}^0 = (10, 20)$ are infeasible starting points. At $\mathbf{x}^0 = (10, 20)$, the directions towards feasibility and optimality are near–orthogonal.

Table 5.3 shows the number of high–fidelity function evaluations taken for each of the methods to converge to a point at which the objective function is within 10^{-2} of the minimum and the constraints are satisfied to within 10^{-2}. As a benchmark, a single–fidelity SQP method was run on the same problem. The reduction in computational complexity of the multifidelity methods was measured as a ratio of the number of high–fidelity function calls taken by the multifidelity method to the number taken by the benchmark SQP method for each starting point. For each method, the average ratio from all three starting points is also shown in the table.

The table shows that the augmented Lagrangian methods and MAESTRO take, on average, more high–fidelity function evaluations than the benchmark single–fidelity SQP. Two methods resulted in computational savings on average: direct surrogate optimization and the SQP–like method. According to these measures, direct surrogate optimization is the most efficient method. It performs slightly better when combined with corrected space mapping than with POD mapping.

Constraint Management	Mapping Method	From (30,40) Calls	Ratio	From (10,20) Calls	Ratio	From(65,5) Calls	Ratio	**Average Ratio**
SQP in high–fidelity space	N/A	10	1.00	12	1.00	8	1.00	**1.000**
Approx. whole Lagrangian	POD	16	1.60	35	2.92	34	4.25	**2.922**
Approx. whole Lagrangian	SM	49	4.90	38	3.17	33	4.12	**4.064**
Lagrangian w/ sep. approx.	POD	30	3.00	35	2.92	11	1.38	**2.431**
Lagrangian w/ sep. approx.	SM	21	2.10	39	3.25	29	3.62	**2.992**
Direct surrogate approx.	POD	10	1.00	4	0.33	8	1.00	**0.778**
Direct surrogate approx.	SM	7	0.70	4	0.33	10	1.25	**0.761**
SQP–like	POD	7	0.70	11	0.92	10	1.25	**0.956**
SQP–like	SM	8	0.80	11	0.92	7	0.88	**0.864**
MAESTRO	POD	20	2.00	40	3.33	15*	1.88*	**2.403**
MAESTRO	SM	27	2.70	43	3.58	14	1.75	**2.678**

Table 5.1: Convergence of various optimization methods, with both POD mapping and corrected space mapping, from three starting points. *Converged to the global optimum rather than the local optimum.

It can be concluded from these trials that the most promising methods are the SQP–like method and direct surrogate optimization. MAESTRO should be reserved for the applications for which it was designed: multidisciplinary problems solved over a distributed system. The augmented Lagrangian methods have no significant advantages over the SQP–like method or direct surrogate optimization, and should be rejected in favor of one of the latter.

5.4 Variation of Parameters

Sensitivity studies were performed varying many of the parameters of the algorithms in order to determine the optimal parameters and the effect of the parameters on the convergence. The variation was performed for three combinations of methods: direct surrogate optimization with POD mapping; the augmented Lagrangian method with separate corrections to the objective and each constraint with POD mapping; and direct surrogate optimization with space mapping. This allows the variation of parameters that are only relevant in some subset of the methods. For example, the number of POD basis functions is only relevant in POD mapping and the number of space mapping sample points is only relevant in space mapping.

Figures are only shown in the case where the parameter in question has an effect on the convergence of the method. When a parameter has similar effects on all three methods, only one set of figures is shown.

5.4.1 Low–fidelity objective function

The low–fidelity objective function was varied in order to determine whether a more accurate low–fidelity function provided improved convergence.

Figures 5-3 and 5-4 show the variation of parameters on the direct surrogate optimization method using POD mapping. They show that, even using a simple quadratic as a low–fidelity objective function, the multi–fidelity method converges more quickly than the benchmark single–fidelity method. Using a second–order Taylor series (expanded around the point (30,40)) of the high–fidelity function provides improved convergence. However, using a third–order Taylor series expansion of the objective function did not provide additional computational savings over the second–order Taylor series. While the former reduced the objective slightly more quickly, the latter reduced the constraint violation more quickly, resulting in them both finding the optimum after the same number of high–fidelity function calls.

This would seem to indicate that the Taylor series are more useful than the simple quadratic, since they are more similar to the high–fidelity model. It was expected that the third–order Taylor series would prove more useful than the second–order Taylor series; however, this was not shown to be the case on this problem. It is possible that this problem is too simple to show a clear difference. These methods, which use additive corrections, are expected to converge most quickly when the difference between the high–fidelity model and low–fidelity model is smooth and can be well approximated by a quasi–second–order correction function. Knowledge of the relationship between the high–fidelity and low–fidelity models may aid the designer in choosing between additive, multiplicative, and combination correction functions. The lack of significant dependence on the quality of the low–fidelity model may be due to the quasi–second–order additive corrections, which provide excellent surrogates even when the underlying low–fidelity model is poor. Stronger dependence on the low–fidelity model are expected with less accurate corrections, such as those that enforce only first–order consistency.

5.4.2 Low–fidelity constraints

Similarly, the accuracy of the low–fidelity constraints was varied. As Figures 5-5 and 5-6 show, the choice between linear constraints and the exact high–fidelity constraints had little effect on the path and none on the convergence rate of direct surrogate optimization. This was true both for POD mapping, shown in the figures, and corrected space mapping.

However, for the augmented Lagrangian multifidelity method, the choice of low–fidelity constraints changed the solution. Figures 5-7 and 5-8 show that the method using the exact constraints converged to a local minimum, while the problem with the linear low–fidelity constraints converged to the global minimum. This shows that small changes in the low–fidelity model can produce small changes in the early steps, resulting in different paths and possibly different solutions.

The constraints on the Barnes problem are quadratic, and the corrections are quasi–second order. Therefore, on each trust–region iteration, the corrected surrogate constraints are nearly exact, independent of the choice of the underlying low–fidelity constraint functions. Therefore, the early steps taken will vary only slightly

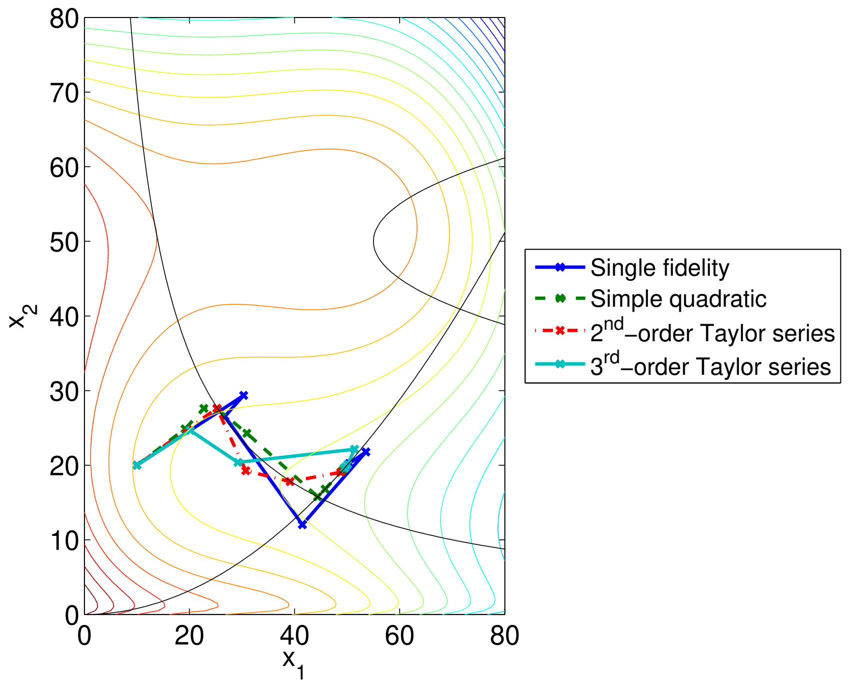

Figure 5-3: Optimization paths, in the high–fidelity space, of direct surrogate optimization, using POD mapping, varying the low–fidelity objective function.

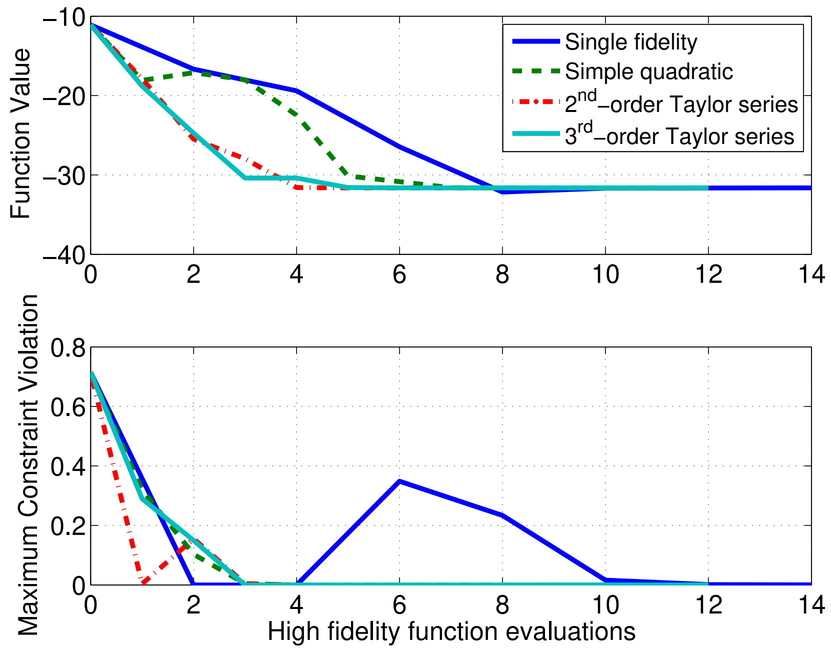

Figure 5-4: Objective function and maximum constraint violation of direct surrogate optimization, using POD mapping, varying the low–fidelity objective function.

depending on the accuracy of the low–fidelity constraints. However, as the augmented Lagrangian shows, even these slight variations can accumulate to different results.

It is expected that on realistic design problems, with non–quadratic constraints — and therefore less accurate surrogate constraints — the path may depend even more strongly on the accuracy of the low–fidelity constraint functions.

Earlier results showed that direct surrogate optimization, on average, converges more quickly than the benchmark SQP method on this problem. They also showed that the augmented Lagrangian multifidelity method converges less quickly on average. The results in this section show that this ordering, with direct surrogate optimization faster than the benchmark method, which is in turn faster than the augmented Lagrangian method, does not change with the accuracy of the low–fidelity constraints. Therefore, even if the low–fidelity constraints are relatively inaccurate, direct surrogate optimization is expected to achieve convergence to a local minimum with computational savings.

Figure 5-9 shows how the surrogate to the constraints improve over the course of optimization. In the first iteration, labeled $k = 0$, little information is known about the surrogates, and they are inaccurate. As the algorithm progresses, the surrogates are increasingly corrected to become more accurate. The boundaries of the two active and near–active constraints become accurate more quickly than those of the inactive constraint. After only three iterations, the two most relevant constraints are very accurate in the region surrounding the optimum. The local accuracy of the constraints improves with successive iterations. The boundaries of the inactive constraint are inaccurate; however, this does not affect the convergence of the method.

5.4.3 Trust–region algorithm parameters

Next, the trust–region algorithm parameters were varied. First, the parameter c_1 was varied. This parameter is the factor by which the trust–region radius is decreased when the improvement attained at a trial step is negative or small. The method is provably convergent as long as $0 < c_1 < 1$. The value of this parameter has very little effect on the convergence of the methods on this problem, and figures are not

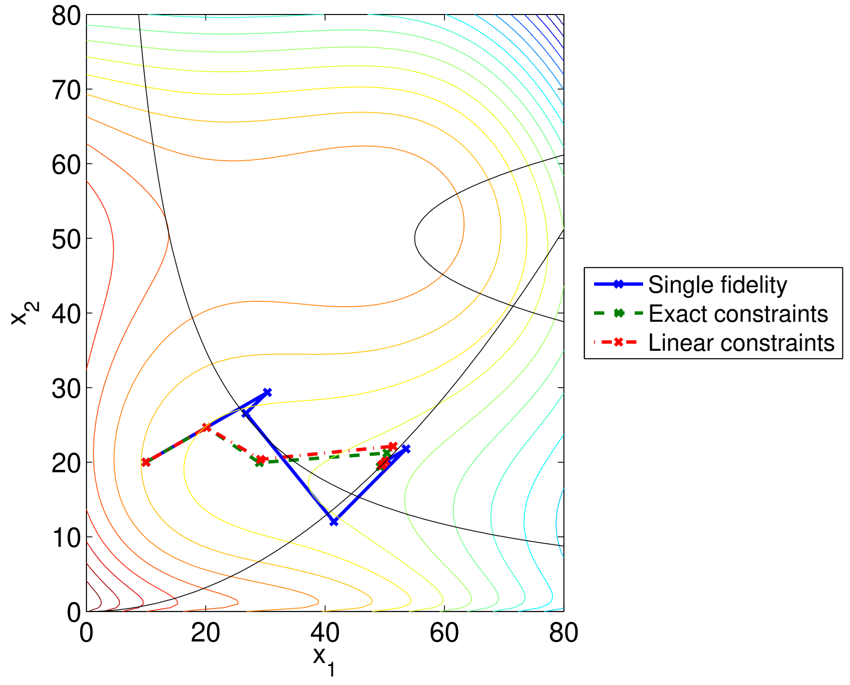

Figure 5-5: Optimization paths, in the high–fidelity space, of direct surrogate optimization, using POD mapping, varying the low–fidelity constraints.

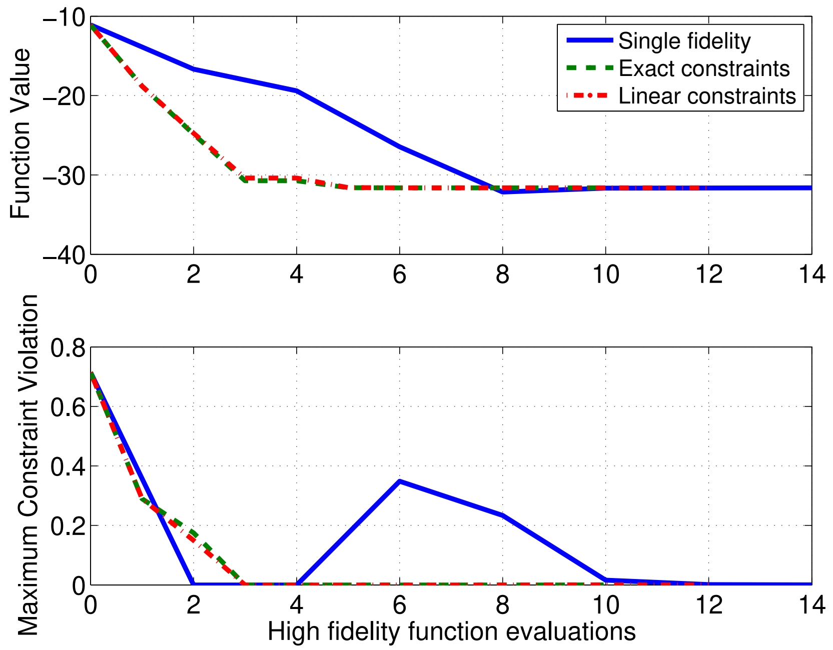

Figure 5-6: Objective function and maximum constraint violation of direct surrogate optimization, using POD mapping, varying the low–fidelity constraints.

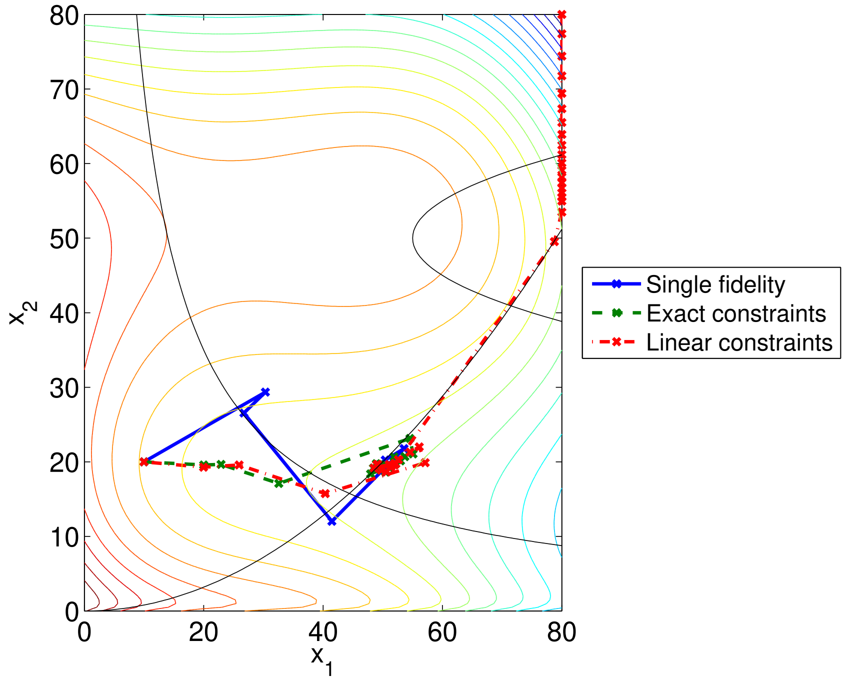

Figure 5-7: Optimization paths, in the high–fidelity space, of the augmented Lagrangian multifidelity method, using separate corrections to the objective and the constraints and POD mapping, varying the low–fidelity constraints.

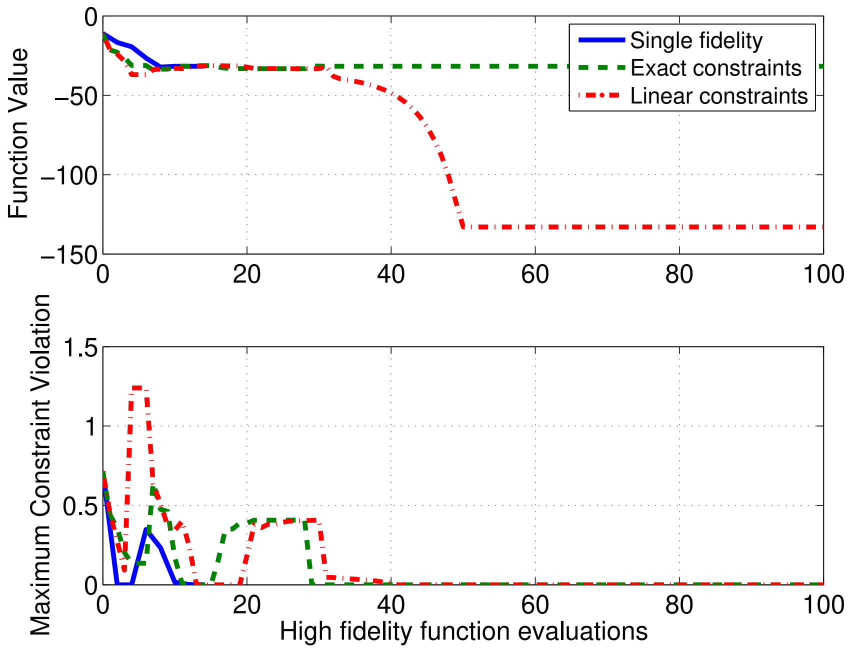

Figure 5-8: Objective function and maximum constraint violation of the augmented Lagrangian multifidelity method, using separate corrections to the objective and the constraints and POD mapping, varying the low–fidelity constraints.

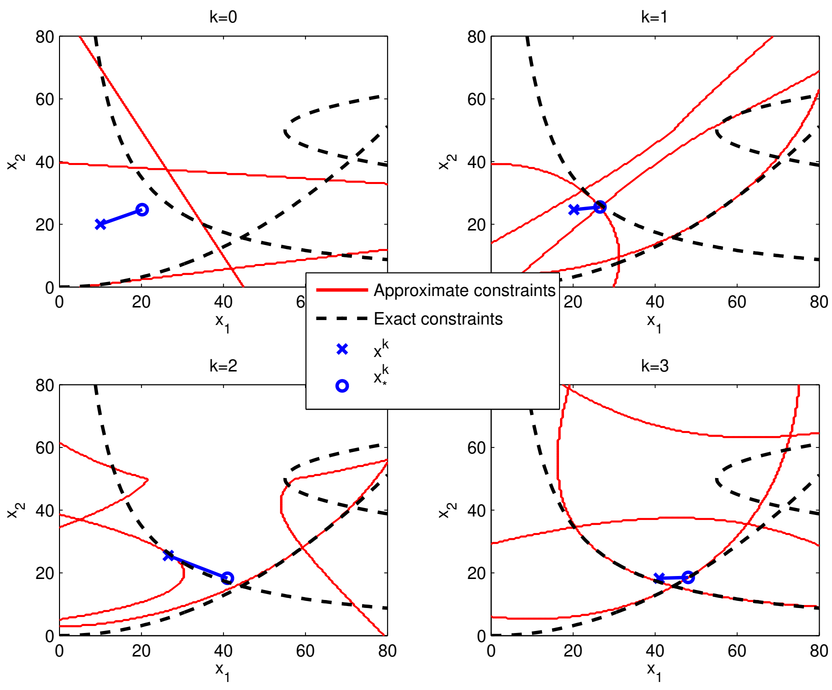

Figure 5-9: Corrected surrogates to constraints for the first four trust–region iterations of direct surrogate optimization, using POD mapping.

shown. Therefore, using the advice in the book by Conn, Gould, and Toint, [22], the standard value of c_1 in the remaining problems was set at $c_1 = 0.25$.

The next parameter varied was c_2, the factor by which the trust–region radius is increased when the improvement at a trial step is large. The path taken changed as this parameter varied; however, the convergence rate remained constant for any value of c_2 greater than 2. When c_2 is less than 2, the convergence rate suffers. Figures 5-10 and 5-11 show the results using direct surrogate optimization with POD mapping; the results followed the same trend for the other two methods. This confirms that c_2 should always be at least 2 so as to allow for the method to take appropriately large trial steps in the case of an accurate surrogate model. The results on this problem do not show an upper limit on c_2. Conn, Gould, and Toint, however, performed a similar analysis on 26 Constrained and Unconstrained Testing Environment (CUTE) test problems[17], and found that on average the best value for c_2 was approximately 2.5.

Next, r_1 was varied; the trust–region ratio below which the trust–region radius is

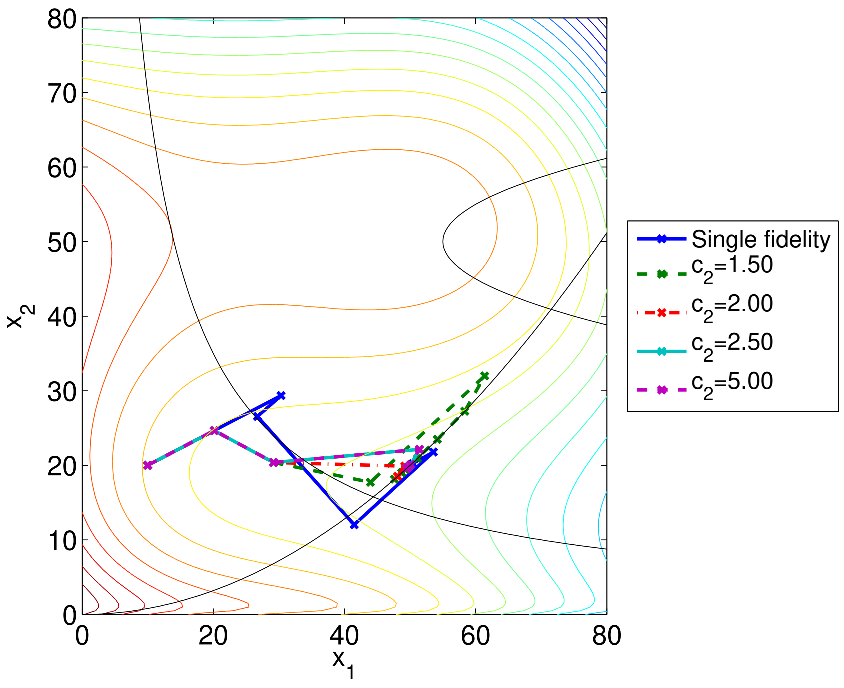

Figure 5-10: Optimization paths, in the high–fidelity space, of direct surrogate optimization, using POD mapping, varying the trust–region algorithm parameter c_2.

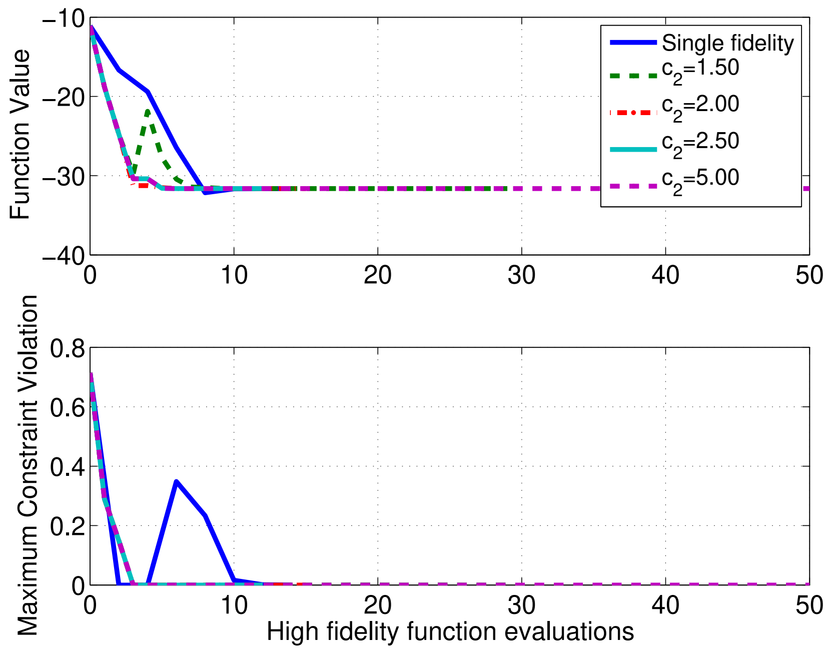

Figure 5-11: Objective function and maximum constraint violation of direct surrogate optimization, using POD mapping, varying the trust–region algorithm parameter c_2.

decreased. The performance of direct surrogate optimization on this problem does not depend on r_1. Figures 5-12 and 5-13 show that the path taken by the augmented Lagrangian multifidelity method, and even the final point to which it converges, can depend on this parameter, but in a non–monotonic way. That is, using the two smallest and one largest value of the parameter, the augmented Lagrangian method converged to the global optimum, but using the two values in between, it converged to the local optimum. Conn, Gould, and Toint showed that the method convergence was also relatively insensitive to this parameter on their CUTE test problems, and they suggest $r_1 = 0.05$.

The convergence depends slightly more on r_2, the trust–region ratio above which the trust–region radius is increased. Figures 5-14 and 5-15 show that at values of r_2 from 0.2 to 0.8 have slower convergence than values of 0.9 and 0.95. The implication is that the trust–region size should only be increased if the surrogate is predicting the descent with very good accuracy. While figures are only shown for direct surrogate optimization with POD mapping, the same trends were observed in direct surrogate optimization with space mapping and in the augmented Lagrangian method with POD mapping. Conn, Gould, and Toint suggest a value of 0.9, which is used in the remainder of this work.

In sum, these trials showed that variable–parameterization problems are not notably different from existing variable–fidelity problems in their dependence on the algorithm parameters, and that recommendations by previous researchers in variable–fidelity methods are applicable.

5.4.4 Initial algorithm parameters

Some initial algorithm parameters were also varied. The convergence of none of the three methods depended on the initial penalty parameter μ^0 or the initial convergence tolerances η^0, and ω^0.

The initial trust–region radius Δ^0, however, had an effect. Figures 5-16 and 5-17 show that for $\Delta^0 = 1$, direct surrogate optimization converged to the global minimum, while for other values it converged to the local minimum. In general, larger values

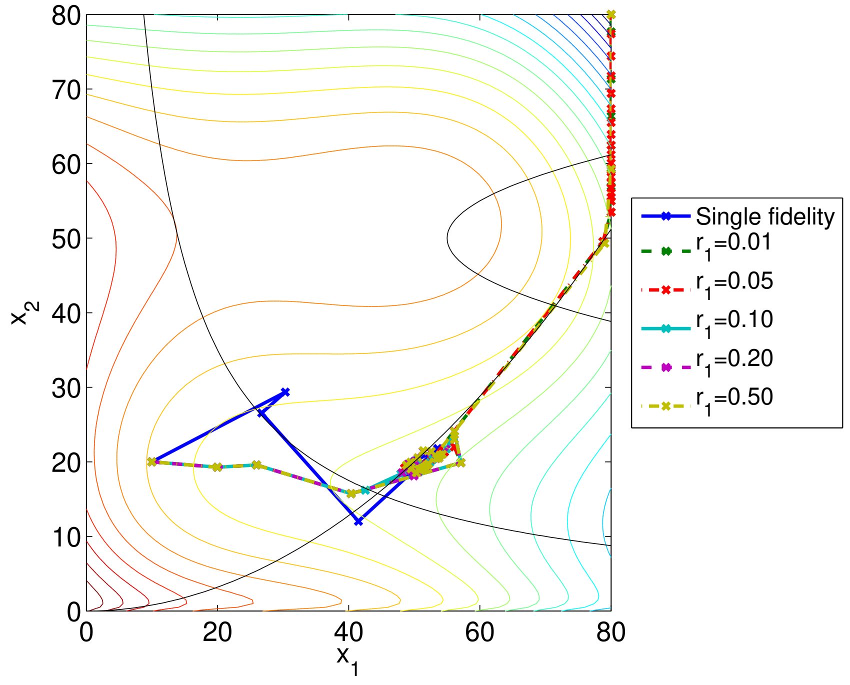

Figure 5-12: Optimization paths, in the high–fidelity space, of the augmented Lagrangian multifidelity method, using separate corrections to the objective and each constraint and POD mapping, varying the trust–region algorithm parameter r_1.

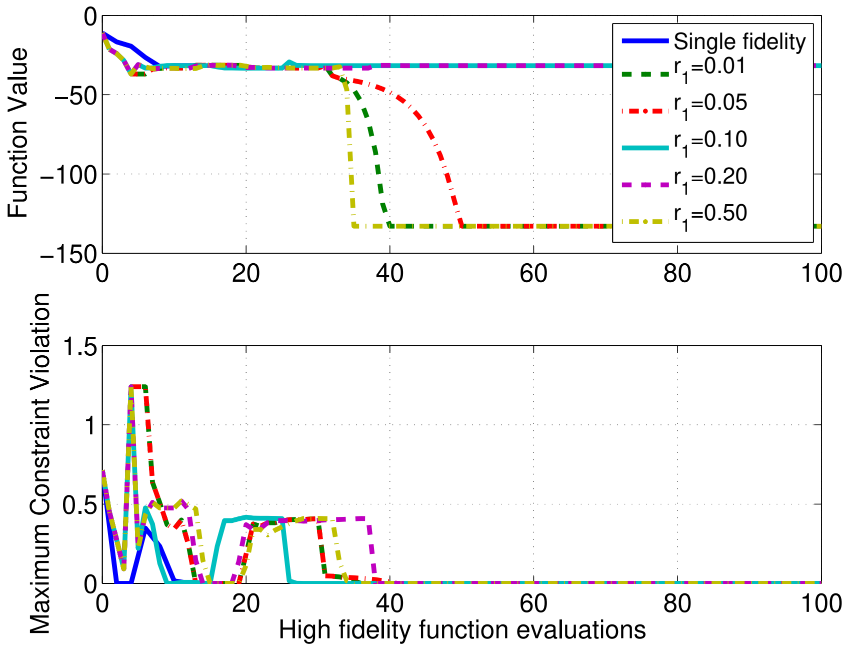

Figure 5-13: Objective function and maximum constraint violation of the augmented Lagrangian multifidelity method, using separate corrections to the objective and each constraint and POD mapping, varying the trust–region algorithm parameter r_1.

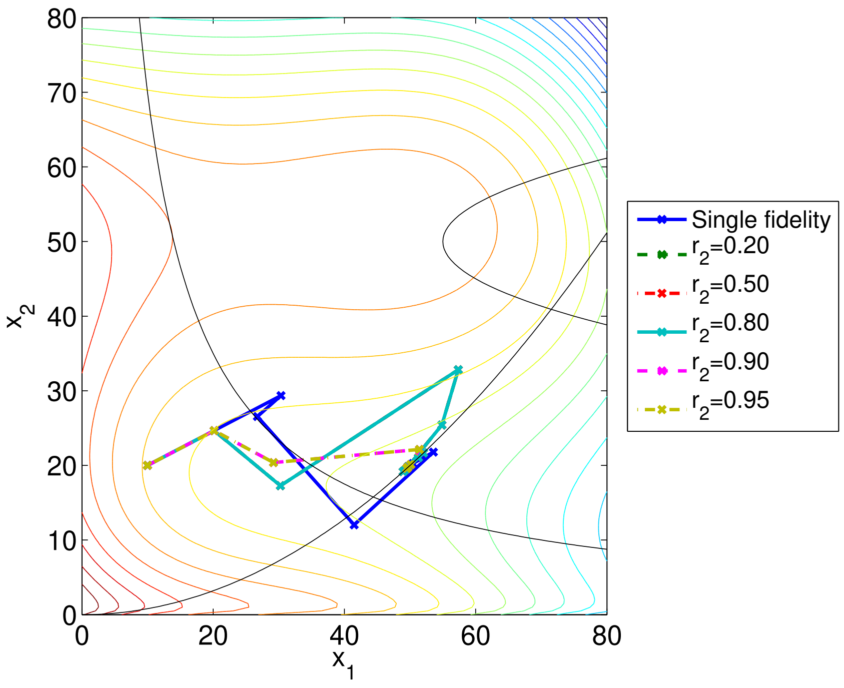

Figure 5-14: Optimization paths, in the high–fidelity space, of direct surrogate optimization, using POD mapping, varying the trust–region algorithm parameter r_2.

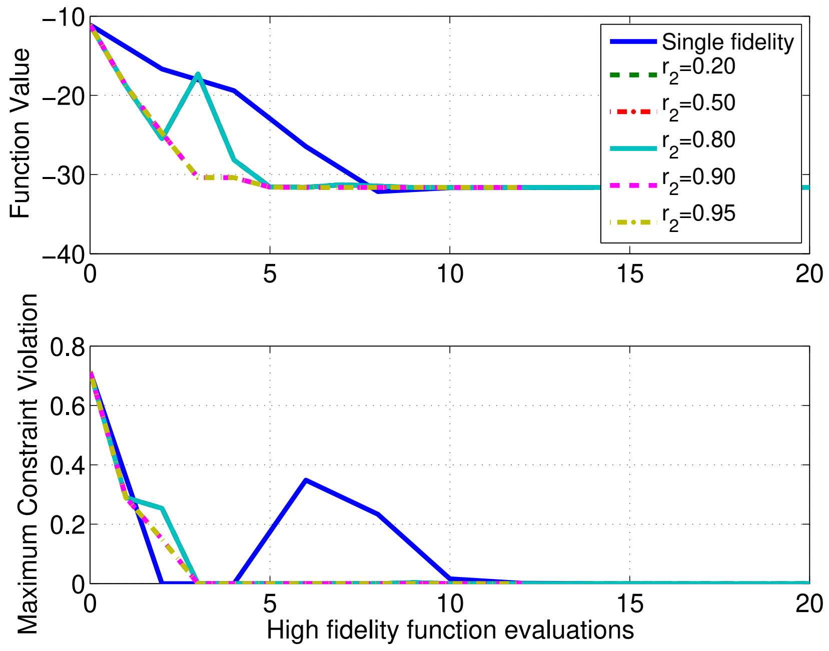

Figure 5-15: Objective function and maximum constraint violation of direct surrogate optimization, using POD mapping, varying the trust–region algorithm parameter r_2.

of Δ^0 converged more quickly than smaller values. This is because the method can take larger steps early in the optimization, rather than a number of small steps each requiring high–fidelity function calls. The results show similar trends for the other two methods.

Setting Δ^0 may require expertise on the part of the designer, using a subjective estimate of the size of a region near the initial point for which the low–fidelity and high–fidelity models will have similar trends.

5.4.5 POD mapping parameters

The number of POD basis vectors, d, was varied for the two relevant methods — direct surrogate optimization with POD mapping and augmented Lagrangian with POD mapping. Figures 5-18 and 5-19 show that with only one basis vector, direct surrogate optimization converged to the global optimum, rather than the local optimum to which most of the variations of the method converged. With three basis vectors, one of the basis vectors was spurious, and generated noise, causing the methods to fail to make progress from the initial point. Similar results were found for the augmented Lagrangian method with POD mapping. These results underscore the importance of using no more basis vectors than the rank of the snapshot matrix. This study is limited by its use in a two–dimensional problem. Higher–dimensional problems will need more basis vectors, and the relationship between the number of basis vectors used and the rate of convergence of the optimization algorithm may be more complex.

5.5 Chapter summary

This chapter has shown that all multifidelity methods discussed so far converge to a local minimum of the Barnes problem. However, only two of the constraint–management methods, the SQP–like multifidelity method and direct surrogate optimization, achieve computational savings on average for this problem. POD mapping and corrected space mapping appeared to provide approximately equal computational savings.

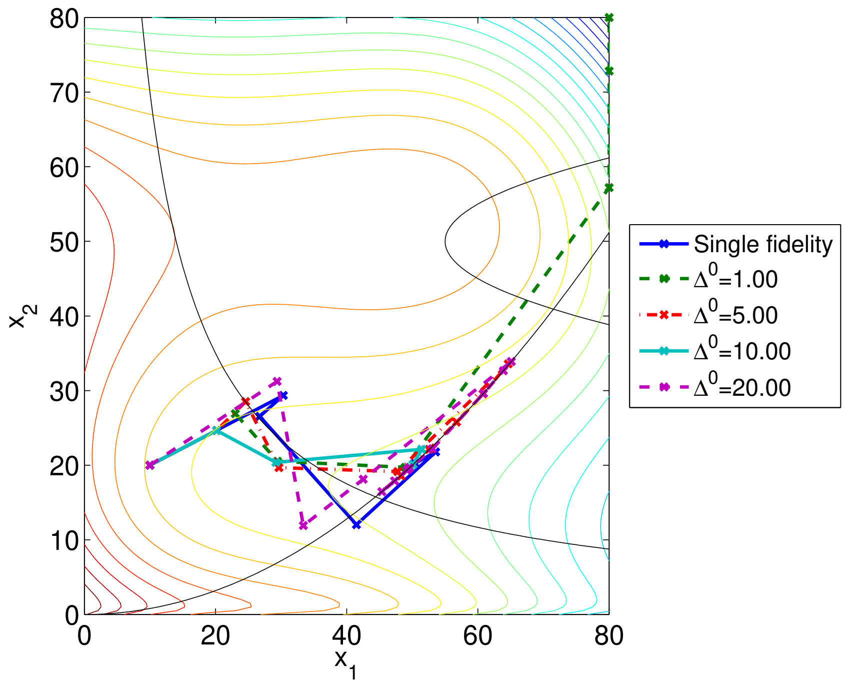

Figure 5-16: Optimization paths, in the high–fidelity space, of direct surrogate optimization, using POD mapping, varying the initial trust–region radius Δ^0.

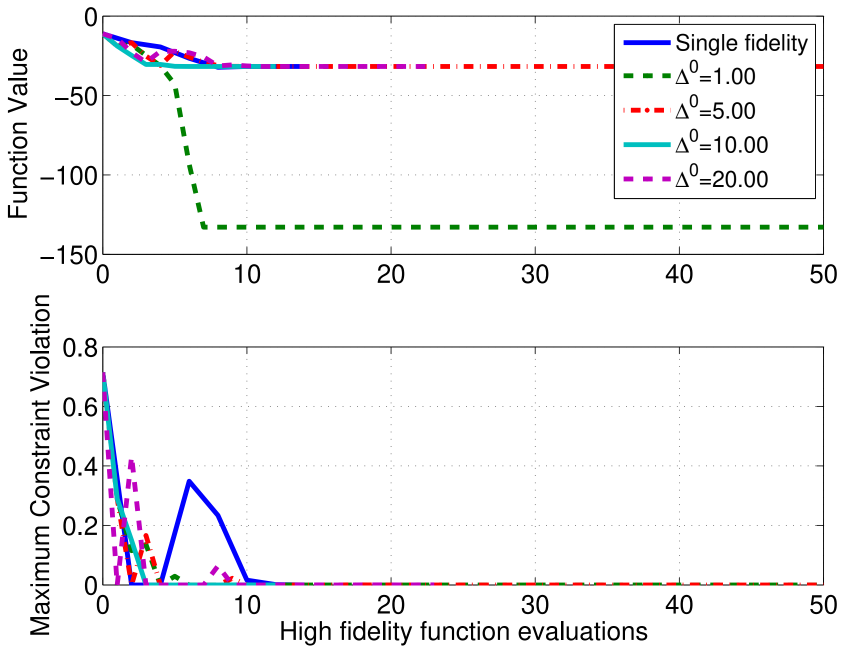

Figure 5-17: Objective function and maximum constraint violation of direct surrogate optimization, using POD mapping, varying the initial trust–region radius Δ^0.

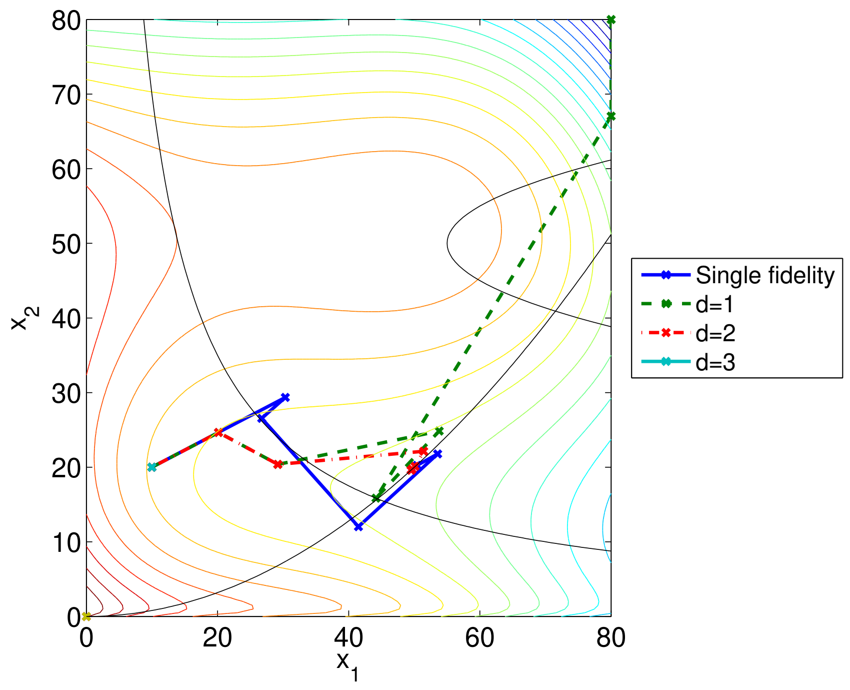

Figure 5-18: Optimization paths, in the high–fidelity space, of direct surrogate optimization, using POD mapping, varying the number of POD basis vectors d.

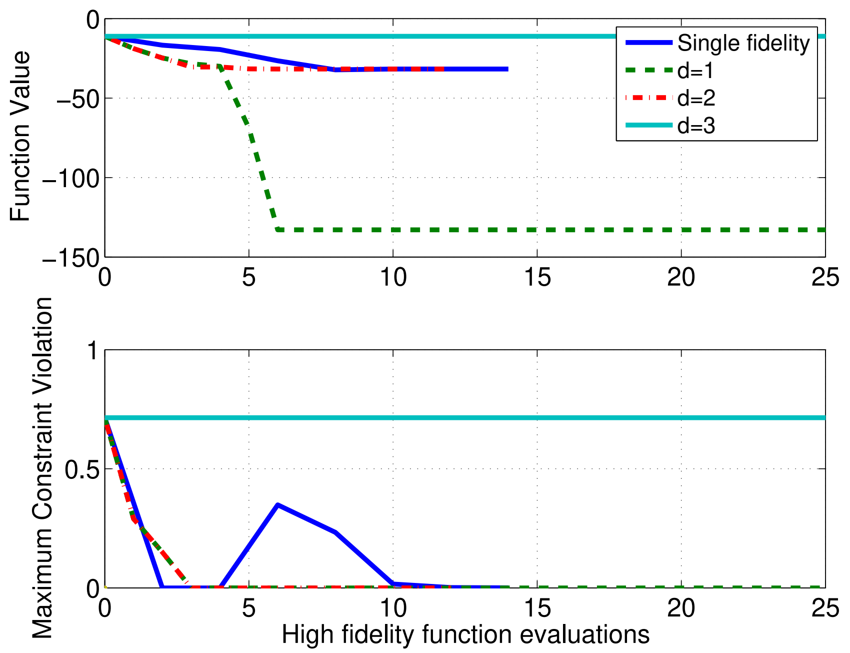

Figure 5-19: Objective function and maximum constraint violation of direct surrogate optimization, using POD mapping, varying the number of POD basis vectors d.

The accuracy of the low–fidelity model was varied. On this problem, more accurate low–fidelity models, analyzed separately as objective and constraint accuracy, showed only marginal computational advantages. However, it is expected that in more complex real–world design problems, the accuracy of the algorithm will have a more significant effect.

A systematic variation in algorithm parameters was then performed, in order to provide guidance for future users of these methods in the choice of values or methods for determining these values. The values of the algorithm parameters μ^0, η^0, ω^0, and r_1 had little to no effect on the performance of the methods. In addition, for the example using corrected space mapping, the number of space mapping sample points had no effect. The trust–region parameters r_1, r_2 and c_1, however, showed an effect on the convergence, and specific constant values were suggested. The initial trust–region size Δ^0 impacted the convergence of the method; however, no specific guidelines can be given on its size. The designer must rely on experience and judgement to choose Δ^0.

Chapter 6

Constrained Design Problems

Chapter 4 showed results of the multifidelity methods with design parameterization mapping both on unconstrained analytic problems and on an unconstrained engineering design problem. Chapter 5 then presented results of the various constrained optimization methods on a constrained analytic problem. This chapter concludes the testing of the methods with two constrained engineering design problems. The first is a design of a conventional aircraft wing and the second is the design of the flapping motion of a bat wing.

6.1 Wing Design Problem

The first constrained design problem is planform design of a wing. The wingspan is constant at 10 metres and the angle of attack is set to a constant value of 0.5 degrees. The quarter chord is unswept. The objective function is the coefficient of induced drag, and the lift of the wing is constrained from below. This problem, therefore, is linked to a more complex aircraft design problem in which the lift must be equal to the weight while the drag of the aircraft is minimized. The design variables specify the chord at each of 10 evenly–distributed points along the wing. The optimization problem is given by

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = C_{Di}(\mathbf{x}) \\ \text{subject to} \quad & c(\mathbf{x}) = (0.2 - A(\mathbf{x}) * C_L(\mathbf{x})) \leq 0 \\ & 0.01 \leq x_j \leq 10, \quad j = 1 \ldots 10, \end{aligned} \tag{6.1}$$

where C_{Di} is the coefficient of induced drag, C_L is the coefficient of lift, $A(\mathbf{x})$ is the wing area, $\mathbf{x}$ is a vector containing the chord design variables, and x_j is the chord at the j^{th} spanwise station.

The high–fidelity code uses a vortex–lattice method, using a zero–thickness, constant–collocation, doublet–lattice model [46]. By imposing Neumann boundary conditions and a zero–spanwise–vorticity trailing–edge Kutta condition, the wing and wake surface doublet strengths can be uniquely determined. The discretization is performed using quadrilateral panels with uniform chordwise spacing and cosine spanwise refinement. A standard Trefftz–Plane analysis [46] is used to compute both the lift and induced drag. A single analysis of the high–fidelity code takes approximately 90 seconds on a 2.4 GHz Intel Pentium 4 desktop workstation.

The low–fidelity code uses lifting–line theory. A standard lifting–line method following Ashley and Landahl [9] has been implemented to compute both the lift and the induced drag. Because it assumes that the wing–bound vorticity can be approximated by a single spanwise line vortex, the lifting–line method is less accurate than the high–fidelity vortex–lattice method. The low–fidelity design variables are also chords, but in this case at only three points, again equally spaced from the root to the tip. This problem is therefore an example of variable parameterization: the low–fidelity design space is of lower dimension than the high–fidelity design space. A single analysis of the low–fidelity code takes approximately 30 milliseconds on a 2.4 GHz desktop workstation.

For both the high–fidelity and low–fidelity models, gradients were calculated using centered finite differences. The evaluations required for these finite differences are included in the count of the number of function calls. The benchmark method is

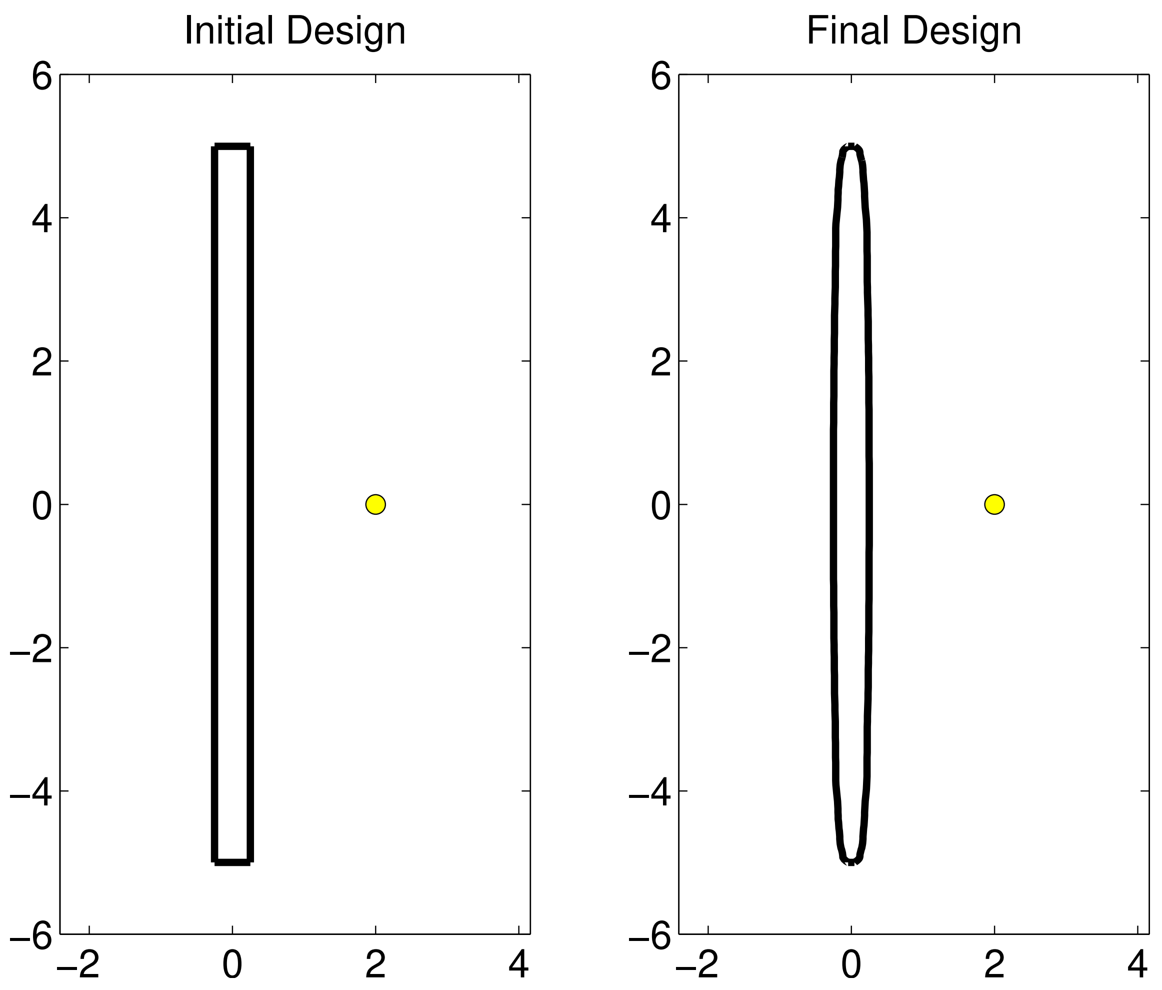

Figure 6-1: Initial and final wing planforms

SQP. The multifidelity method is the SQP–like trust–region method in conjunction with corrected space mapping. The eight previous iterates were used as the control points. The linear space mapping of equation (3.2) was used. The initial design was a rectangular wing and was feasible, with the lift constraint inactive.

Given infinite degrees of freedom, the planform that yields minimum induced drag is elliptic. The optimization problem is expected to find a distribution which most closely approximates an elliptical planform. Figure 6-1 shows the initial and final planforms. Figure 6-2 shows the objective function value and constraint violation of each method versus the number of high–fidelity function calls. Both the benchmark method and the multifidelity method converged to the same near–elliptic design. The high–fidelity SQP method took 1344 high–fidelity function calls to achieve the

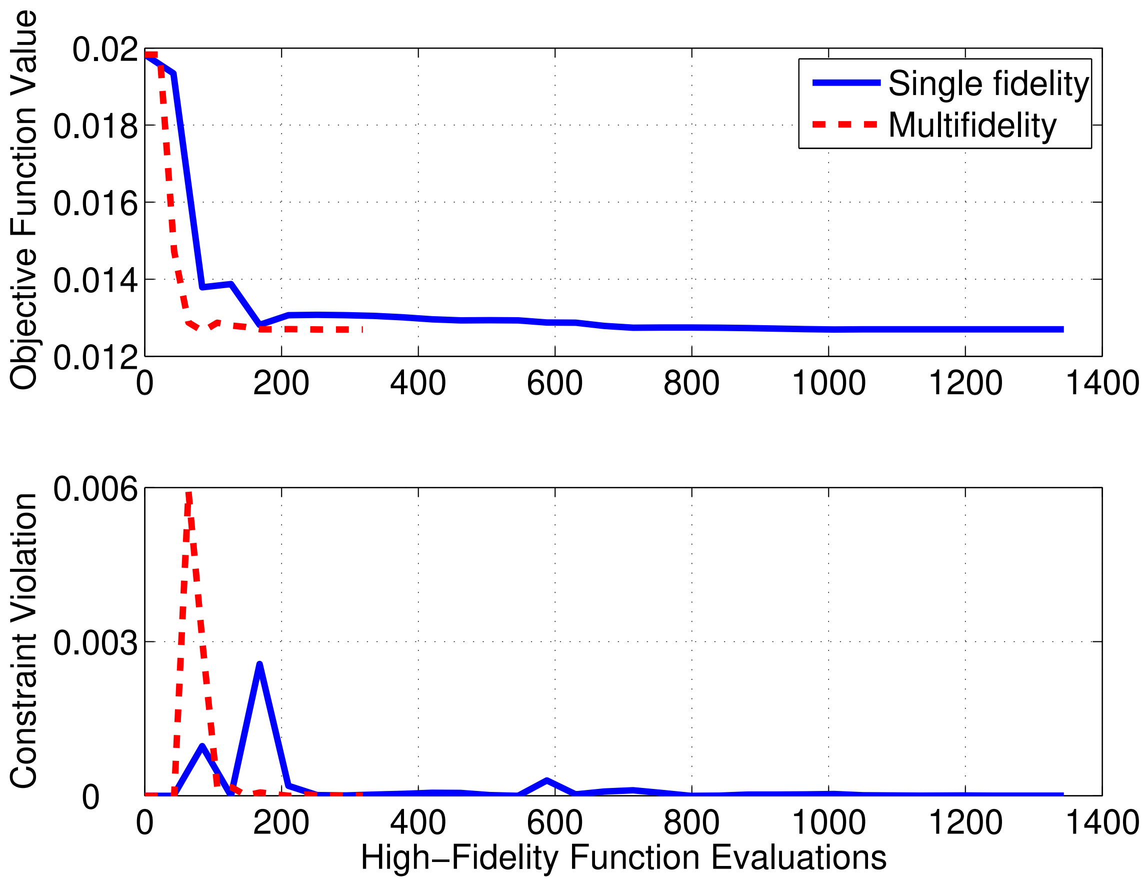

Figure 6-2: Objective function value and constraint violation of each of a single–fidelity SQP method and an SQP–like multifidelity method using corrected space mapping.

optimum design, with an objective within 10^{-5} of the best design found, with a constraint violation less than 10^{-6}. The multifidelity method found the optimum, using the same criteria, in 319 high–fidelity function calls. This is a savings of 76%. The computational time was reduced from approximately 34 hours to approximately 8 hours.

6.2 Flapping–flight problem

The final constrained design problem aims to explore biologically–inspired flapping flight. Heaving wings and airfoils are commonly used in nature as a form of force production and have gained the attention of many researchers, both to further un-

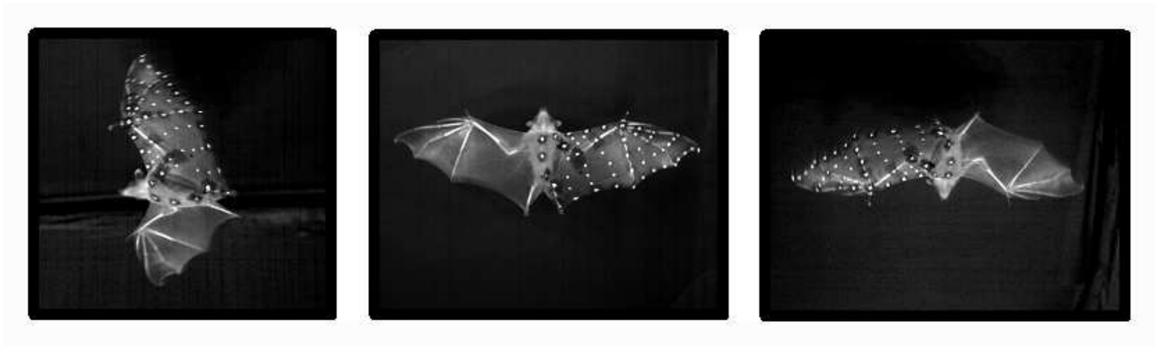
Figure 6-3: Marker locations on a *Cynopterus brachyotis*, or short–nosed fruit bat, outfitted for testing in the Harvard Concord Field Station Wind Tunnel. Figure courtesy of Kenneth Breuer and Sharon Swartz, Brown University.

derstand biological flight and to develop bioinspired aircraft [69, 53, 71, 41].

6.2.1 Problem Description

Researchers at Brown University study bats in a wind tunnel in order to understand the mechanics of bat flight. Figure 6-3 shows a bat outfitted with markers to track bat motion in flight in a wind tunnel. Researchers postulate that bats' physiological structure and flight motion are optimized for minimum power input at a specified forward velocity. In order to test that hypothesis, a model problem was formulated to investigate the minimization of power in order to generate thrust, using leading edge compliance. The goal of this problem is to examine passive strategies for minimizing the heaving motion power input for a given thrust output. It models bat flight using a heaving airfoil with a passive load–alleviation leading–edge torsional spring. While birds and bats in nature use more complex forms of both active and passive control, this optimization problem is a first step in the determination of the optimal passive strategy. A very similar problem, minimizing the power requirements for a swimming robot using passive springs, has been investigated using analytic approaches [42].

The design problem has eight design parameters. The first is a spring constant. This represents the structural compliance of the bat wing, modeled as a single leading–edge torsional spring providing passive load alleviation. The second design parameter is the flapping frequency. The next three are amplitudes and lags of each of three harmonics of flapping motion. Table 6.1 shows the eight design variables, their units, their lower and upper bounds, their start values, and their final values in each of the single–fidelity and multifidelity runs. The optimization problem is

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) = P_{in}(\mathbf{x}) \\ \text{subject to} \quad & c(\mathbf{x}) = (0.2 - C_T(\mathbf{x})) \leq 0 \\ & LB \leq x_j \leq UB, \quad j = 1 \ldots 8, \end{aligned} \tag{6.2}$$

where the input power P_{in}, the thrust coefficient C_T, and the elements of the design parameter vector $\mathbf{x} = (K, \omega, A_0, A_1, A_2, \phi_0, \phi_1, \phi_2)^T$ are as described in Table 6.1. The design variables were scaled to improve the numerical conditioning of the problem.

	Name	Unit	LB	UB	Initial	SF	MF
			Design Variables				
K	Spring Constant	$\frac{\text{N}\cdot\text{m}}{\text{rad}}$	.0001	.25	.005	.0136	.00949
ω	Flapping frequency	$\frac{\text{rad}}{\text{s}}$	10	120	20	11.4643	13.41
A_0	Amplitude of harmonic 0	m	0	.3	.2	.3	.3
A_1	Amplitude of harmonic 1	m	0	.15	0	7.93×10^{-5}	2.20×10^{-3}
A_2	Amplitude of harmonic 2	m	0	.15	0	4.82×10^{-3}	.0145
ϕ_0	Lag of harmonic 0	rad	$-\pi$	π	0	-3.058	-2.148
ϕ_1	Lag of harmonic 1	rad	$-\pi$	π	0	0	-.208
ϕ_2	Lag of harmonic 2	rad	$-\pi$	π	0	-1.943	.0626
			Constraint				
C_T	Coefficient of thrust	—	.2		.1076	.2	.2
			Objective Function				
P_{in}	Power input	W			.5461	.8966	.9049

Table 6.1: Design variables and results for flapping–wing problem. The columns are: The symbol for the variable, the name of the variable, its unit, the lower bound on the variable, the upper bound, the initial value, the final value found by the single–fidelity method, and the final value found by the multifidelity method.

6.2.2 High–fidelity analysis

The high–fidelity solver is a two–dimensional, unsteady, linear strength, source–doublet formulation [46, 52]. An advantage of using panel method approximations in an unsteady setting is that it requires neither remeshing nor moving body formulations (such as arbitrary Lagrange–Eulerian formulations of the Navier–Stokes equations). The unsteady forces and moments were computed by integrating the airfoil surface pressure, computed using the unsteady form of the Bernoulli equation [46]. In order to correct the results for viscous effects, a simple quasi–steady, drag–polar approximation was used. Additionally, a simple stall penalty scheme consisting of a quartic drag penalty on airfoil incidences over a specified value was also incorporated in order to ensure that the angle of incidence of the airfoil remained in the non–separated regime. Although the viscous model is not as rigorous as one which depends on the unsteady motion of the geometry (as would be the case for an Integral Boundary Layer method [25, 24, 29]), the incorporation of a simple viscous correction yields more realistic computations than an inviscid formulation. The passive structural load alleviation and airfoil rotation were accomplished by modeling the airfoil as a mass and a leading edge torsional spring. The following moment balance equation was enforced strongly for each timestep at the leading edge of the airfoil:

$$I\ddot{\theta} + K\theta + mx_{cg}\ddot{h} - M_{aero} = 0, \tag{6.3}$$

where I is the moment of inertia about the leading edge of the airfoil, θ is the angle of the wing, K is the spring constant of the torsional spring at the leading edge, m is the mass of the wing, x_{cg} is the x position of the center of gravity of the wing in the wake direction, h is the vertical position of the wing (and thus $\ddot{h}$ is its vertical acceleration), and M_{aero} is the moment due to aerodynamic forces.

A low–Reynolds–number HT–13 airfoil was used as the input geometry. The vertical heaving motion was described by a 3rd order series of harmonic functions as follows:

$$Z(t) = \sum_{m=o}^{2} A_m \cos(2\pi(m+1)\omega t + \phi_m), \tag{6.4}$$

where $Z(t)$ is the z–position, defined along a vertical axis, of the airfoil in time, t is time, A_i and ϕ_i, $i = 0 \ldots 2$, and ω are as defined in Table 6.1. The horizontal velocity is constant at

$$U(t) = U_\infty = 5m/s. \tag{6.5}$$

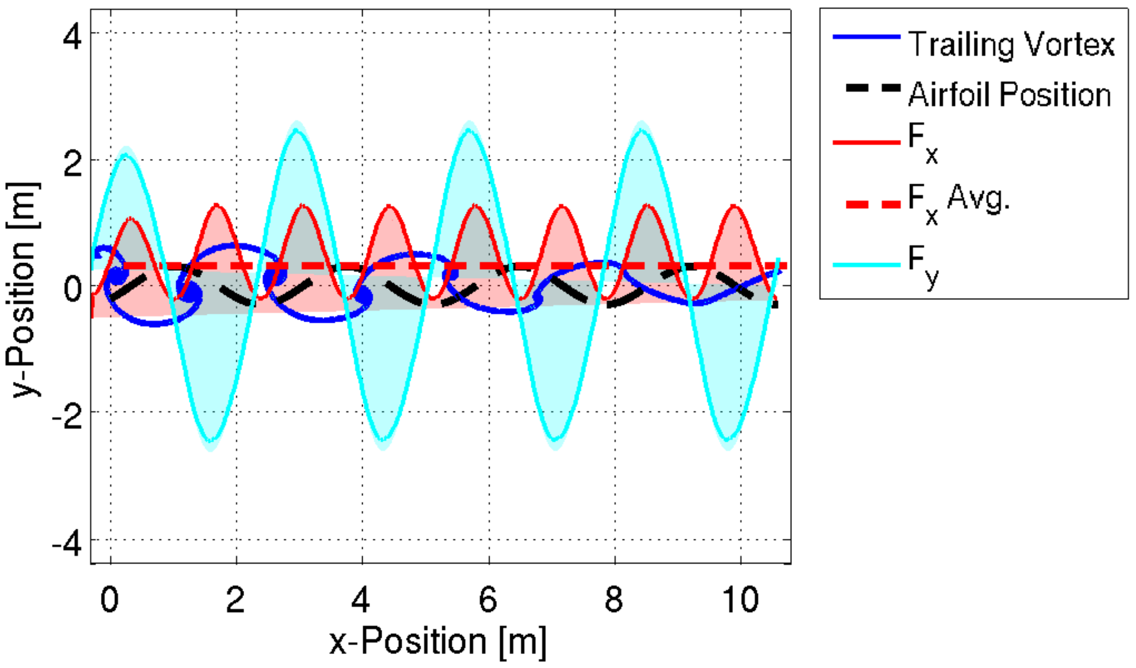

Figure 6-4: Results of the high–fidelity analysis of a flapping wing, showing the airfoil position, the trailing vortex, and the forces in the x and y directions.

The high–fidelity code requires approximately 30 seconds on a 2.4 GHz desktop workstation for a single evaluation.

6.2.3 Low–fidelity analysis

The low–fidelity analysis uses a simplified representation of the wing as a point airfoil. The calculations are quasi–steady. That is, the airfoil changes in position and angle of attack over time, but the aerodynamic equations are steady. The airfoil is assumed massless. The moment balance around the leading edge is therefore

$$K\theta + M_{c/4}(\alpha) - \frac{c}{4}L_{aero}(\alpha) = 0, \tag{6.6}$$

where $M_{c/4}$ is the aerodynamic moment about the quarter–chord, c is the chord, and L_{aero} is the aerodynamic lift. Using a thin airfoil theory approximation, the lift is

$$L_{aero} = 2\pi\alpha, \tag{6.7}$$

and the moment of the symmetric airfoil about the quarter–chord is

$$M_{c/4}(\alpha) = 0. \tag{6.8}$$

The angle of attack, α, can be approximated as the difference between the relative airflow direction (which can be determined from the forward and heaving velocities) and the angular deflection, θ of the airfoil. As a result, the spring deflection in equation 6.6 is the only remaining unknown, and is easily determined if the terms in the equation are approximated using harmonic representations. Once the spring deformations are known, the time–varying lift and thrust forces are computed using the magnitude of the force prediction and the angle of attack of the airfoil. In addition to the ideal aerodynamic forces, a simple quasi–steady drag polar viscous correction was implemented post solution. This was added to the model in order to ensure that adversely high angles of attack or velocities did not provide unrealistically high thrust values. The low–fidelity code takes three design variables: a spring constant, a flapping frequency, and a flapping amplitude. Only one flapping harmonic is included. Figure 6-5 shows the flapping motion of the wing and the resultant forces. The low–fidelity code requires approximately 20ms on a 2.4 GHz desktop workstation to evaluate a single design.

6.2.4 Results

Two methods were run: a single–fidelity SQP method as a benchmark, and the SQP–like multifidelity method with corrected space mapping. As in the previous problem, the gradients of the objectives and of the constraints were computed using centered finite differences. The function evaluations required for these finite–difference

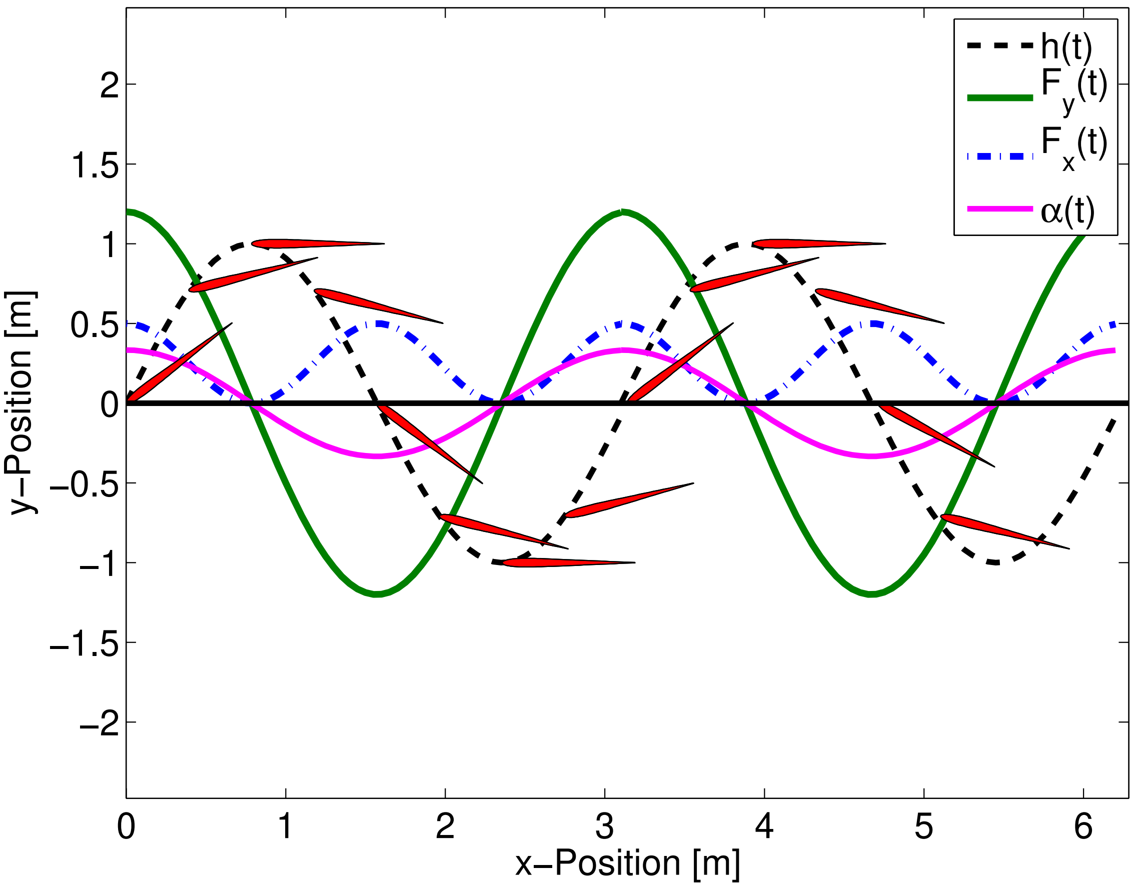

Figure 6-5: Results of the low–fidelity analysis of a flapping wing, showing the airfoil position, angle of attack, and resultant forces in the x and y directions.

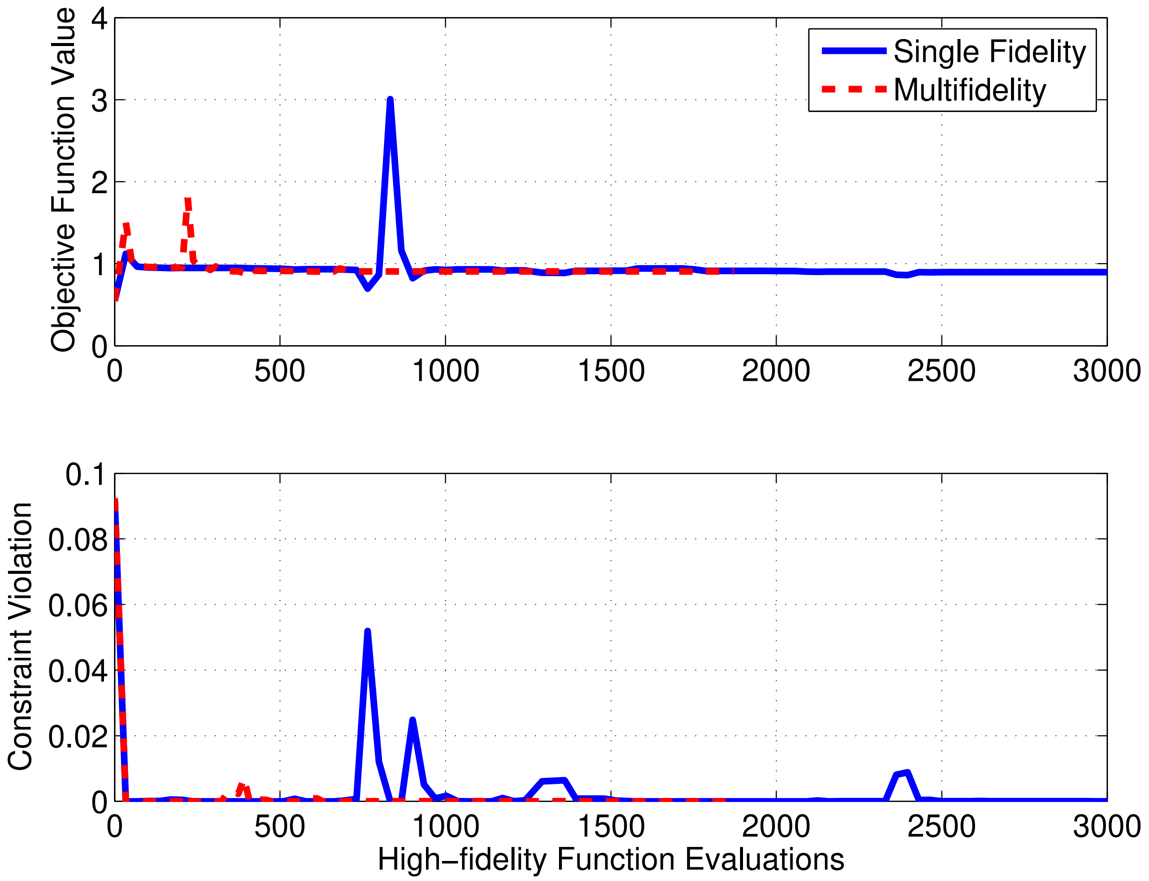

Figure 6-6: Objective function value and constraint violation for the flapping–wing problem, for the single–fidelity method and SQP–like multifidelity method with corrected space mapping.

calculations are included in the total function count. The methods converged to two different local minima. The Karush–Kuhn–Tucker condition was satisfied at both minima. This result highlights the point that the multifidelity method is guaranteed to converge to an optimum of the high–fidelity problem, but not necessarily the global optimum nor the same — possibly local — optimum found by the single–fidelity method. For both methods, the amplitude of the first harmonic (that is, the flapping motion at the base frequency) was set to its upper bound. No other design parameters were set to their bounds.

The flapping frequencies were similar: 11.46 rad/s for the single–fidelity method and 13.41 rad/s for the multifidelity method. These correspond to Strouhal numbers of 0.218 and 0.256 respectively. These are both within the 0.2 to 0.4 range found in birds, bats, and fish in nature [69, 68]. The multifidelity method found a design with an input power coefficient of 0.9049 while the single–fidelity method found a design with an input power coefficient of 0.8966. The difference between the two values is very small and below the predicted accuracy of the high–fidelity analysis.

The objective function and constraint violations are shown in Figure 6-6, with the number of high–fidelity function calls on the x–axis. Using the criterion that the constraint violation is less than 10^{-6} and the objective function is within 10^{-6} of its ultimate converged value, the single fidelity method required 3125 high–fidelity function calls and the multifidelity method required 1617. Since they did not converge to the same minimum, it is difficult to make a direct comparison between the numbers. The multifidelity method found a solution with a slightly higher objective function value with 48.2% less high–fidelity function evaluations. The multifidelity method required 77 hours to run while the single–fidelity method required 140 hours. This is a savings of approximately 45% in time. The difference in ratios between the function evaluation metric and the time metric are explained by the low–fidelity function calls used by the multifidelity method and the additional multifidelity algorithm overhead.

Chapter 7

Conclusion

7.1 Thesis Summary

This thesis developed and demonstrated new methods of reducing the computational cost of numerical optimization of a high–fidelity computational model when a less costly model exists and uses a different set of design variables. To that purpose, it extended multifidelity optimization methods to become capable of solving variable–parameterization — and specifically variable–dimensional — problems.

Chapter 2 presented existing TRMM methods for optimization, and expanded their applicability to variable–parameterization problems. It also presented a new TRMM method for optimization — including proof of convergence — that solves the trust–region subproblem in the low–fidelity space. Chapter 3 detailed the four methods used to perform mapping between variable–parameterization spaces: space mapping, corrected space mapping, POD mapping, and a hybrid space mapping/POD method. Chapter 4 contained the results of these new methods applied to two unconstrained problems — the Rosenbrock problem and an airfoil design problem — and compared the results to existing methods applied to the same problems. Chapter 5 presented detailed results of these new methods applied to the Barnes problem, including a comparison against existing methods. It also presented the results of varying the constraint–management method and a number of algorithm parameters. Chapter 6 showed the results of these new methods applied to two constrained design problems — the planform design of a wing and a flapping flight problem — and compares the results to existing methods applied to the same problems.

7.2 Conclusions

The new mapping methods shown in this thesis and the new variable–parameterization TRMM framework have accomplished their goal: computational savings for variable–parameterization problems. The TRMM framework has similar computational savings using either corrected space mapping or POD mapping. Corrected space mapping is more straightforward to implement, since it does not require the generation of training pairs required by POD mapping. Conversely, POD mapping has lower algorithm overhead. Therefore, if training pairs are available or trivial to generate, POD mapping is preferred. Otherwise, corrected space mapping is more appropriate. In very large–scale problems, with large numbers of design variables, either POD mapping or the hybrid POD/space mapping method should be used, since the algorithm overhead required by corrected space mapping may negate the computational savings due to fewer high–fidelity function evaluations.

With regard to limitations, a high–fidelity function evaluation is required at each trust–region iteration, providing a lower bound on computational costs. Designers can improve the performance of the method by using experience and judgment to choose an appropriate initial trust–region radius. This work found that variable–parameterization problems depend on the TRMM parameters in a similar manner to other variable–fidelity problems, and that therefore the recommendations of Conn et al. [22] are appropriate. Further, the methods discussed in this work are gradient–based methods. The computation of gradients may be impossible or expensive in some problems. Alternatives are gradient–free methods such as pattern–search methods.

A number of large, significant variable–parameterization problems exist. One example is that considered in the introduction to this thesis: the supersonic business jet studied by Choi et al. [21]. In this example, there exists a significantly less expensive low–fidelity model using a different set of design variables from those used by the high–fidelity model. In addition, adjoint–based gradient calculations have been implemented on the problem, making high–fidelity gradients available at approximately the cost of a high–fidelity function evaluation. On this and similar problems, the methods presented in this work should yield significant computational savings.

Moreover, the mapping methods developed herein have broader uses in design, outside of formal optimization methods. Working with various analysis models and parameterizations is an integral part of the design process, and this mapping methodology provides a systematic way to achieve links between the models computationally.

7.3 Contributions

This thesis contains the following contributions:

1. Developed new techniques for mapping between analysis models with different design parameterizations.

2. Presented a new, provably–convergent optimization framework for multifidelity models with variable parameterizations for both unconstrained and constrained optimization problems.

3. Compared these new mapping methods and optimization techniques with one another and existing techniques.

4. Demonstrated these new mapping methods and optimization techniques on engineering design problems within the context of aerospace engineering.

For unconstrained problems, the new mapping methods combined with TRMM achieved consistent computational savings, as measured by high–fidelity function calls. On an airfoil design problem, TRMM with POD mapping achieved 40% savings while TRMM with corrected space mapping achieved 12% savings. On the Barnes problem, a two–dimensional analytic constrained problem, the SQP–like and direct surrogate optimization methods achieved consistent computational savings, while the augmented Lagrangian methods and MAESTRO did not. On a constrained wing design problem, TRMM with corrected space mapping achieved 76% savings. Finally, on a bat flight design problem, it achieved approximately 45% time savings, although it converged to a different local minimum than the benchmark did.

7.4 Future Work

One avenue of interest is using more than two models, each of different fidelity, within this framework. While much of the literature outlines that the trust–region method is infinitely nestable and applicable to any number of models [6], there is little published work with more than two models. A first step would be to extend this work to a three–model hierarchy.

A related concept is that of adaptive–fidelity methods. In this method, instead of — or in addition to — changing the size of the trust region as in TRMM, the fidelity of the model could be changed depending on the model's predictive performance. The use of variable–parameterization models, in which the number of design variables change as needed, is one way to nearly infinitely vary the fidelity of an existing model. For instance, the design variables for a wing design could be the positions of points on the surface of the wing. When needed, the number of points, and therefore the number of design variables, would be increased. The mapping methods demonstrated in this thesis would then be used to link the resulting variable–parameterization models. This would borrow some ideas from the multigrid method of computational fluid mechanics.

A further area of interest is the exploitation of the structure of multidisciplinary problems. In some cases, there exist separate low–fidelity models of the same system in two disciplines and one high–fidelity model that couples the disciplines. For example, there may be a high–fidelity model that calculates coupled aerodynamic forces and structural displacements and two low–fidelity models that apply each discipline independently. Variable–fidelity models could be developed to exploit the structure of these problems in order to accelerate optimization of the coupled problem. One proposal is to use separate trust–regions for the low–fidelity models of each of the disciplines, based on the performance of that model.

The POD method described in this thesis has one significant limitation: it requires the manual generation of training pairs by the designer. It would be preferable if the training pairs could be automatically generated. One method for achieving automatic training–pair generation is the application of space–mapping–like parameter extraction to the models. This requires the solution of an optimization problem for the generation of each training pair.

The POD method is currently static. That is, it uses one mapping — based on one set of training pairs — throughout the optimization process. However, it is possible that the original mapping becomes less useful as the optimization progresses. If the automatic generation of training pairs described above is achieved, the TRMM algorithm could be modified to recognize when the mapping is becoming poor, and generate a new mapping. It would then use new training pairs within the local region of the current iterate, making the mapping locally accurate. The trust–region size is a potential indicator of the need to create a new mapping, and thus a new set of training pairs.

Bibliography

[1] AIAA MDO Technical Committee. AIAA White Paper, 1991.

[2] N. Alexandrov, J.E. Dennis, R.M. Lewis, and V. Torzon. A trust region framework for managing the use of approximation models in optimization. NASA CR–201735, 1997.

[3] N. M. Alexandrov. Multilevel methods for MDO. In N. M. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State of the Art*, pages 79–89. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[4] N. M. Alexandrov. On managing the use of surrogates in general nonlinear optimization, September 1998. AIAA Paper 98–4798.

[5] N. M. Alexandrov, E.J. Nielsen, R.M. Lewis, and W.K. Anderson. First–order model management with variable–fidelity physics applied to multi–element airfoil optimization. In *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 2000. AIAA Paper 2000–4886.

[6] N.M. Alexandrov, J.E. Dennis, R.M. Lewis, and V. Torczon. A trust–region framework for managing the use of approximation models in optimization. *Structural and Multidisciplinary Optimization*, 15(1):16–23, February 1998.

[7] N.M. Alexandrov, R.M. Lewis, C.R. Gumbert, L.L. Green, and P.A. Newman. Optimization with variable–fidelity models applied to wing design. In *Proceedings*

of the 38th Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2000. AIAA 2000–0841.

[8] J.D. Anderson. *Fundamentals of Aerodynamics, 2nd ed.* McGraw–Hill, 1991.

[9] H. Ashley and M. Landahl. *Aerodynamics of Wings and Bodies.* Dover Publications, New York, 1985.

[10] C. Audet and J.E. Dennis. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal of Optimization*, 14(4):980–1010, 2004.

[11] C. Audet, J.E. Dennis, D. W. Moore, A. Booker, and P. D. Frank. A surrogate–model–based method for constrained optimization. In *Proceedings of the 8th AIAA/USAF/NASA/ASSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, Sept. 6–8 2000. AIAA Paper 2000–4891.

[12] M. H. Bakr, J. W. Bandler, N. Georgieva, and K. Madsen. A hybrid aggressive space mapping algorithm for EM optimization. *IEEE MTT–S International Microwave Symposium Digest*, 1:265–268, 1999.

[13] J.W. Bandler, R.M. Biernacki, and S.H. Chen. Fully automated space mapping optimization of 3D structures. In *Proceedings of the IEEE MTT–S International Microwave Symposium*, pages 753–756, San Francisco, CA, June 1996.

[14] J.W. Bandler, R.M. Biernacki, S.H. Chen, P.A. Grobelny, and R.H. Hemmers. Space mapping technique for electromagnetic optimization. *IEEE Transactions on Microwave Theory and Techniques*, 42:2536–2544, 1994.

[15] J.W. Bandler, R.M. Biernacki, S.H. Chen, R.H. Hemmers, and K. Madsen. Electromagnetic optimization exploiting aggressive space mapping. *IEEE Transactions on Microwave Theory and Techniques*, 43:2874–2882, 1995.

[16] G.K. Barnes. Master's thesis, The University of Texas, Austin, Texas, 1967.

[17] I. Bongartz, A. R. Conn, N. I. M. Gould, and P. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.

[18] A.J. Booker, J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17:1–13, 1999.

[19] C.G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematical Computation*, 19:577–593, 1965.

[20] T. Bui-Thanh, M. Damodaran, and K. Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA Journal*, 42(8):1505–1516, August 2004.

[21] S. Choi, J. Alonso, S. Kim, and I. Kroo. Two–level multi–fidelity design optimization studies for supersonic jets. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 2005. AIAA Paper 2005–531.

[22] A. R. Conn, N. I.M. Gould, and P. L. Toint. *Trust–Region Methods*. MPS/SIAM Series on Optimization. Society for Interactive and Applied Mathematics, Philadelphia, 2000.

[23] J. E. Dennis. A brief introduction to quasi–Newton methods. In *Numerical Analysis, Proceedings of Symposia in Applied Mathematics 22*, Providence, RI, USA, 1978. American Mathematical Society.

[24] M. Drela. XFOIL: An analysis and design system for low Reynolds number airfoils. In T.J. Mueller, editor, *Low Reynolds number aerodynamics : proceedings of the conference, Notre Dame, Indiana*. Springer–Verlag, June 1989.

[25] M. Drela and M.B. Giles. Viscous–inviscid analysis of transonic and low Reynolds number airfoils. *AIAA Journal*, 25(10):1347–1355, October 1987.

[26] M. El-Alem. A global convergence theory for Dennis, El–Alem, and Maciel's class of trust–region algorithms for constrained optimization without assuming regularity. *Siam Journal on Optimization*, 9(4):965–990, 1999.

[27] M. Eldred, S. Giunta, and S. Collis. Second–order corrections for surrogate–based optimization with model hierarchies. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary analysis and optimization conference*, Albany, New York, Aug 30. – Sept. 1 2004. AIAA.

[28] M. S. Eldred and D. M. Dunlavy. Formulations for surrogate–based optimization with data fit, multifidelity, and reduced–order models. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, VA, September 2006. AIAA paper 2006–7000.

[29] R. Eppler and D.M. Somers. Low speed airfoil design and analysis, advanced technology airfoil research – volume I. NASA CP 2045, 1979.

[30] R. Everson and L. Sirovich. The Karhunen–Loève Procedure for Gappy Data. *Journal of the Optical Society of America*, 12(8):1657–1664, 1995.

[31] R. Fletcher, S. Leyffer, and P. L. Toint. On the global convergence of a filter–SQP algorithm. *SIAM Journal of Optimization*, 13(1):44–50, 2002.

[32] R. Fletcher, S. Leyffer, and Ph. L. Toint. On the global convergence of an SLP–filter algorithm. Technical Report 98/13, 1998.

[33] A. I.J. Forrester, N. W. Bressloff, and A. J. Keane. Optimization using surrogate models and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society A*, 462:2177–2204, March 2000.

[34] J. Giesing and J. Barthelemy. A summary of industry MDO applications and needs. AIAA White Paper, 1998.

[35] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison–Wesley, Reading, Massachusetts, 1989.

[36] D. Goldfarb. A family of variable metric methods derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[37] S. Gratton, A. Sartnaer, and Ph. L. Toint. Recursive trust–region methods for multiscale nonlinear optimization (part1): Global convergence and complexity. Technical report, FUNDP, Namur (B), 2004.

[38] B. Grossman, R.T. Haftka, P.-J. Hao, D. Polen, M. Rais-Rohani, and J. Sobieszczanski-Sobieski. Integrated aerodynamic and structural design of a transport wing. *Journal of Aircraft*, 27:1050–1056, 1990.

[39] A. Grothey and K. McKinnon. A superlinearly convergent trust region bundle method. Technical report, Department of Mathematics and Statistics, University of Edinburgh, December 1998.

[40] R.T. Haftka, B. Grossman, W.M. Eppard, P.J. Kao, and D. Polen. Efficient optimization of integrated aerodynamic–structural design. *International Journal of Numerical Methods in Engineering*, 28:593–607, 1989.

[41] K.C. Hall, S.A. Pigott, and S.R. Hall. Power requirements for large–amplitude flapping flight. *Journal of Aircraft*, 35(3):352–361, 1998.

[42] K.A. Harper, M.D. Berkemeier, and S.M. Grace. Decreasing the energy costs of swimming robots through passive elastic elements. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997.

[43] R.M. Hicks and P.A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, July 1978.

[44] D.M Himmelblau. *Applied Nonlinear Programming.* McGraw Hill, 1972.

[45] P. J. Holmes, J. L. Lumley, G. Berkooz, J. C. Mattingly, and R. W. Wittenberg. Low–dimensional models of coherent structures in turbulence. *Physics Reports*, 287(4):337–384, 1997.

[46] J. Katz and A. Plotkin. *Low–Speed Aerodynamics*. Cambridge University Press, Cambridge, second edition, 2001.

[47] S.A. Kazarlis, S.E. Papadakis, and J.B. Theocharis. Microgenetic algorithms as generalized hill–climbing operators for GA optimization. *IEEE Transactions on Evolutionary Computation*, 5(3):204–217, 2001.

[48] I. Kroo, S. Altus, R. Braun, P. Gage, and J. Sobieszczanski-Sobieski. Multidisciplinary optimization methods for aircraft preliminary design. In *Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, volume 1, pages 697–707, Panama City Beach, FL, 1994. AIAA Paper 94–4325.

[49] C.L. Lawson and R.J. Hanson. *Solving Least–Squares Problems*, page 161. Prentice–Hall, 1974.

[50] R. M. Lewis. A trust region framework for managing approximation models in engineering optimization. AIAA paper 96–4101, presented at the Sixth AIAA/NASA/ISSMO Symposium on Multidisplinary Analysis and Design, Bellevue, Washington, 1996.

[51] K. Madsen and J. Søndergaard. Convergence of hybrid space mapping algorithms. *Optimization and Engineering*, 5:145–156, 2004.

[52] L. Morino and C.C. Kuo. Subsonic potential aerodynamics for complex configurations: A general theory. *AIAA Journal*, 12(2):191–197, 1974.

[53] T.J. Mueller, editor. *Fixed and Flapping Wing Aerodynamics for Micro Air Vehicles*. AIAA Progress in Aeronautics and Astronautics, Reston, VA, 2001.

[54] V. Perez, M. Eldred, and J. Renaud. Solving the infeasible trust–region problem using approximations. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, August 2004. AIAA Paper 2004–4312.

[55] V.M. Perez, J. E. Renaud, and L. T. Watson. Adaptive experimental design for construction of response surface approximations. In *Proceedings of the 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Sttructural Dynamics, and Materials Conference and Exhibit*, Seattle, WA, April 16–19 2001. AIAA=2001–1622.

[56] V.M. Perez, J.E. Renaud, and L.T. Watson. Homotopy curve tracking in approximate interior point optimization. In *Proceedings of the 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Norfolk, VA, April 2003. AIAA–2003–1670.

[57] M. J. D. Powell. A new algorithm for unconstrained optimization. pages 31–65.

[58] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker. Surrogate–based analysis and optimization. *Progress in Aerospace Sciences*, 41(1):1–28, January 2005.

[59] J.F. Rodriguez, J.E. Renaud, and L.T. Watson. Convergence of trust region augmented Lagrangian methods using variable fidelity approximation data. *Structural Optimization*, 15:121–156, 1998.

[60] J.F. Rodriguez, J.E. Renaud, and L.T. Watson. Trust region augmented Lagrangian methods for sequential response surface approximation and optimization. *Journal of Mechanical Design*, 120(1):58–66, March 1998.

[61] H.H. Rosenbrock. An automatic method for finding the greatest or least value if a function. *Computer Journal*, 3:175–184, 1960.

[62] S. J. Sadjadi and K. Ponnambalam. Advances in trust region algorithms for constrained optimization. *Applied Numerical Mathematics*, 29(3):423–443, 1999.

[63] Y.-W. Shang and Y.-H. Qiu. A note on the extended Rosenbrock function. *Evolutionary Computation*, 14(1):119–126, 2006.

[64] D.F. Shanno. Conditioning of quasi–Newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970.

[65] L. Sirovich. Turbulence and the Dynamics of Coherent Structures. Part 1 : Coherent Structures. *Quarterly of Applied Mathematics*, 45(3):561–571, October 1987.

[66] J. Sobieszczanski-Sobieski and I. Kroo. Aircraft design using collaborative optimization. AIAA Paper 96–0715, 1996.

[67] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[68] G.K. Taylor, R.L. Nudda, and A.L.R. Thomas. Flying and swimming animals cruise at a Strouhal number tuned for high power efficiency. *Nature*, pages 707–711, October 2003.

[69] M.S. Triantafyllou, G.S. Triantafyllou, and R. Gopalkrishnan. Wake mechanics for thrust generation in oscillating foils. *Physics of Fluids A*, 3(12):2835–2837, 1991.

[70] G. Venter, R.T. Haftka, and Jr. J.H. Starnes. Construction of response surface approximations for design optimization. *AIAA Journal*, 36(12):2242–2249, December 1998.

[71] Z.J. Wang. Vortex shedding and optimal flapping flight. *Journal of Fluid Mechanics*, 410(323), 2000.

[72] X. Yao and Y. Liu. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999.

[73] Y. Yuan. A review of trust region algorithms for optimization. Technical Report ICM-99-038, 1999.