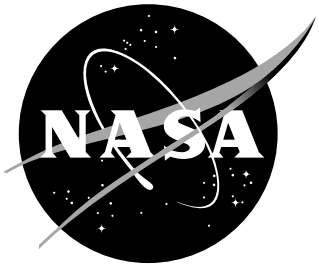
NASA/CR–2010–216874



Multifidelity Analysis and Optimization for Supersonic Design

Ilan Kroo
Stanford University, Stanford, California

Karen Willcox and Andrew March
Massachusetts Institute of Technology, Cambridge, Massachusetts

Alex Haas and Dev Rajnarayan
Stanford University, Stanford, California

Cory Kays
Massachusetts Institute of Technology, Cambridge, Massachusetts

December 2010

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

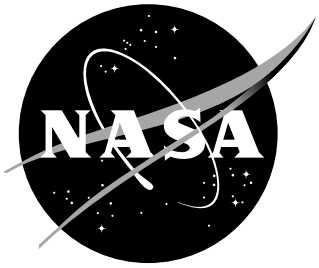
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***
- E-mail your question via the Internet to help@sti.nasa.gov
- Fax your question to the NASA STI Help Desk at (443) 757–5803
- Phone the NASA STI Help Desk at (443) 757–5802
- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/CR–2010–216874



Multifidelity Analysis and Optimization for Supersonic Design

Ilan Kroo
Stanford University, Stanford, California

Karen Willcox and Andrew March
Massachusetts Institute of Technology, Cambridge, Massachusetts

Alex Haas and Dev Rajnarayan
Stanford University, Stanford, California

Cory Kays
Massachusetts Institute of Technology, Cambridge, Massachusetts

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

December 2010

Available from:

NASA Center for AeroSpace Information (CASI)
7115 Standard Drive
Hanover, MD 21076–1320
(443) 757–5802

Abstract

Supersonic aircraft design is a computationally expensive optimization problem and multifidelity approaches offer a significant opportunity to reduce design time and computational cost. This report presents tools devloped to improve supersonic aircraft design capabilities including: aerodynamic tools for supersonic aircraft configurations; a systematic way to manage model uncertainty; and multifidelity model management concepts that incorporate uncertainty. The aerodynamic analysis tools developed are appropriate for use in a mutltifidelity optimization framework, and include four analysis routines to estimate the lift and drag of a supersonic airfoil, a multifidelity supersonic drag code that estimates the drag of aircraft configurations with three different methods: an area rule method, a panel method, and an Euler solver. In addition, five multifidelity optimization methods are developed, which include local and global methods as well as gradient-based and gradient-free techniques.

Contents

1 Overview

The following report (deliverable number 15c) presents work done at Stanford University and MIT on multifidelity design strategies for supersonic aircraft design as part of the NRA for Research Opportunities in Aeronautics (NNL07AA33C). The project's goals were the development of tools that can be used for the analysis and design of supersonic aircraft configurations; a systematic way to manage model uncertainty; and development of multifidelity model management concepts that incorporate uncertainty.

A number of aerodynamic analysis tools appropriate for use in a mutltifidelity optimization framework were developed as part of this project. They are presented in Chapter 2. Sec. 2.1 details four analysis routines that can be used to estimate the lift and drag of a supersonic airfoil. Sec. 2.2 describes a multifidelity supersonic drag code that estimates the drag of aircraft configurations with three different methods: an area rule method, a panel method, and an Euler solver. A comparison of the drag predictions made by these methods on selected wing-body configurations is also presented. These drag prediction methods were then integrated with PASS SE to create a mission analysis method with low- and high-fidelity estimates of inviscid cruise drag (Sec. 2.3). In addition, an architecture to use a ModelCenter project as a multifidelity optimization problem has been developed. Sec. 2.4 will discuss the architecture and present a sample project.

Five multifidelity optimization methods were developed within this project, including local and global methods as well as gradient-based and gradient-free methods. The first method, Chapter 3, stems from the underlying principles of SQP methods, which are efficient single-fidelity optimization algorithms. These methods solve a subproblem in which the real function is approximated by a quadratic and a line search is then performed in the direction of the quadratics optimum. We propose the use of a corrected low-fidelity model rather than the purely quadratic model in the subproblem, with the subsequent line search in the direction of the point it identifies.

Chapter 5 describes a two-fidelity gradient-free method that is based on the expected improvement [see 1, 2] algorithm. This method uses Gaussian Process (GP) regression [3] (also known as Kriging[4]) to model the difference between low- and high-fidelity functions. This algorithm combines both uncertainty and the mean performance from the Kriging model into a single metric which is then used to determine new points for evaluation with the high-fidelity analysis method. The method presented in Chapter 6 keeps uncertainty and mean performance separate and uses a multiobjective search to find designs that should be evaluated using the high-fidelity analysis.

Chapter 9 describes a multifidelity optimization algorithm that combines ideas from multifidelity expected improvement methods and SQP-based methods. The algorithm uses Bayesian model calibration and a Gaussian Process to model the error between a high- and low-fidelity function. The algorithm never computes gradients of the high-fidelity function; however it demonstrates convergence to a high-fidelity optimum using sensitivity information from the calibrated low-fidelity function which are constructed to have negligible error in the neighborhood around the optimum. The algorithm will be extended to address general nonlinearly constrained multifidelity optimization problems in Chapter 10. In this chapter a similar calibration strategy will be used for both the objective function and constraints in order to demonstrate convergence to a point where the Karush-Kuhn-Tucker (KKT) conditions are satisfied.

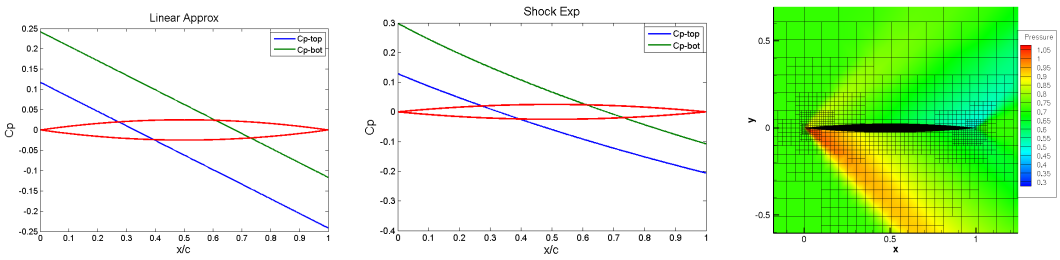
In addition to discussing the various multifidelity optimization techniques, a number of supersonic design problems are also presented. These problems include the design of a body

of revolution with minimum wave drag (Chapter 7), the minimization of drag for a wing-body configuration at fixed C_L (Chapter 8), the design of a supersonic aircraft (Chapter 4), the minimization of drag for a supersonic airfoil (Secs. 9.5.2 and 10.4), and the maximization of L/D with a upper bound on drag (Sec. 10.4).

2 Multifidelity Analysis Methods

2.1 Supersonic Airfoil Analysis Methods

Three supersonic airfoil analysis methods were developed to enable testing of multifidelity optimization methods that include multiple lower-fidelity estimates to optimize a single high-fidelity analysis result. The three methods are a supersonic linear panel method, a shock-expansion theory panel method, and an Euler solver, Cart3D [5, 6]. Figure 2.1 shows the approximate level of detail used in the models, and Table 2.1 compares the lift and drag estimates from each of the models for a 5% thick biconvex airfoil at Mach 1.5 and 2° angle of attack. The linear panel method and shock-expansion theory both require sharp leading and trailing edges which will be enforced by construction for all of the airfoil designs considered. In addition to these three methods, the linear panel method will also be used on the camberline of an airfoil which is akin to subsonic thin airfoil theory. The camberline analysis will in general give a poor result for supersonic airfoil drag as the trends estimated using only the airfoil camberline can be significantly different from the trends using a method considering thickness. A specific example of when the camberline method produces a poor result is a symmetric airfoil at 0° angle of attack which has no drag considering only the camberline, but which may have considerable thickness drag that will be estimated using any of the higher-fidelity models.



(a) Panel method. (b) Shock-expansion. (c) Cart3D.

Figure 2.1. Supersonic airfoil model comparisons at Mach 1.5 and 2° angle of attack.

	Panel	Shock-Expansion	Cart3D
C_L	0.1244	0.1278	0.1250
% Diff	0.46%	2.26%	0.00%
C_D	0.0164	0.0167	0.01666
% Diff	1.56%	0.24%	0.00%

Table 2.1. 5% thick biconvex airfoil results comparison at Mach 1.5 and 2° angle of attack. Percent difference is taken with respect to the Cart3D results.

2.2 Multifidelity Wave Drag Code

Three different methods have been integrated to create a multifidelity wave drag routine. The lowest fidelity code is based on the linear area rule method. The middle fidelity code is A502/Panair[7], a high-order surface panel method, and the high fidelity code is CART3D, an Euler solver. These methods are all capable of analyzing wing, body, tail, and nacelle geometries. Nacelles are currently modeled as parabolic bodies of revolution with a maximum cross section area equal to the maximum area of the nacelle minus the streamtube capture area. In CART3D, however, flow-through nacelles can also be analyzed.

The input to the multifidelity wave drag code is an xml file containing the parameterization of the aircraft, described below in Sec. 2.2.1 . A schematic of the program is shown in Fig. 2.2.

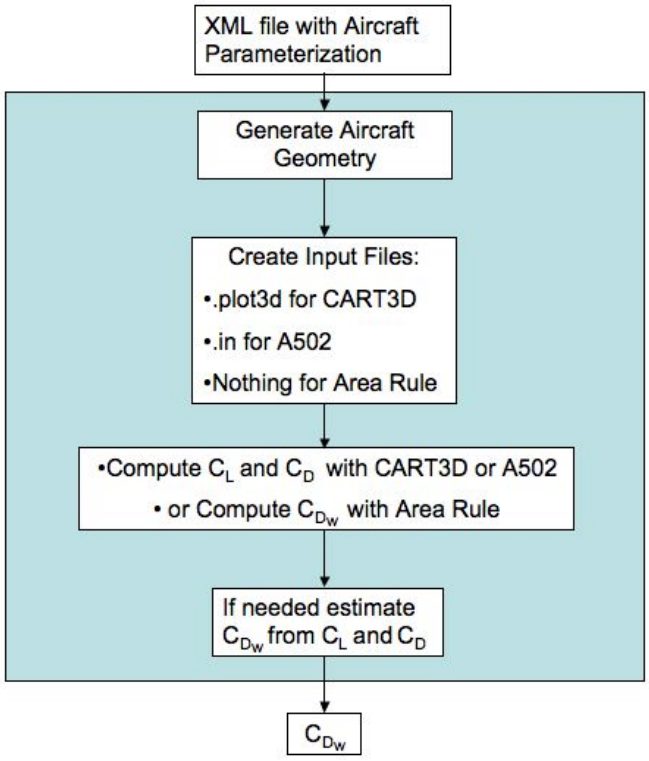


Figure 2.2. Schematic of the multifidelity wave drag code.

Because A502 and CART3D compute the total inviscid drag the wave drag component is estimated by subtracting the vortex drag of an elliptically loaded wing with the same lift and span.

2.2.1 Aircraft Parameterization

For initial tests of the multifidelity optimization tools, aircraft are defined using a set of parameters appropriate for conceptual design. The following tables and figures describe those parameters related to the geometry of the fuselage, wing, body, tail, and nacelles.

Wing

The main wing consists of two trapezoidal elements with section geometry assumed to vary linearly from the root to the tip of each element. Table. 2.2 defines the main wing parameters.

S_{ref}	Trapazoidal reference area (excluding chord extensions)
AR	Aspect ratio of reference trapezoidal planform, b^2/S_{ref}
$\Lambda_{c/4}$	Quarter chord sweep of trapezoidal wing
λ	Trap wing taper ratio, c_{tip}/c_{root}
lex	Leading edge extension as fraction of trap root chord
tex	Trailing edge extension as a fraction of trap root chord
chordextspan	Chord extensions span fraction
ϕ	Wing dihedral
x/L	x location of wing root leading edge as a fraction of fuselage length
z/L	z location of wing root leading edge as a fraction of fuselage length

Table 2.2. Wing parameterization.

Fig. 2.3 illustrates some of these parameters for a generic supersonic wing.

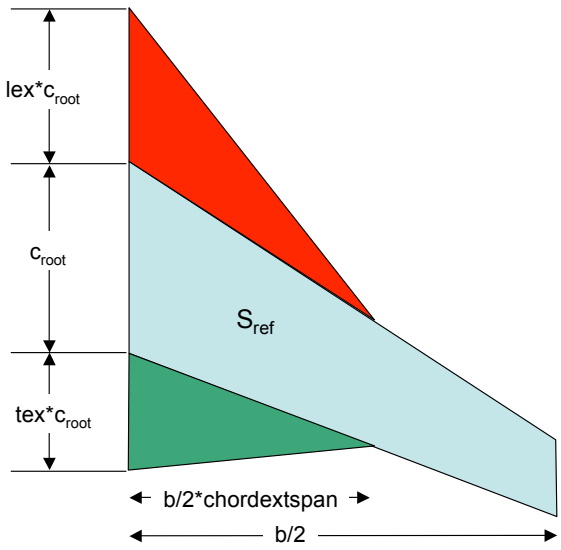


Figure 2.3. Wing geometric parameters.

Tail Geometry

The horizontal and vertical tail geometry parameterization is similar to that of the wing, but the planform is assumed to be made of a single trapezoid.

S_h/S_{ref}	Ratio of horizontal tail area to S_{ref}
AR_h	Horizontal tail aspect ratio
Λ_h	Quarter chord sweep
λ_h	Taper ratio
ϕ_h	Dihedral
(z_h/b_v)	Height of horizontal tail root as a fraction of vertical tail height
S_v/S_{ref}	Ratio of vertical tail area to S_{ref}
AR_v	Vertical tail aspect ratio, $height^2/S_v$
Λ_v	Quarter chord sweep
λ_v	Taper ratio
x_v/L	Location of vertical tail root trailing edge as a fraction of fuselage length.

Table 2.3. Horizontal and vertical tail parameterization.

The wing is defined by three airfoil sections at the root, the chord break, and the tip. The tails have the same airfoil section (one for the vertical tail and one for the horizontal tail) for the root and tip. Airfoil sections may be NACA 4 series, NACA65 series or biconvex. Airfoils sections are parameterized in the same way regardless of the airfoil section being used. The parameterization is given in Table. 2.4.

(t/c)	Thickness to chord ratio
(x_{maxcam}/c)	Location of maximum camber as fraction of chord
$camber_{max}/c$	Max camber as fraction of chord
θ	Incidence of the section

Table 2.4. Airfoil parameterization.

Fuselage

The fuselage is parameterized by specifying the radius, height-to-width ratio, and camber at a number of stations. Given these distributions an Akima spline[8] is fit to each one fully define the fuselage width and camber. The magnitude of r_h/r_w is assumed to be constant over the length of the fuselage.

Table 2.5. Fuselage parameterization.

L	length of the fuselage
r_w/L	Fuselage width as fraction of fuselage length
z_{camber}/L	Height of fuselage centerline as fraction of fuselage length
r_h/r_w	Ratio of fuselage height to width

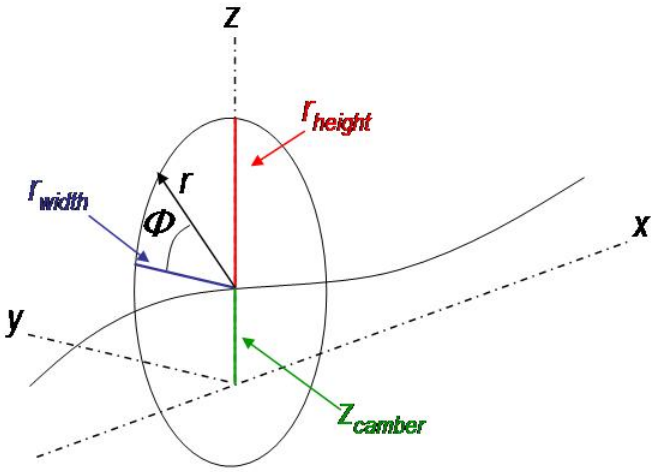


Figure 2.4. Fuselage geometry parameterization.

Nacelle

For drag purposes, in the area rule method and A502, nacelles are modeled as parabolic bodies of revolution with a maximum cross-section equal to the the maximum cross-section of the nacelle minus the stream tube area. A flow-through nacelle is modeled with an outer surface defined by a quadratic such that the maximum diameter is D_{max}. The inner surface is a cylinder with cross-sectional area A_o.

Table 2.6. Nacelle parameterization.

$L_{nacelle}$	Nacelle length
D_{max}	Maximum diameter of nacelle
A_o	Stream tube area
x_n/L	x location of nacelle inlet as fraction of fuselage length
y_n/L	y location of nacelle center line as fraction of fuselage length
z_n/L	z location of nacelle center line as fraction of fuselage length

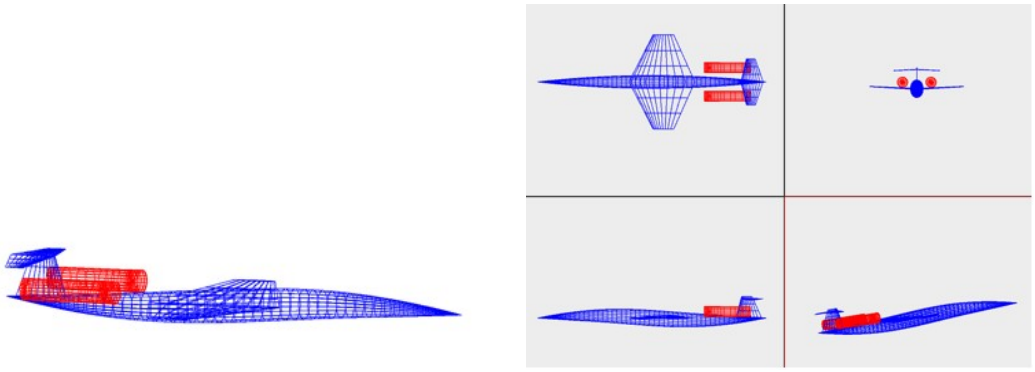
2.2.2 Integration of Vehicle Sketch Pad with the Multifidelity Wave Drag Code

A Vehicle Sketch Pad (VSP) interface has been developed to visualize the geometries created with the current codes. For testing purposes, a VSP model is exported by adding the following line to an xml input file.

```
<var name="VSPfilename"><val>vsptest</val></var>
```

This will result in a VSP model with the name "vsptest.xml" containing the fuselage, wing, tails, and nacelles. The airfoils imported to VSP are the airfoils that are generated by

the geometry routines in the multifidelity wave drag code. A sample supersonic aircraft configuration is shown in Fig. 2.5.



(a). Side view of VSP model. (b). Four view of VSP model.

Figure 2.5. VSP model generated by multifidelity wave drag code.

2.2.3 Area Rule Method

The area rule method (see reference [9] for a good discussion of area rule methods) for this project was developed to analyze configurations that correspond to the parameterization described in Sec. 2.2.1. We also assume the lift distribution of the main wing to be elliptic over the span, which is a valid assumption for the low-aspect-ratio wings found on most supersonic aircraft. Δ_P is assumed to be constant over the chord. Other Δ_P distributions, if desired, can be incorporated into the code with very minor modifications. Currently it is assumed that the tail carries no lift.

In order to avoid double-counting the wing area inside the fuselage, we model the wing-fuselage intersection as a straight line. In other words, the fuselage remains the same, but we assume the wing root to be a straight line between the intersection of the wing's leading and trailing edges with the fuselage, as shown in Fig. 2.6. This approximation may result in a small amount of area being double-counted or neglected, depending on the exact fuselage shape.

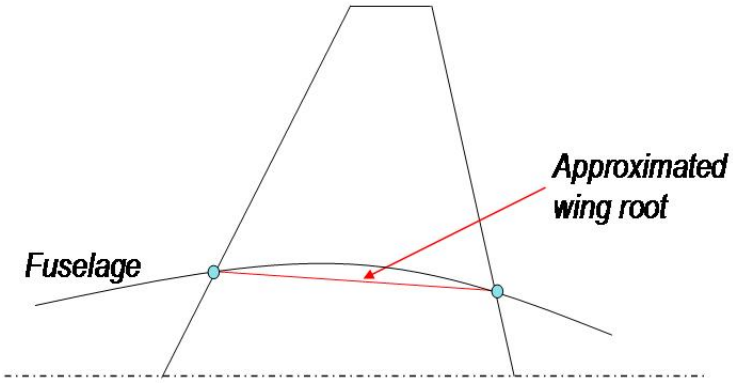


Figure 2.6. Approximate wing-fuselage intersection.

2.2.4 A502 Geometry Generation

A routine was developed to create the required input file for the panel method A502, which primarily contains geometry information. Input files for configurations ranging from full aircraft to a fuselage by itself can be generated using the parameterization described in Sec. 2.2.1. Below is an example of an aircraft configuration generated for A502. The different panel colors correspond to the different boundary conditions. Orange specifies panels with the flow tangency condition (i.e., actual aircraft geometry), light blue indicates lifting surface wake panels, and black corresponds to body-wing wake panels.

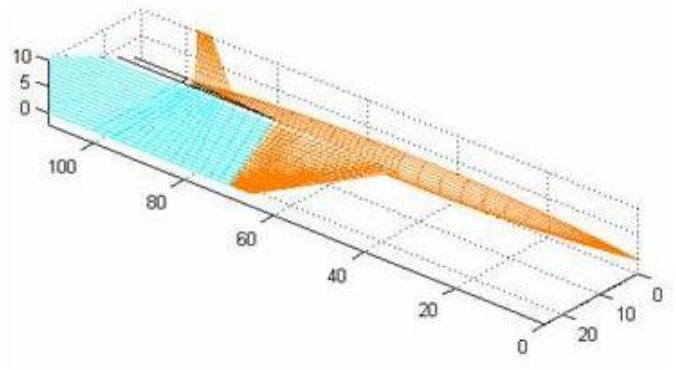
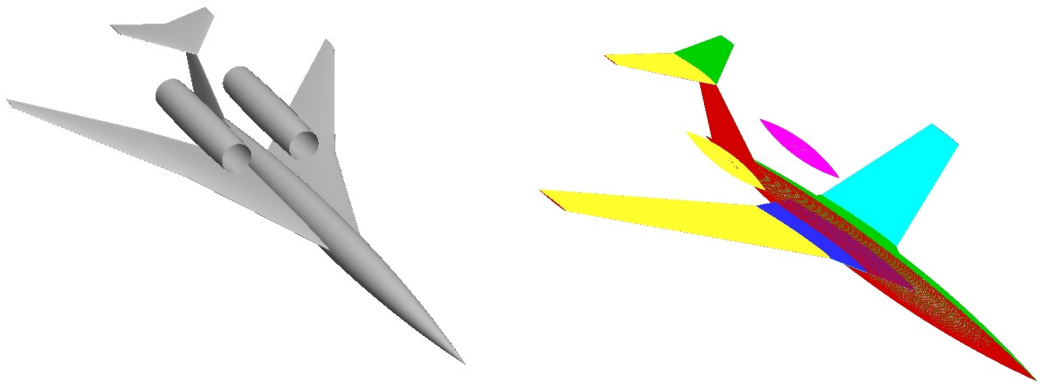


Figure 2.7. Aircraft representation for A502.

2.2.5 CART3D Geometry Generation

A routine was developed to generate `plot3d` files, which are used by CART3D to generate grids, based on the parameterization specified in Sec. 2.2.1. The method can generate full configurations, or any subset of the components used in a full aircraft configuration (i.e., just a fuselage, or wing, etc.). Some sample geometries are shown in Fig. 2.8.



(a) Flow through nacelles modeled

(b) Nacelles modeled as parabolic bodies

Figure 2.8. Sample geometries of supersonic aircraft generated with the new routine used to create `plot3d` files from PASS input files.

2.2.6 Code Comparisons

In this section the three methods are compared to one another for three wing-body configurations. The first geometry, shown in Fig. 2.9, is a low sweep configuration with a leading edge sweep of 21.80^o, $AR = 4$, $S_{ref} = 625$ ft^2 and 2% thick biconvex airfoils. The fuselage is a parabolic body with a fineness ratio of 20 and is 100 ft long. Fig. 2.10a-c show good agreement between the three methods for the low sweep configuration.

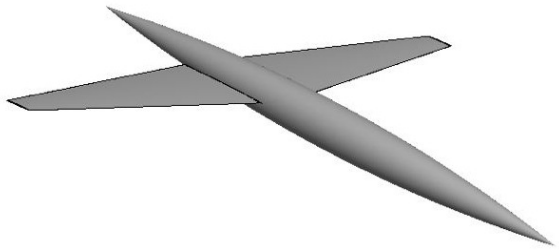
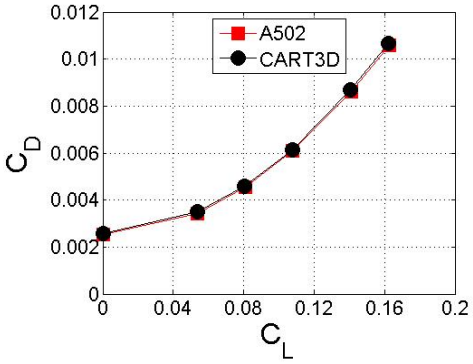
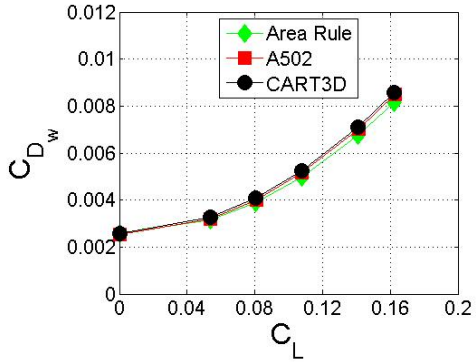


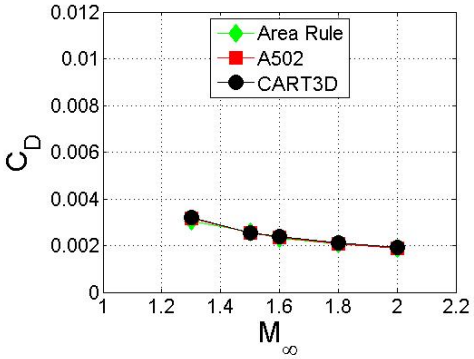
Figure 2.9. Low sweep configuration.



(a) C_D as a function of C_L, as wing incidence is varied, at $M_\infty = 1.5$.



(b) C_{D_w} as a function of C_L, as wing incidence is varied, at $M_\infty = 1.5$.



(c) C_D as a function of M_∞ at $C_L = 0$.

Figure 2.10. Comparison between CART3D, A502, and the area rule method for the configuration in Fig. 2.9.

Wing C_p distributions were extracted from CART3D and A502 solutions for one of the lifting cases and compared (Fig. 2.11).

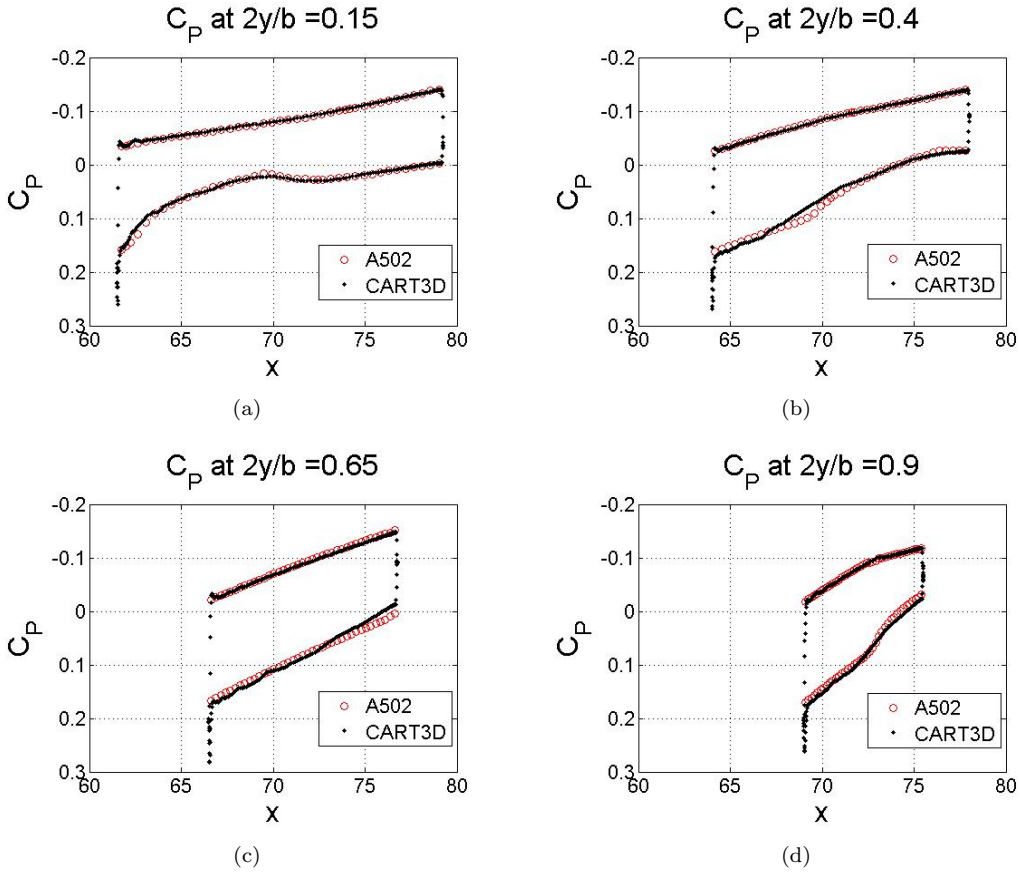


Figure 2.11. C_p Comparisons at four spanwise locations for $M_\infty = 1.5$ and a wing incidence of 2.61^o. CART3D predicts $C_L = 0.1407$ and $C_D = 0.0087$. A502 predicts $C_L = 0.1413$ and $C_D = 0.0086$.

As one may expect from the agreement seen in Fig. 2.10 the C_p distributions predicted by the two methods also agree quite well.

The second configuration, seen in Fig. 2.12, has a double delta planform that is based on a design study done at NASA [10] . It has $S_{ref} = 1775$ ft^2, $AR = 2.0285$, subsonic leading edges (inner leading edge sweep of 75.04^o and 64.54^o outer) with symmetric 3% thick NACA65 airfoils and a fuselage with a fineness ratio of 20 and a length of 120 ft. Although agreement is quite good for the non-lifting case (Fig. 2.13a), there is some disagreement in the lifting case (Fig. 2.13b). A comparison of C_p distributions is shown in Fig. 2.14.

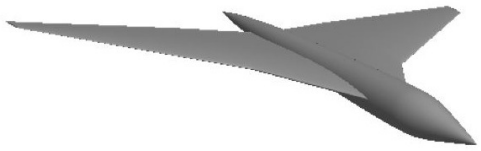
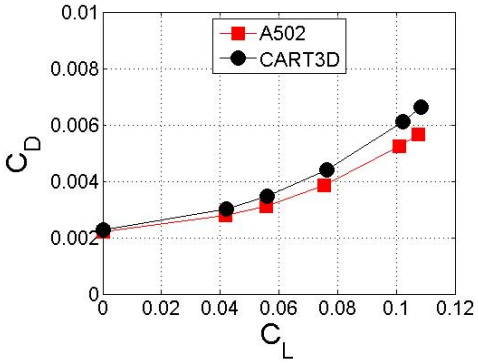
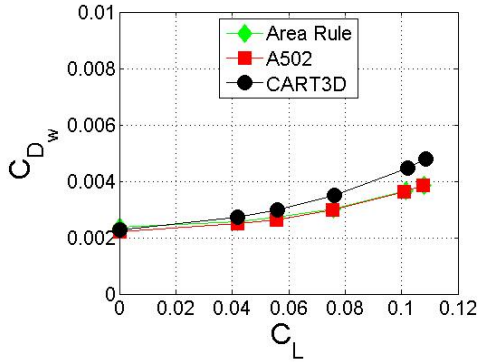


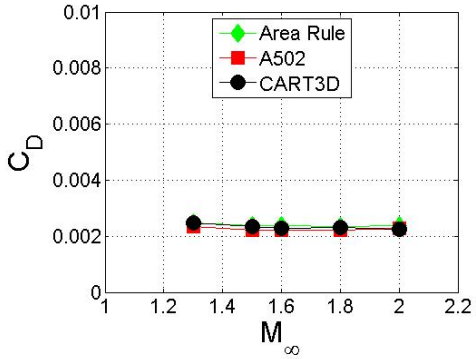
Figure 2.12. Double delta configuration.



(a) C_D as a function of C_L, as wing incidence is varied, at $M_\infty = 1.6$.



(b) C_{D_w} as a function of C_L, as wing incidence is varied, at $M_\infty = 1.6$.



(c) C_D as a function of M_∞ at $C_L = 0$.

Figure 2.13. Comparison between CART3D, A502, and the area rule method for the configuration in Fig. 2.12.

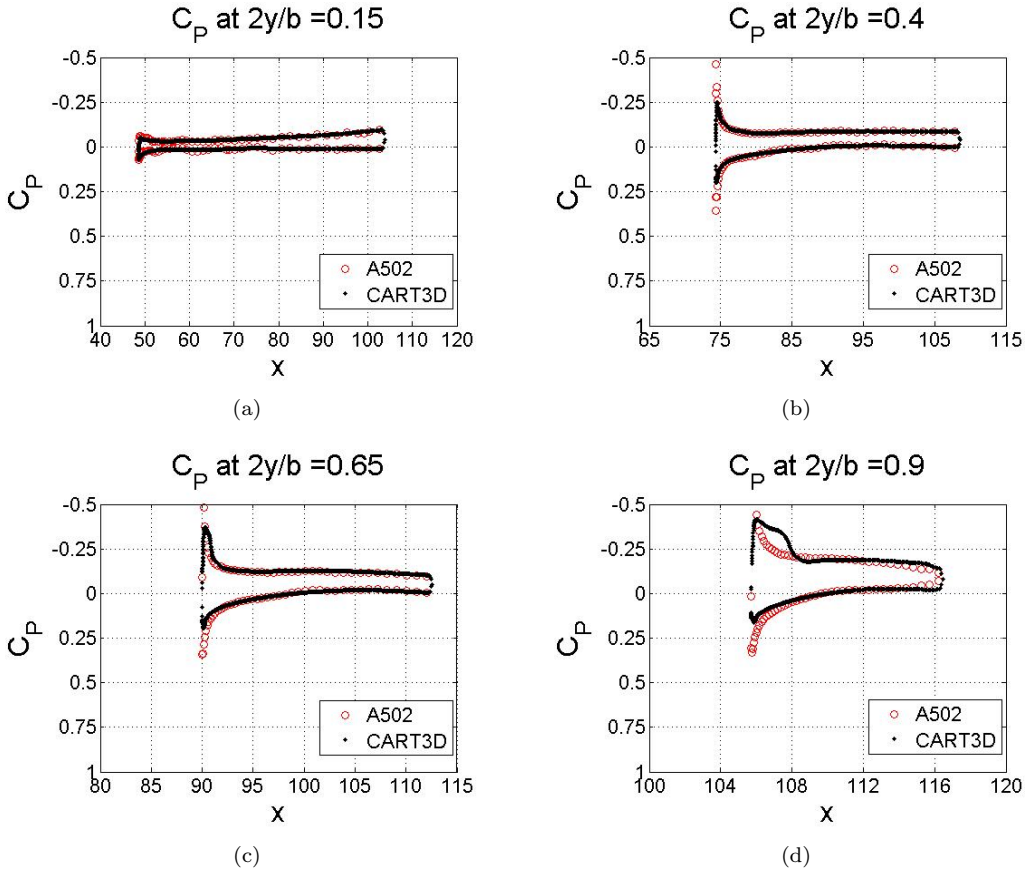


Figure 2.14. C_p comparisons at four spanwise locations for $M_\infty = 1.6$ and a wing incidence of 2.61^o. CART3D predicts $C_L = 0.1021$ and $C_D = 0.0061$. A502 predicts $C_L = 0.1010$ and $C_D = 0.0052$.

Fig. 2.14 shows that CART3D is predicting some transonic flow, most notably seen in Fig. 2.14d which appears to show a shock occurring roughly at 107 ft. The presence of the transonic flow over the outer portion of the wing will be missed by A502, and thus it makes sense that the drag predicted by CART3D is higher than that from A502.

Fig. 2.15) shows the high sweep configuration with $S_{ref} = 1812$ ft^2, $AR = 2.89$, a leading edge sweep of 54.45^o and symmetric 3% thick NACA65 airfoils and a fuselage with a fineness ratio of 20 and a length of 128 ft. The agreement between the three methods in C_D at zero lift, Fig. 2.16a, is good as long as $M_\perp$ is sufficiently below one (at $M_\infty = 1.6$ $M_\perp = 0.93$). Although A502 and the area rule method agree well in the lifting case (Fig. 2.16a and Fig. 2.16b), there is some disagreement with CART3D at higher values of C_L. As with the double delta configuration, these differences can be attributed to the regions of transonic flow occurring over the outer portion of the wing (see Fig. 2.17d).

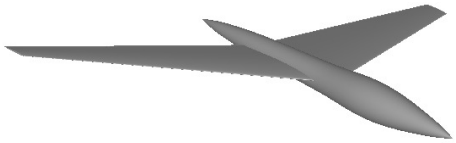
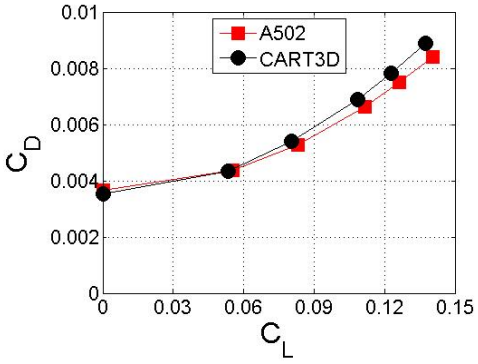
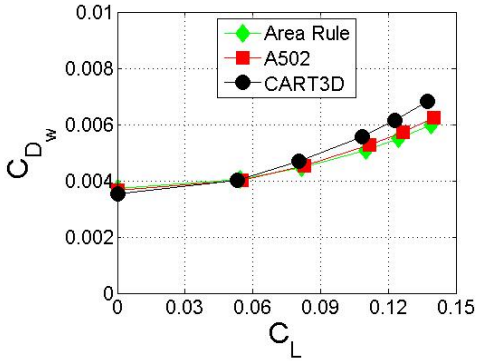


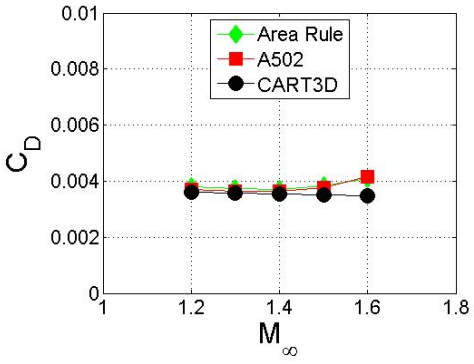
Figure 2.15. High sweep configuration.



(a) C_D as a function of C_L, as wing incidence is varied, at $M_\infty = 1.4$.



(b) C_{D_w} as a function of C_L, as wing incidence is varied, at $M_\infty = 1.4$.



(c) C_D as a function of M_∞ at $C_L = 0$.

Figure 2.16. Comparison between CART3D, A502, and the area rule method for the configuration in Fig. 2.15.

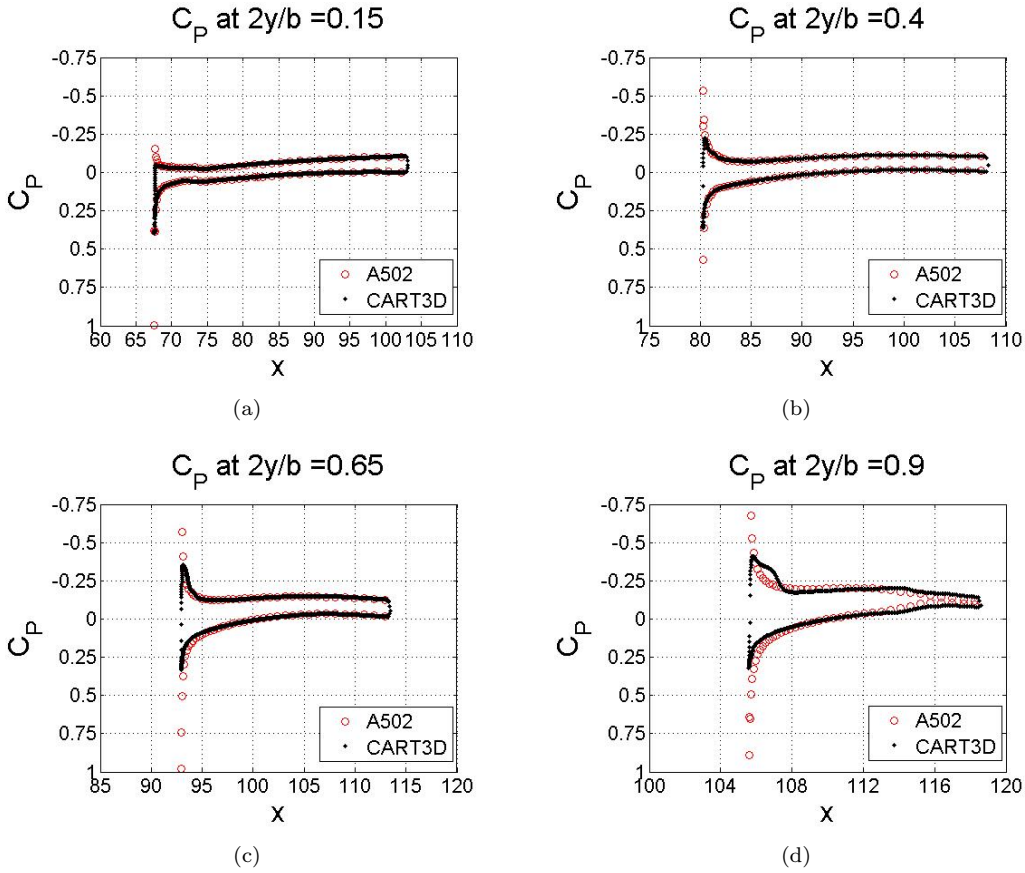


Figure 2.17. C_p Comparisons at four spanwise locations for $M_\infty = 1.4$ and a wing incidence of 2.25^o. CART3D predicts $C_L = 0.1228$ and $C_D = 0.0078$. A502 predicts $C_L = 0.1264$ and $C_D = 0.0075$.

2.3 Mission Analysis with Multifidelity Cruise Drag Estimates

In order to use the multifidelity optimization methods developed on full aircraft design problems a multifidelity mission analysis method was needed. The various methods developed to estimate wave drag have been integrated into a mission analysis routine to provide low- and high-fidelity estimates of cruise drag, thus providing low- and high-fidelity estimates of certain performance goals such as range, L/D and T/D.

The multifidelity mission analysis method developed revolves around PASS SE[11], which Desktop Aeronautics has supplied free of charge for this project. PASS SE is a modified version of PASS[12] that uses an area rule method to estimate supersonic wave drag (vortex drag is calculated assuming ideal spanwise loading on each surface), and a vortex-lattice method to estimate low speed aerodynamic performance. Thus the area rule method developed for the multifidelity wave drag code is not used in the multifidelity mission analysis code. PASS SE is used to estimate component weights, take-off and landing field lengths, subsonic aerodynamic performance, and subsonic stability. PASS SE can also estimate supersonic aerodynamic performance using its own area rule method. The routines previously developed to run A502 and CART3D were integrated with PASS SE to provide multifi-

delity estimates of cruise drag. This allows mission performance goals, such as range, to be computed in a multifidelity manner.

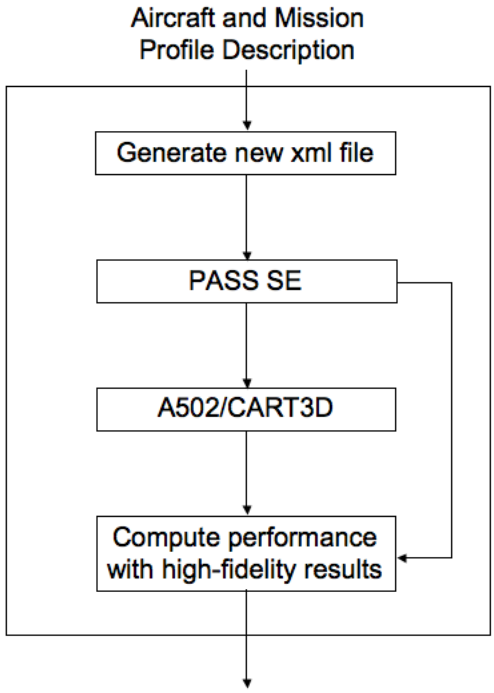


Figure 2.18. Multifidelity analysis method.

All data required for this analysis method is contained in an `xml` file that contains the aircraft parameterization described in Sec. 2.2.1. In addition to containing the parameterization of the aircraft's geometry these input files also contain parameters describing the engine (e.g., sea level static thrust), the mission profile (e.g., cruise altitude, take-off flap deflection and the number of passengers), and weight related parameters (e.g., maximum take-off weight).

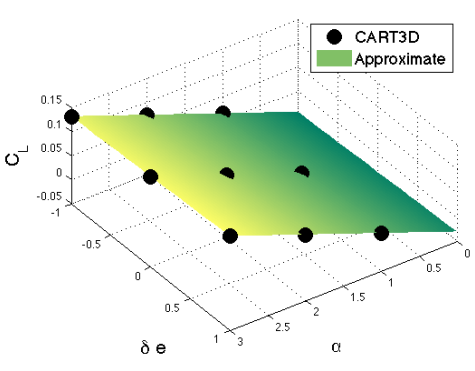
2.3.1 Trimming and C_L Matching with A502 and CART3D

Low-fidelity drag prediction routines, such as area rule methods like the one in PASS SE, assume the aircraft is trimmed at the desired C_L. Therefor, in order to compare drag estimates with high-fidelity methods, like A502 and CART3D , they must be done with the aircraft trimmed at the proper C_L. The same method (described below) does this for both A502 and CART3D.

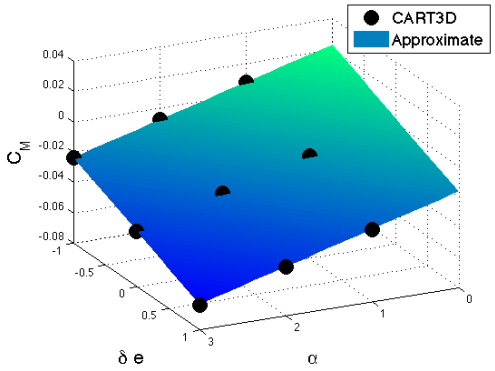
For a given aircraft configuration, C_L and C_m are functions of the aircraft's angle of attack (α) and horizontal tail deflection (δe). Furthermore, both C_L and C_m are well approximated by linear functions of α and δe (see Fig. 2.19).

$$C_L = C_{L_\alpha}\alpha + C_{L_{\delta e}}\delta e + C_{L_0} \quad (2.1)$$

$$C_m = C_{m_\alpha}\alpha + C_{m_{\delta e}}\delta e + C_{m_0} \quad (2.2)$$



(a) C_L as a function of α and δe.



(b) C_M as a function of α and δe.

Figure 2.19. C_L and C_M estimates as a function of angle of attack and elevator deflection.

A502 (or CART3D) is run three times with different combinations of α and δe, and the resulting values for C_L and C_m are used to determine the coefficients in Eqs. 2.1 and 2.2 from Eqs. 2.3 and 2.4.

$$\begin{bmatrix} \alpha_1 & \delta e_1 & 1 \\ \alpha_2 & \delta e_2 & 1 \\ \alpha_3 & \delta e_3 & 1 \end{bmatrix} \begin{pmatrix} C_{L_\alpha} \\ C_{L_{\delta e}} \\ C_{L_0} \end{pmatrix} = \begin{pmatrix} C_{L_1} \\ C_{L_2} \\ C_{L_3} \end{pmatrix} \tag{2.3}$$

$$\begin{bmatrix} \alpha_1 & \delta e_1 & 1 \\ \alpha_2 & \delta e_2 & 1 \\ \alpha_3 & \delta e_3 & 1 \end{bmatrix} \begin{pmatrix} C_{m_\alpha} \\ C_{m_{\delta e}} \\ C_{m_0} \end{pmatrix} = \begin{pmatrix} C_{m_1} \\ C_{m_2} \\ C_{m_3} \end{pmatrix} \tag{2.4}$$

Next, Eqs. 2.1 and 2.2 are solved for estimates of the angle of attack and tail deflection, $\tilde{\alpha}$ and $\tilde{\delta e}$, that should result in the aircraft being trimmed at the proper C_L (C_L^* is the desired C_L).

$$\tilde{\alpha} = \frac{-C_{L_0} C_{m_{\delta e}} + C_{m_0} C_{L_{\delta e}} + C_{m_{\delta e}} C_L^*}{C_{L_\alpha} C_{m_{\delta e}} - C_{L_{\delta e}} C_{m_\alpha}} \tag{2.5}$$

$$\tilde{\delta e} = \frac{-C_{L_\alpha} C_{m_0} + C_{m_\alpha} C_{L_0} - C_{m_\alpha} C_L^*}{C_{L_\alpha} C_{m_{\delta e}} - C_{L_{\delta e}} C_{m_\alpha}} \tag{2.6}$$

The analysis routine is run one more time at $\tilde{\alpha}$ and $\tilde{\delta e}$ giving $\tilde{C_L}, \tilde{C_m}, and \tilde{C_D}$. Note that because the mission consists of two cruise points, a total of eight A502 (or CART3D) runs are performed per function evaluation. This process results in $\tilde{C_L}$ being within 0.001 of C_L^* and the aircraft nearly trimmed, $\tilde{C_m} \approx 0.0005$. The estimate of drag, Eq. 2.7, is calculated by assuming that L/D is nearly constant for the configuration trimmed about the desired C_L.

$$C_D^{high} = \frac{\tilde{C_D}}{\tilde{C_L}} C_L^* \tag{2.7}$$

Based on the error in C_L and C_m typical of this method, the drag values are well within one count of the true drag (i.e.,the drag value if the aircraft were perfectly trimmed exactly at the desired C_L).

2.3.2 Surrogate Modeling of Inviscid Cruise Drag

The multifidelity algorithms discussed in this work all use surrogate models that represent the difference between estimate of a given function from low- and high-fidelity analysis methods. The method discussed in Chapter 3 uses quadratic model, Eq. 2.8, to represent the difference between low- and high-fidelity drag predictions. Because the analysis methods influence how the surrogate model is built, namely the computation of gradients, we discuss the methods here.

$$\tilde{\epsilon}(x - x_c) = \epsilon_0 + \nabla\epsilon(x_c)^T(x - x_c) + \frac{1}{2}(x - x_c)^T H(x_c)(x - x_c) \tag{2.8}$$

In Eq. 2.8 x_c denotes the current best design (analogous to the trust region center in a trust region method), ϵ_0 is the difference in drag at x_c, $\nabla\epsilon(x_c)$ is the gradient of the difference at x_c, and H is a quasi-Newton estimate of the Hessian of the difference. Currently the quasi-Newton method being used is the well-known BFGS update scheme.

Estimates of the gradient of drag from the low- and high-fidelity methods are needed to build the quadratic correction model. Thus, we must be able to compute the partial derivatives of drag with respect to the response surface parameters, i.e., aircraft geometry parameters and C_L. The following two sections discuss the methods used to compute these derivatives using the low- and high-fidelity methods.

Low-Fidelity Sensitivities

Computation of the gradient of drag from the low-fidelity analysis is complicated by the fact that it is not smooth, due to the area rule that provides estimates of wave drag. Examples of this behavior can be seen in Fig. 2.20.

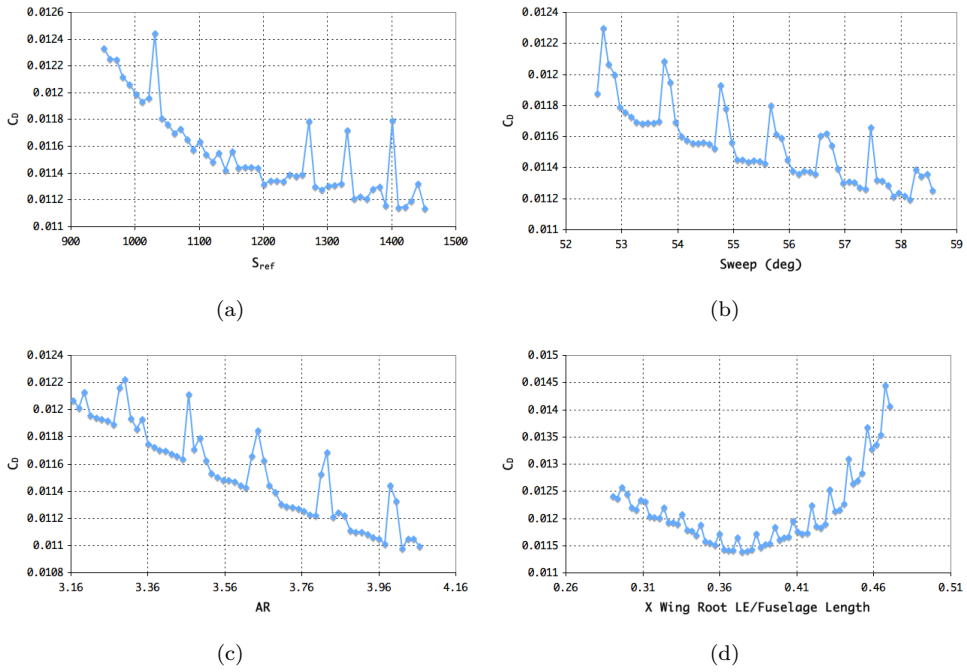


Figure 2.20. Low-fidelity estimates of inviscid cruise drag.

Two characteristics of the variation in drag predicted by the low-fidelity analysis are apparent from Fig. 2.20. First, the data are reasonably smooth apart from some spikes. Secondly,

the overall trends tend to be quadratic (and even linear in some cases). It seems reasonable to assume that if the "bad" data can be thrown out and a curve fit to the remaining data, a satisfactory estimate can be obtained of the derivative of drag with respect to a given parameter. Although this would be far too time consuming if the data were generated with CART3D, they are being generated by a method that takes roughly one second to run.

Therefore it seems quite reasonable, especially compared to the time it takes CART3D to run, to compute low-fidelity derivatives of drag in the following manner. Evaluate the drag at 20-30 points in some neighborhood of the current design, filter the data, and then estimate the derivative from the remaining data. The "bad" drag estimates show up as large spikes, as is evident from Fig. 2.20, and are thus simple to remove. A method has been developed that filters the drag data by removing any points for which the increase in drag is above a maximum rate of change (e.g., half a count for a change of only 1% in the design variable). An example of this smoothing technique is shown in Fig. 2.21.

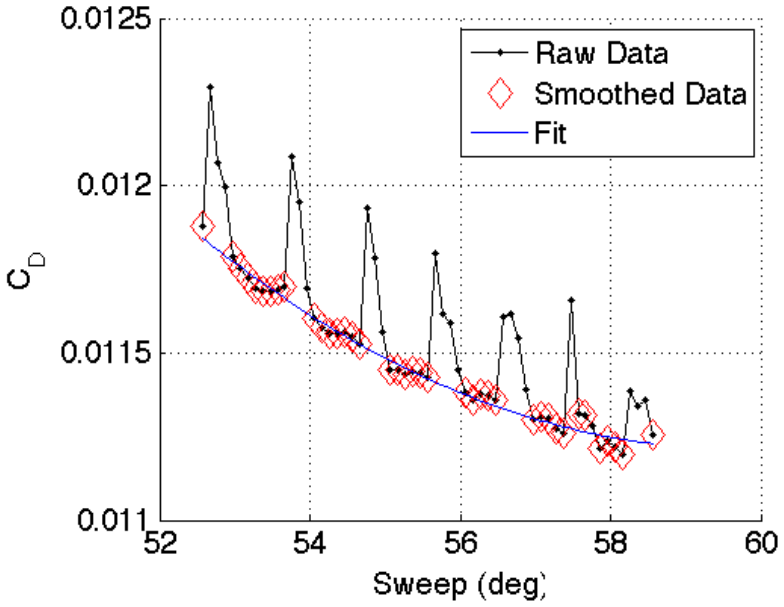


Figure 2.21. Example of data filtering technique used to estimate derivatives from a noisy area rule method.

High-Fidelity Sensitivities

Unlike the low-fidelty method, it is not feasible to run CART3D many times to get an estimate of the gradient. Because an adjoint method is not available to us finite differencing will be used, however, for this to work the high-fidelty estimates of inviscid drag must be smooth (because A502 is not smooth the discussion here assumes the high-fidelity method is CART3D).

To show that the high-fidelity drag function (Eq. 2.7) is smooth it must be so with respect to parameters that define the aircraft's geometry, x_i, and with respect to C_L. Differentiating Eq. 2.7 with respect to x_i or C_L^* gives the following (note that C_L^* is not a function of x_i).

$$\frac{\partial C_D^{high}}{\partial x_i} = \frac{\frac{\partial \tilde{C_D}}{\partial x_i}\tilde{C_L} - \frac{\partial \tilde{C_L}}{\partial x_i}\tilde{C_D}}{\tilde{C_L}^2} C_L^* \tag{2.9}$$

$$\frac{\partial C_D^{high}}{\partial C_L^*} = \frac{\frac{\partial \tilde{C_D}}{\partial C_L^*}\tilde{C_L} - \frac{\partial \tilde{C_L}}{\partial C_L^*}\tilde{C_D}}{\tilde{C_L}^2} C_L^* + \frac{\tilde{C_D}}{\tilde{C_L}} \tag{2.10}$$

So long as $\frac{\partial \tilde{C_D}}{\partial x_i}$, $\frac{\partial \tilde{C_L}}{\partial x_i}$, $\frac{\partial \tilde{C_D}}{\partial C_L^*}$, and $\frac{\partial \tilde{C_L}}{\partial C_L^*}$ are smooth, and $\tilde{C_L}$ is never zero (as we are concerned with cruise conditions, this should not occur), the drag function will be smooth. Recall that both $\tilde{C_L}$ and $\tilde{C_D}$ are functions of x_i, $\tilde{\alpha}$, $\tilde{\delta e}$, and that C_L^* is not a function of any of those parameters . Furthermore, $\tilde{\alpha}$ and $\tilde{\delta e}$ are functions of x_i andC_L^*. Differentiating $\tilde{C_D}$ and $\tilde{C_L}$ with respect to x_i or C_L^* gives the following.

$$\frac{\partial \tilde{C_L}}{\partial x_i} = \left.\frac{\partial \tilde{C_L}}{\partial x_i}\right|_{\tilde{\alpha},\tilde{\delta e}=const} + \frac{\partial \tilde{C_L}}{\partial \tilde{\alpha}}\frac{\partial \tilde{\alpha}}{\partial x_i} + \frac{\partial \tilde{C_L}}{\partial \tilde{\delta e}}\frac{\partial \tilde{\delta e}}{\partial x_i} \tag{2.11}$$

$$\frac{\partial \tilde{C_D}}{\partial x_i} = \left.\frac{\partial \tilde{C_D}}{\partial x_i}\right|_{\tilde{\alpha},\tilde{\delta e}=const} + \frac{\partial \tilde{C_D}}{\partial \tilde{\alpha}}\frac{\partial \tilde{\alpha}}{\partial x_i} + \frac{\partial \tilde{C_D}}{\partial \tilde{\delta e}}\frac{\partial \tilde{\delta e}}{\partial x_i} \tag{2.12}$$

$$\frac{\partial \tilde{C_L}}{\partial C_L^*} = \frac{\partial \tilde{C_L}}{\partial \tilde{\alpha}}\frac{\partial \tilde{\alpha}}{\partial C_L^*} + \frac{\partial \tilde{C_L}}{\partial \tilde{\delta e}}\frac{\partial \tilde{\delta e}}{\partial C_L^*} \tag{2.13}$$

$$\frac{\partial \tilde{C_D}}{\partial C_L^*} = \frac{\partial \tilde{C_D}}{\partial \tilde{\alpha}}\frac{\partial \tilde{\alpha}}{\partial C_L^*} + \frac{\partial \tilde{C_D}}{\partial \tilde{\delta e}}\frac{\partial \tilde{\delta e}}{\partial C_L^*} \tag{2.14}$$

The partial derivatives of $\tilde{C_L}$ and $\tilde{C_D}$ with respect to x_i (with $\tilde{\alpha}$ and $\tilde{\delta e}$ held constant), $\tilde{\alpha}$, and $\tilde{\delta e}$ should be smooth.

Thus, if $\frac{\partial \tilde{\alpha}}{\partial x_i}$, $\frac{\partial \tilde{\delta e}}{\partial x_i}$, $\frac{\partial \tilde{\alpha}}{\partial C_L^*}$ and $\frac{\partial \tilde{\delta e}}{\partial C_L^*}$ are smooth then this method should provide smooth predictions of drag. From Eqs. 2.5 and 2.6 it is clear that $\frac{\partial \tilde{\alpha}}{\partial C_L^*}$ and $\frac{\partial \tilde{\delta e}}{\partial C_L^*}$ are smooth. For $\frac{\partial \tilde{\alpha}}{\partial x_i}$ and $\frac{\partial \tilde{\delta e}}{\partial x_i}$ to be smooth all of the coefficients in Eqs. 2.5 and 2.6 must be smooth with respect to x_i. This will be true if $C_{L_{1,2,3}}$ and $C_{m_{1,2,3}}$ from Eqs. 2.3 and 2.4 are smooth with respect to x_i, which should be true. Note that we have assumed the absence of complex shock interactions that could cause sudden changes to the flow field. Although this condition should hold true for the aircraft geometries being analyzed in this work, the possibility, no matter how remote, should not be disregarded. In addition, it has been assumed that the flow solver itself, including mesh generation, and geometry generation, will behave in a smooth manner.

2.4 ModelCenter Interface

ModelCenter has been shown to be an efficient platform for multifidelity optimization of supersonic aircraft [13]. It enables a designer to choose a desired fidelity level and to integrate many analysis methods into a single multidisciplinary optimization. Therefore, a method to integrate ModelCenter with the multifidelity methods discussed in this report has been developed.

The multifidelity line search method discussed in Chapter 3 has been implemented in Java. Thus, `ModelCenter.jar` can be used to create a simple and direct interface with ModelCenter using the API. This allows the optimization method to use existing analysis

routines that have been developed in ModelCenter. Information on the Java code can be found in the `MultifidelityOptManual.pdf`, which was sent along with this report.

The local multifidelity optimization methods based on Bayesian model calibration (Chapters 9 and 10) are written in Matlab. Those methods behave just like the built-in Matlab optimization toolboxes and can optimize any Matlab function. Therefore, a Matlab function that instantiates the ModelCenter API, runs a ModelCenter project, and returns the result has been developed. This enables seamless integration between the multifidelity optimization methods and ModelCenter. Specific instructions to install this interface, for the Matlab code, on a Windows platform are included in the Appendix.

The ModelCenter interface, in Java and Matlab, was tested with the simple multifidelity Rosenbrock example shown in Figure 2.22. These methods are able to run the model and return the result in a fraction of a second. Thus these interfaces provide an effective, low-overhead approach to optimizing ModelCenter projects using multifidelity optimization methods that are not native to ModelCenter.

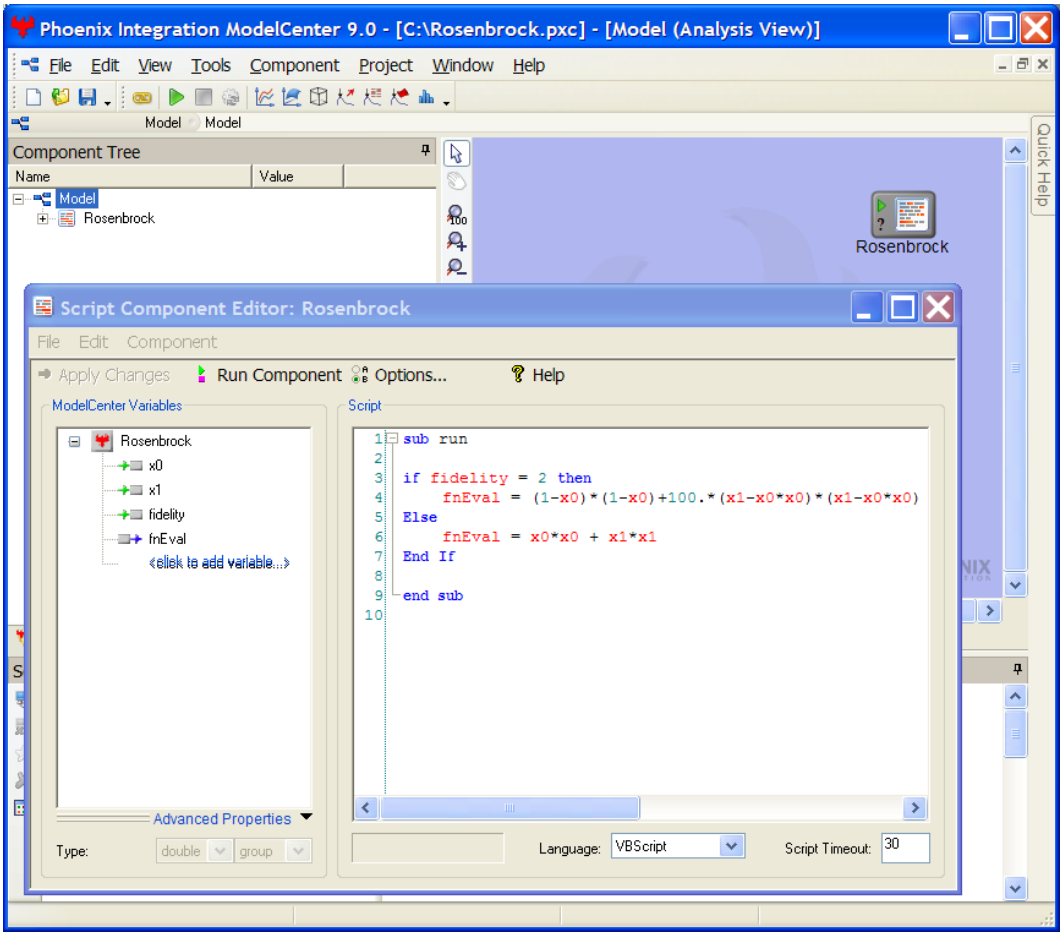


Figure 2.22. Screenshot of a ModelCenter project to optimize a multifidelity Rosenbrock problem.

3 A Multifidelity Line Search Method

Trust region methods often work quite well on simple or low-dimensional problems because the response surface is accurate over large regions of the design space. As the functions of interest become more complex or the dimensionality increases, however, the convergence rates of these methods can be quite slow. This can happen when the response surface provides poor estimates of the true function, and the trust region must shrink, sometimes significantly, before any progress can be made. Excessively small trust regions often result in very limited progress at each iteration, thus making these methods inefficient on more complex problems, such as those being considered in this work.

Removing the trust region from the process may permit a more efficient multifidelity method on complex problems. One possibility comes from the theory underlying SQP methods. These methods solve a subproblem in which the real function is approximated by a quadratic and a line search is then performed in the direction of the quadratic's optimum. We propose the use of a corrected low-fidelity model rather than the purely quadratic model, with the subsequent line search in the direction of the point it identifies. Algorithm 1 shows how this method may be implemented.

Algorithm 1 Overview of a multifidelity line search method.

```
Initialize surrogate models
repeat
    Find x̃*_k by minimizing corrected low-fidelity model
    Perform a line search in the direction of x̃*_k − x_c to find x*_k
    if Improvement is not made then
        Take a step in the steepest descent direction to find x*_k and reset the Hessian
    end if
    Set x_c = x*_k and compute information needed for the fit (such as the high-fidelity
    gradient)
    Update surrogate models with any new information
    Update Lagrange multipliers if constraints exist
until convergence
```

When constraints are present the objective function becomes the Lagrangian, or Augmented Lagrangian, and the Lagrange multipliers can be estimated in the same way as in trust region algorithms [14].

If the surrogate model is poor for large distances from x_c there may be little to no imrpovement in the true function along the line $x_k^* - x_c$. The surrogate model may, however, still be quite reasonable for smaller distances. This implies that the corrected low-fidelity optimization may perform well initially and only becomes misleading when the distance from x_c has increased significantly. Therefore, even if the true performance of the high-fidelity function is worse at x_k^*, and there may be no improvement along path A (see Fig. 3.1), there should be a point along path B showing improved performance (as we approach x_c along path B the step will start to point in the steepest descent direction). Thus, we propose a second method that backtracks along the trajectory of the corrected low-fidelity optimization. This method is outlined in Algorithm 2.

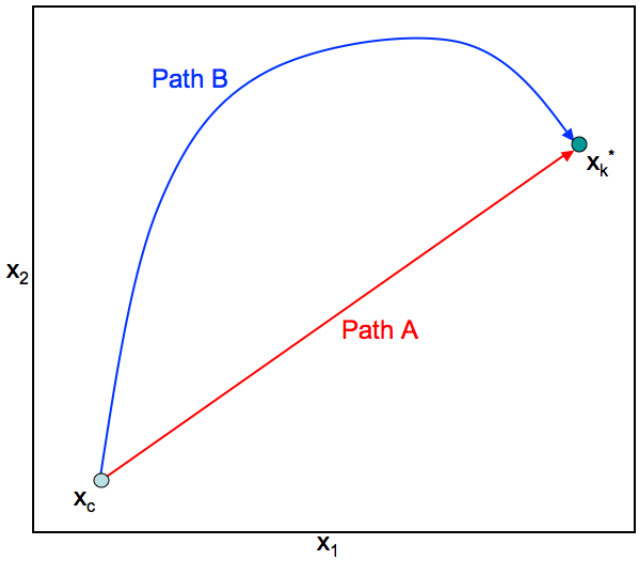


Figure 3.1. Possible search directions based on the optimum of the corrected low-fidelity model. Path A represents the line from x_c to x_k^*, while path B represents the path taken while optimizing the corrected low-fidelity model.

Algorithm 2 An alternative to the multifidelity line search described by Algorithm 1.

1: Initialize surrogate models
2: **repeat**
3: Find $\tilde{x}_k^*$ by minimizing corrected low-fidelity model
4: Backtrack along the path of the low-fidelity optimization to find x_k^*
5: **if** Improvement is *not* made **then**
6: Take a step in the steepest descent direction to find x_k^* and reset the Hessian
7: **end if**
8: Set $x_c = x_k^*$ and compute information needed for the fit (such as the high-fidelity gradient)
9: Update surrogate models with any new information
10: Update Lagrange multipliers if constraints exist
11: **until** convergence

As an example we compared the two methods described above, as well as a trust region method, in minimizing the Rosenbrock function. The low-fidelity model is given by Eq. 3.2.

$$f_{High} = (1 - x_1)^2 + 100\left(x_2 - x_1^2\right)^2 \tag{3.1}$$

$$f_{Low} = 50\left(x_2 - x_1^2 - 0.3\right)^2 \tag{3.2}$$

This low-fidelity model is somewhat representative of what might be expected from a low-fidelity model in a realistic engineering problem. The general shape of the function will point the optimizer toward promising regions of the design space, i.e., where $x_2 \approx x_1^2$, however, the location of optima differ. In this case, any point along the curve $x_2 = x_1^2 + 0.3$ will yield the minimum of the low-fidelity function, zero, while the high-fidelity optimum of (1,1) does not lie along this curve. Fig. 3.2 shows the performance of each method, including

direct optimization of the high-fidelity function with Matlab's SQP solver `fmincon`, for 50 high-fidelity function evaluations.

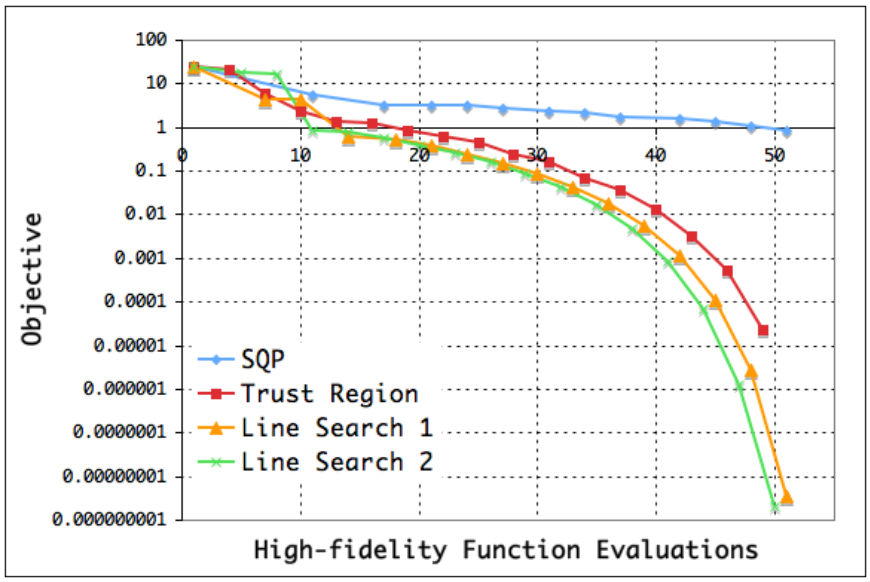


Figure 3.2. Minimization of the Rosenbrock function using four different optimization methods.

The three multifidelity methods clearly outperform the SQP method. Of the three, the two line search methods converge faster than the trust region method. The first line search method appears to have a modest performance advantage initially, while the second one shows better performance later in the optimization process. The poor performance of the second line search method during the first few iterations is likely due to the back tracking algorithm employed here. Simply, we back track along the low-fidelity optimization path until any improvement in the objective function is found. The use of a more intelligent back tracking scheme (e.g., Brent's method[15], which is used in the first line search method) should improve the performance of this method during the first few iterations. Despite the promising performance of these two line search methods in this example, far more experimentation is needed before any conclusions can be reached about the effectiveness of either method.

4 Supersonic Aircraft Design Problem

The multifidelity optimization method presented in Chapter 3 is currently being used to design an aircraft that falls within NASA's "N+2" generation of supersonic aircraft, i.e., a small supersonic airliner (SSA). We have chosen to design an aircraft that will carry 39 passengers, cruise at Mach 1.6, and have a range of 4200 nmi. TOFL and LFL are those specified by NASA for their "N+2" public domain configuration. Table. 4.1 summarizes these design goals.

Cruise Mach Number	1.6
Range	4200 nmi
TOFL	9000 ft
LFL	6500 ft
Payload	39 pax

Table 4.1. SSA design goals.

4.1 Preliminary Design

Prior to applying our multifidelity design techniques to this problem a preliminary design was generated using PASS SE for mission analysis and inviscid drag predictions. The following optimization problem was used to find the preliminary design and optimization was done with the Nelder-Mead simplex method[16].

Minimize	Cost		
w.r.t.	Design Variables		
s.t.	Range	$\geq$	4200 nmi
	TOFL	$\leq$	9000 ft
	LFL	$\leq$	6500 ft
	2^{nd} Segment Climb	$\geq$	0.024
	$(D/T)_{Initial}$	$\leq$	0.95
	$(D/T)_{Final}$	$\leq$	0.95
	$(D/T)_{Transonic}$	$\leq$	0.95
	Subsonic Static Margin	$\geq$	0.0
	Fuselage diameter at stations 4-6	$\geq$	8.79
	Fuel Volume	$\geq$	Required Fuel Volume
	C_L Margins	$\geq$	Min. C_L Margins
	Elevator Deflections	$\leq$	Max. Allowed Deflections
	x/c gear	$\leq$	0.76

Table 4.2. Optimization problem for preliminary design of the SSA.

The objective function is a metric for cost that is both a function of airframe and engine costs as well as operational costs, such as fuel burn. The idea is that we do not want to design an aircraft that is very light, to gain an inexpensive airframe, if this is at the expense of burning significantly more fuel, thus resulting in very high operational costs. The mission

is composed of various segments, e.g., transonic flight, initial cruise, and final cruise, and we require the aircraft to have sufficient thrust during all of these stages. Because some fuselage radii were allowed to vary, a constraint on the minimum radius in the cabin section was imposed. The constraints on wing and horizontal tail C_L margins and elevator deflections are imposed during all flight conditions (take-off, climb, cruise, etc.). Additionally, C_L margin on the vertical tail has been constrained during engine out conditions. Lastly the main landing gear was required to be a sufficient distance from the wing's trailing edge. Although a total of 23 parameters were allowed to vary, only a subset were allowed to vary during any given optimization (numerous optimizations were performed). Table. 4.3 lists all of the design variables, and the wing-related parameters are shown in Fig.2.3.

MTOW
S_{ref}
AR of S_{ref}
Quarter chord sweep of S_{ref}
LEX
TEX
Chord extension span
x position of wing root leading edge
$(t/c)_{Root}$,$(t/c)_{Break}$,$(t/c)_{tip}$
S_H/S_{ref}
Radius at fuselage stations 5-9
S.L.S. Thrust
x position of engine
Initial and final cruise altitudes
Take-off and landing flap deflections

Table 4.3. Design variables. See Fig. 2.3 for the wing related parameters.

Fig. 4.1 shows the optimized fuselage. For the preliminary design we assumed that the cabin runs from just aft of the cockpit (station 3) to station 6, providing roughly 50 ft of cabin space. Thirteen rows of 36" pitch seating requires at least 39 ft. With roughly 50 ft of cabin space this might be feasible, though a slight reduction may be needed to accommodate the galleys and bathrooms. The height over the majority of the cabin is large enough so that people less than 6' 2" will not have to crouch while walking. Although the fineness ratio of this fuselage is roughly 17.3, which may be considered low, this aircraft performed better than other configurations with higher fineness ratios that were looked at.

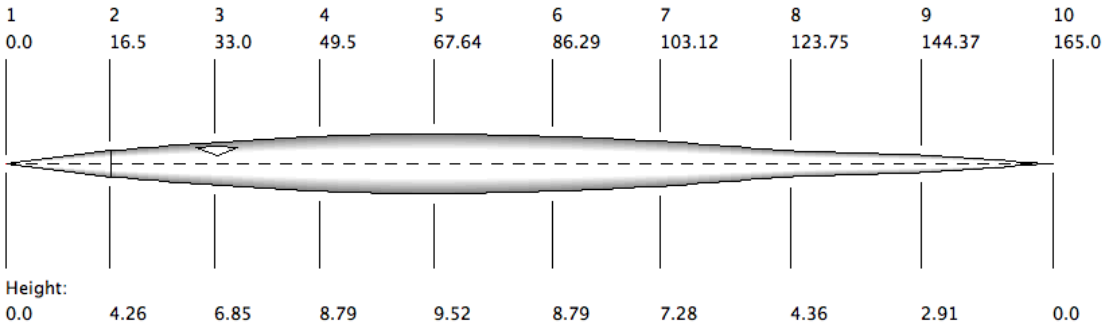


Figure 4.1. Preliminary fuselage design.

Fig. 4.2 shows the preliminary configuration of this aircraft and Table. 4.4 provides some sizing data. Wing size was driven primarily by the LFL and main gear location requirements, which pushed the wing size up, while cruise performance (i.e., reduced drag in cruise) attempted to reduce the wing size. The optimizer appears to have found the smallest wing possible while still satisfying the requirements for LFL and main gear location. In order to prevent a large rise in induced drag during transonic flight as the wing area was reduced, the aspect ratio became larger then might be expected for a supersonic aircraft. TOFL, due to the large allowable distance, was rarely an active constraint. The large TOFL also allowed the aircraft to take-off at a relatively low C_L, resulting in 2^{nd} segment climb being easily satisfied. Although the engines might have been reduced further to take advantage of the large margin in the TOFL constraint, they appeared bound by the requirement for sufficient thrust during initial and final cruise, both of which were active constraints.

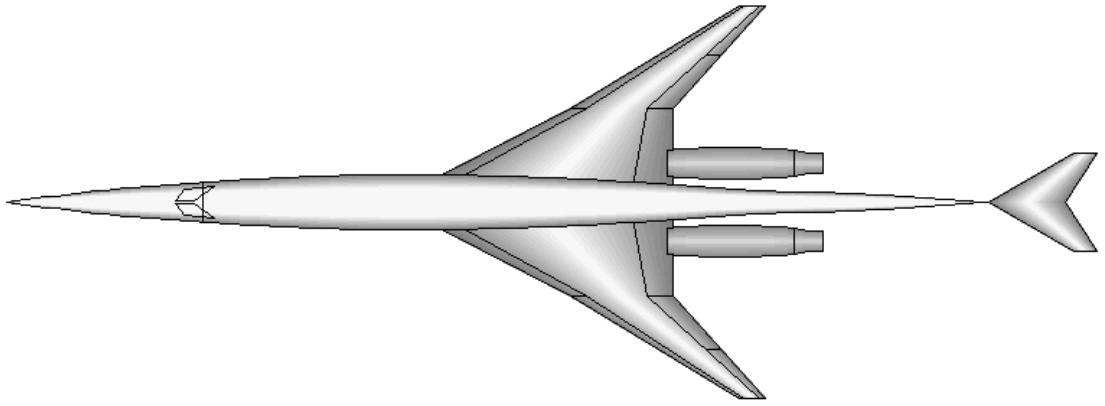


Figure 4.2. Preliminary SSA design.

MTOW (lb)	124195
S_{gross} (ft^2)	1392
AR_{gross}	3.15
Inner/outer leading edge sweeps (deg)	62.5/58.74
S_H/S_{total}	0.1
S.L.S. Thrust (lb)	31028
Initial/final cruise altitudes (ft)	45395/55435

Table 4.4. Preliminary SSA sizing.

4.1.1 Baseline SSA Performance with CART3D Drag Estimates

The baseline SSA configuration was analyzed with the high-fidelity method (CART3D) for cruise drag estimation and the results are shown in Table 4.5.

Drag Method	PASS SE	CART3D
Range (nmi)	4200	3690
Initial (D/T)	0.95	1.08
Final (D/T)	0.95	1.08
Initial cruise C_D (drag counts)	119.1	155.5
Final cruise C_D (drag counts)	114.9	152.1

Table 4.5. Cruise performance computed with PASS SE and CART3D drag estimates.

The large increase in inviscid drag, roughly 36 counts, results in an aircraft with insufficient range and thrust during cruise. This large increase results from the baseline configuration lacking ideal spanwise and longitudinal lift distributions along with the presence of shocks over the wing. The spanwise lift distribution, which is clearly not elliptic, is plotted in Fig. 4.3 against an elliptic lift distribution for the desired C_L. The C_p distributions at four spanwise locations are plotted in Fig. 4.4, which clearly show the presence of shocks sitting on the wing's upper surface.

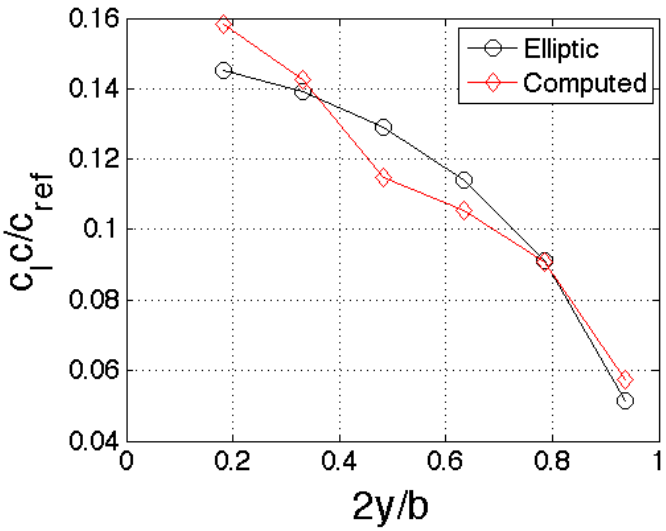
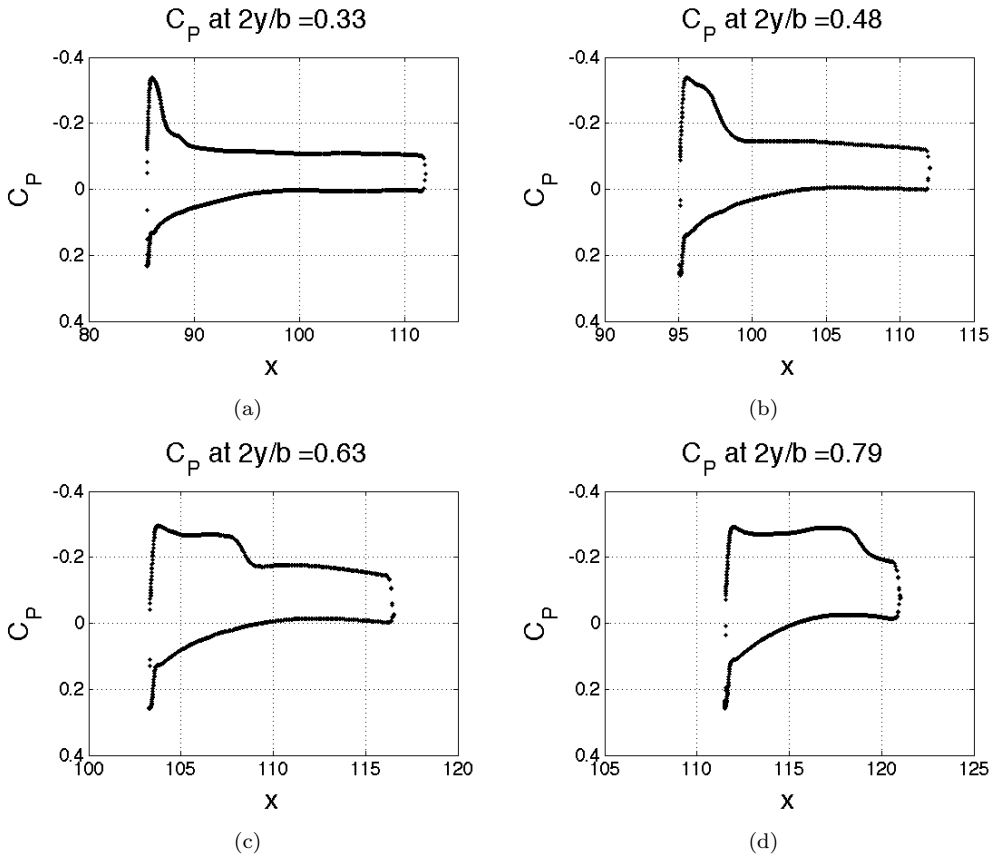


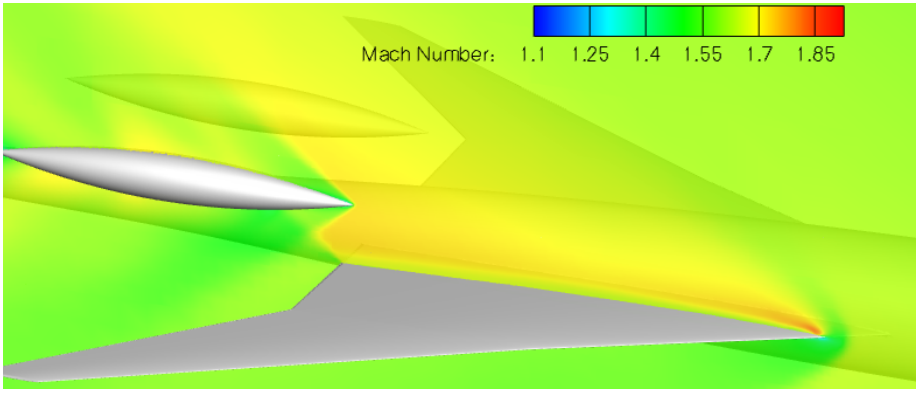
Figure 4.3. Spanwise lift distribution of the baseline SSA at a C_L of 0.1958.



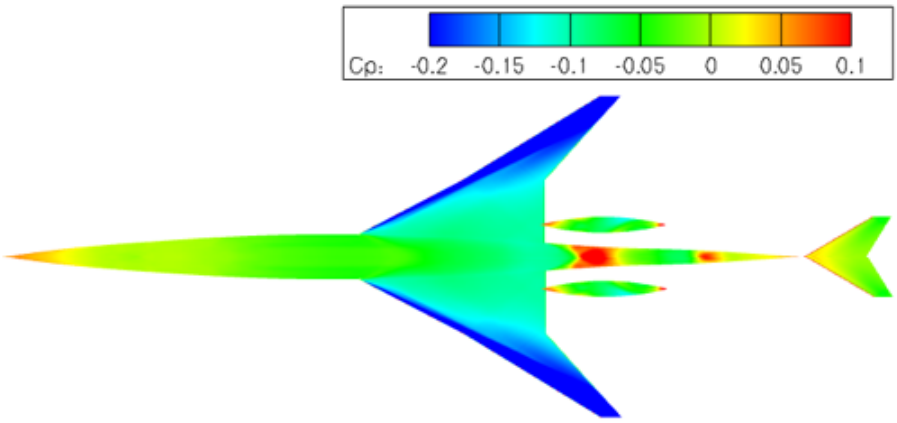
(a) (b) (c) (d)

Figure 4.4. C_p cuts of the baseline SSA at a C_L of 0.1958.

Fig. 4.5a shows the variation in Mach number over the wing as well as the interaction between the flow leaving the wing and the pod (representing the nacelle). Note that the shocks emanating from the pod do not hit the wing's trailing edge. Fig. 4.5b shows the distribution of C_P on the upper surface of the baseline configuration at cruise conditions. Note the sudden rise in C_p in the streamwise direction over the wing (denoted by the abrupt change from dark blue to light blue), which runs along the wing's span.



(a) Mach number plot showing the flow over the wing and pod.



(b) Upper surface C_p distribution.

Figure 4.5. Flow field about the baseline configuration at cruise conditions.

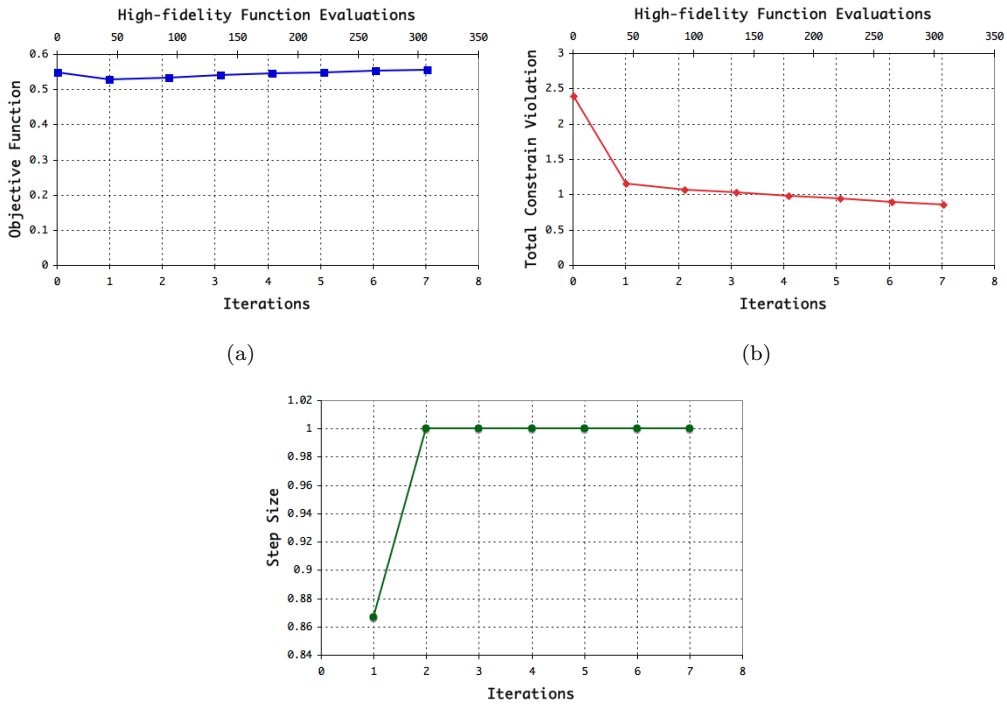
4.2 Multifidelity Optimization Results

The baseline configuration was optimized using the multifidelity line search method discussed in Chapter 3 and the mission analysis routine discussed in Sec. 2.3, with CART3D as the high-fidelity method and the area rule method as the low-fidelity method. Finite differencing was done with a central difference method and Brent's method was used to perform the line search. Because the mission analysis routine is not completely smooth, a Nelder-Mead simplex method was used to perform the low-fidelity optimization. The multifidelity optimization problem is shown in Table 4.2 and the design variables and response surface variables are given in Table. 4.6.

Design Variables	Response Surface Variables
AR	
S_{ref}	
Quarter chord sweep of S_{ref}	
LEX	
TEX	
Chord extension span	
x position of wing root leading edge	
$(t/c)_{Break}$ and $(t/c)_{tip}$	
Incidence$_{Break}$ and Incidence$_{tip}$	
Max. Camber$_{Break}$ and Max. Camber$_{tip}$	
S_H/S_{ref}	
Fuselage radius at stations 7 and 8	
S.L.S. Thrust	
x position of engine	
MTOW	C_L
Initial and final cruise altitudes	–
Take-off and landing flap deflections	–

Table 4.6. Design and response surface variables.

Fig. 4.8 shows the objective function, constraint violation, and step size history for the first run (a total of seven iterations). Note that the objective function, cost, has been scaled to be on the order of one. Each constraint has also been scaled to be on the order of one, and the constraint violation plotted is the sum of all the violated constraints.

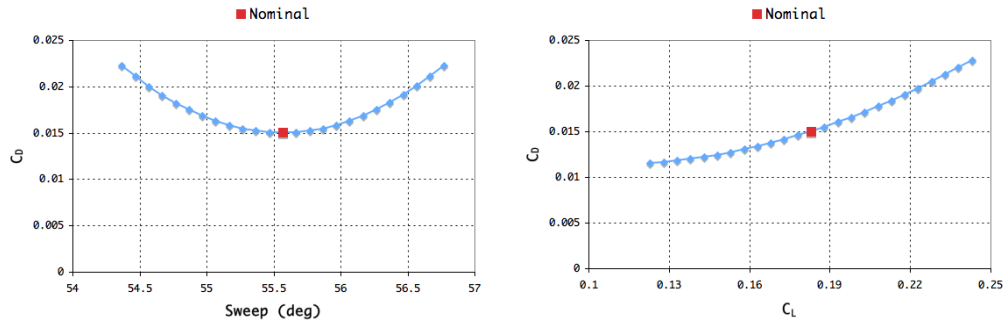


(a) (b)

(c)

Figure 4.6. Results from the first multifidelity optimization of the SSA.

Fig. 4.8 shows that the optimizer was able to significantly reduce the total constraint violation, with a reduction in the objective function, during the first iteration. After this iteration, however, very little progress was made. This behavior is likely due to how the Hessian was initialized. In this case, it was initialized to the identity matrix, i.e., $H_{ii} = 1.0$. A value of one in the Hessian produces a significant amount of curvature in the associated direction. In other words, changing a parameter by a relatively small amount will produce a large change in drag. Fig. 4.7a and b show this behavior for the corrected low-fidelity drag function, from the first iteration, as a single parameter is varied.



(a) (b)

Figure 4.7. Variation in the corrected low-fidelity drag function from the first iteration.

Fig. 4.7a shows that, due to the curvature, there is essentially no reduction in drag about

the nominal design for sweep. In fact, the only response surface parameter with an apparent possibility for a significant reduction in drag was C_L, as Fig. 4.7b shows. In fact, a reduction in drag only seemed possible by reducing C_L, as Fig. 4.7b shows. Therefore, the optimizer exploited this aspect of the corrected low-fidelity drag function and reduced the total constraint violation in the only way it could, i.e., by cruising at a lower altitude and thus reducing C_L. This, of course, improved (D/T) at the expense of range.

A quadratic fit to drag was generated by evaluating a large enough number of designs, throughout the design space, so that least squares could be used to determine the coefficients of the quadratic. For better speed, the fit was generated by running A502, rather then CART3D. The diagonal elements of the initial Hessian were then set to be of the same order of magnitude as the coefficients found from the fit of A502 data. Fig. 4.8 shows the objective function, constraint violation, and step size history for this run (a total of seven iterations).

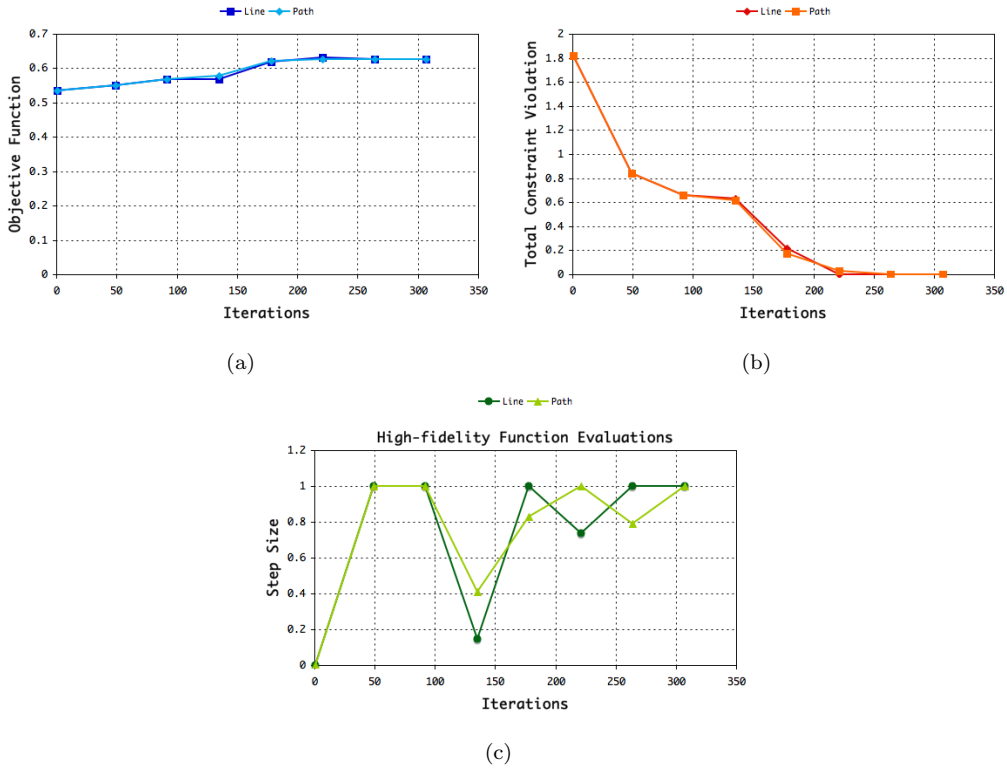


Figure 4.8. Optimization history.

As Fig. 4.8 shows, this time the optimizer was able to find a feasible solution. The correction to the fit allowed both D/T and range constraint violations to improve during the first iteration, rather than only D/T as before, and significant reduction in constraint violation after the first iteration. Fig. 4.8c shows that the optimum of the corrected low-fidelity function, in all but one iteration, was very close to the best point along the line (a step size of 1.0 indicates a step to the optimum of the low-fidelity optimization), suggesting that the drag correction fit was doing a good job this time.

Fig. 4.8 also shows a run done with the path following algorithm discussed in Chapter 3. The points tested during the corrected low-fidelity optimization were clustered using k-medoids[17], and the cluster centers evaluated with the high-fidelity analysis (six points were analyzed each iteration). It should be noted that this clustering algorithm finds a

cluster center that is a data point, rather then an average of nearby points. Fig. 4.8a and b show that the two runs were nearly identical. Fig. 4.8c shows step size which, for the path following run, was computed as: the distance from the current point to the new point divided by the distance from the current point to the optimum of the corrected low-fidelity model.

Fig. 4.9 compares the C_p distribution of the initial and optimized designs and Table. 4.7 provides sizing data for these two designs (because the path following design is so similar to the line search one pictures and plots of it have been omitted).

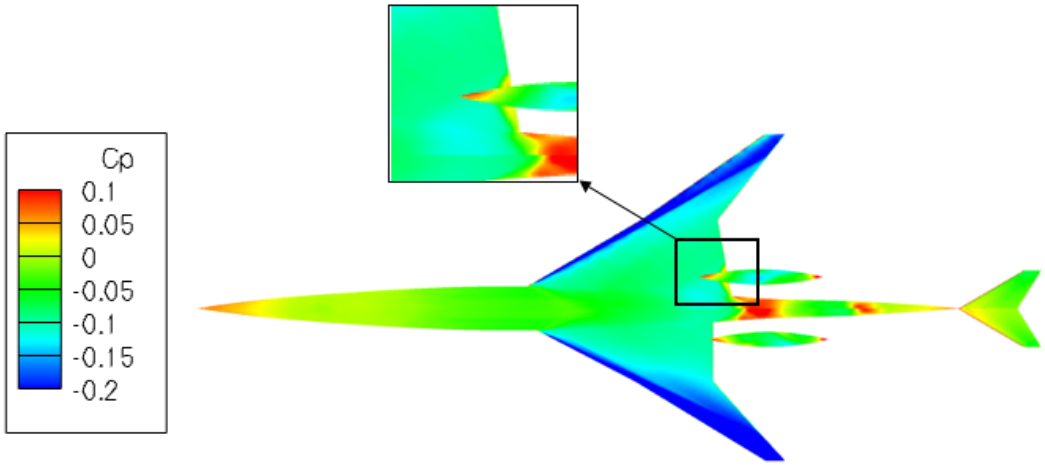


Figure 4.9. Comparison of preliminary (bottom) and optimized (top) SSA configurations.

	Preliminary	Optimized
MTOW (lb)	124195	143976
S_{gross} (ft^2)	1392	1725
AR_{gross}	3.15	3.31
Inner/outer leading edge sweeps (deg)	62.5/58.74	59.58/55.41
S_H/S_{total}	0.1	0.08
S.L.S. Thrust (lb)	31028	32996
Initial/final cruise altitudes (ft)	45395/55435	41292/52511

Table 4.7. Comparison of preliminary and optimized SSA configurations.

Due to the significantly higher drag values predicted by CART3D, the engines increased in size to improve D/T. With bigger engines and lower (L/D) values, significantly more fuel was needed for the aircraft to satisfy the range requirement. Thus the weight went up significantly as did wing area, which had to increase to satisfy field length requirements. In an attempt to reduce drag, to satisfy D/T, the optimizer reduced the cruise altitude to values lower than one would expect. In addition to this, Fig. 4.9 shows that shocks from the nacelle impinge the wing's trailing edge on the optimized design. This does not seem optimal (though this may be necessary in order to satisfy the x/c gear constraint), and suggests that a better solution may exists. Despite these drawbacks, however, a feasible design was found, which represents progress. More work is clearly needed, but the method shows promise.

5 Multifidelity Expected Improvement

In this chapter, we present an extension of the well known two-stage optimization technique known as maximization of expected improvement [see 1, 2]. First, using two models of differing fidelity we formulate a two-fidelity expected improvement technique for bound-constrained optimization. Next the two-fidelity method is compared to the standard expected improvement method on some test problems. Finally, an alternative to exterior penalty methods for handling nonlinear inequality constraints is formulated. For a brief review of surrogate modeling (in this case Kriging) and basic expected improvement please see the work by Rajnarayan et al. [18].

5.0.1 Two-Fidelity Expected Improvement

Assume we have a high-fidelity simulation G, which has high computational cost and high accuracy, and a low-fidelity simulation G_l, which has negligible computational cost compared to G, but also suffers from poor accuracy. Even if the individual models G and G_l themselves are not smooth, the *difference* between these two models is likely to be a smooth, well-behaved function Eldred and Dunlavy [19]. Therefore, we fit a GP regression to the difference between the two models, which can be thought of as a correction to the low-fidelity model. Such a combination of multifidelity models is described by Eldred and Dunlavy [19] in the context of additive and multiplicative surrogate corrections, and is used by Choi et al. [20] in design optimization of a supersonic business jet. These corrected surrogates are then used in a trust-region optimization. To use the vocabulary in the literature, we advocate using a global additive surrogate correction on the low-fidelity model to perform global search using a two-stage method.

Denote the correction values obtained from the GP by $G_{\text{diff}}(x)$, and our approximation by $\tilde{G}(x)$. Mathematically,

$$\tilde{G}(x) = G_l(x) + G_{\text{diff}}(x). \tag{5.1}$$

Thus, each evaluation of the approximate model in Eq. 5.1 will involve evaluating both the low-fidelity model $G_l(x)$ and the GP regression $G_{\text{diff}}(x)$ for the correction values. Note that this approximation agrees exactly with the high-fidelity simulation at the points in $\mathcal{D}$. In reality, the GP actually yields a posterior probability distribution on the values $G_{\text{diff}}(x)$. This distribution completely determines the posterior probability distribution on the values $\tilde{G}(x)$. Under the assumptions of a GP, this posterior distribution is a Gaussian, with mean $\hat{g}(x) = \tilde{G}(x)$ and variance $\sigma(x)$. The method of expected improvement as well as other global optimization techniques are still applicable, but with one significant difference: the objective functions for the auxiliary problem are no longer smooth. The branch and bound method from EGO can no longer be directly applied, and neither can the multi-start gradient descent methods described by Jones [1]. This is depicted in the Fig. 5.1 below. Fig. 5.1.a. shows the low-fidelity model and a small number of high-fidelity samples. Fig. 5.1.b. shows the GP fit to the difference, or surrogate correction. Fig. 5.1.c. shows the corrected approximation that matches the high-fidelity data exactly at existing designs, and Fig. 5.1.d. shows the expected improvement surface for this problem. An important observation about this curve is the fact that is not smooth! Now we can invoke the technique of iterated maximization of expected improvement, exactly as described by Jones et al. [2], with the only difference being that we need to use a gradient-free method to solve the auxiliary problem. In this paper, we use existing gradient-free techniques previously described by [see 21, 22].

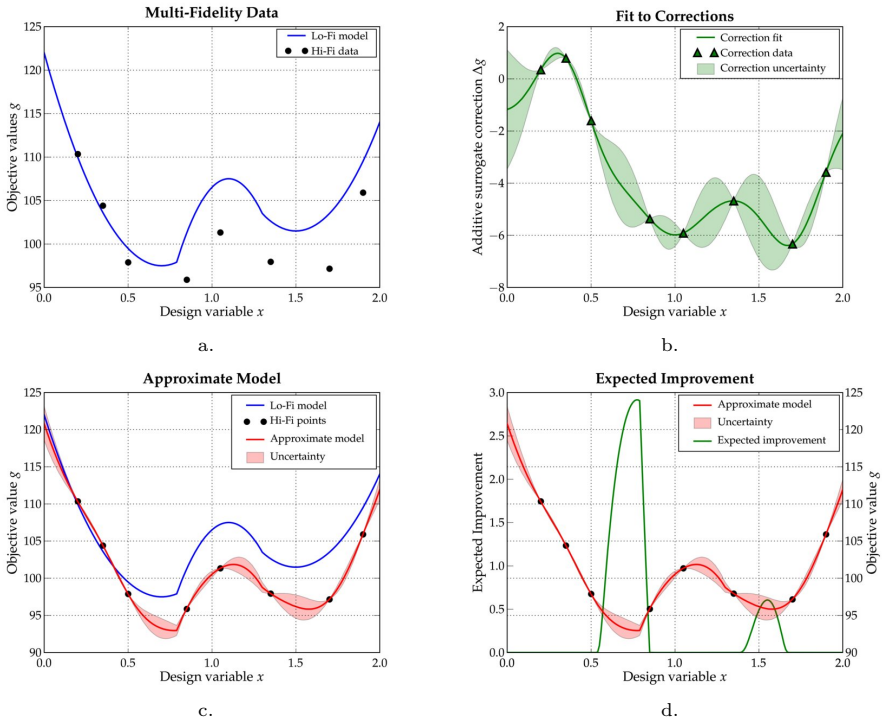


Figure 5.1. Multifidelity Approximation and Auxiliary Optimization Problem.

5.0.2 Two-Fidelity Expected-Improvement Test Problems

We test the technique described above on a few analytic test problems. The problems chosen are the 3- and 6-dimensional Hartman problems, described by Dixon and Szegö [23], and the 4-dimensional Woods problem, given by

$$\begin{aligned} G(x) &= 100(x_2 - x_1)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &\quad +10.1[(1 - x_2)^2 + (1 - x_4)^2] + 19.8(1 - x_2)(1 - x_4),\ x \in [-1, 1]^4. \end{aligned} \tag{5.2}$$

We constructed 'low-fidelity' models for these functions by adding a simple analytical 'error' function to them. In the case of the Hartman problems, the low-fidelity model used was

$$G_l(x) = G(x) + A \sin\left(\omega \sum_{i=1}^{n} (x_i - a)^2\right). \tag{5.3}$$

The error in this case corresponds to a radially undulating error symmetric about the point a, with 'amplitude' A and 'frequency' ω. For both Hartman3 and Hartman6, $a = 0.4, A = 0.5$. For Hartman3, $\omega = 10$, whereas for Hartman6, we used $\omega = 25$. A plot of the 1-dimensional version of this undulating error is shown in Fig. 5.2.

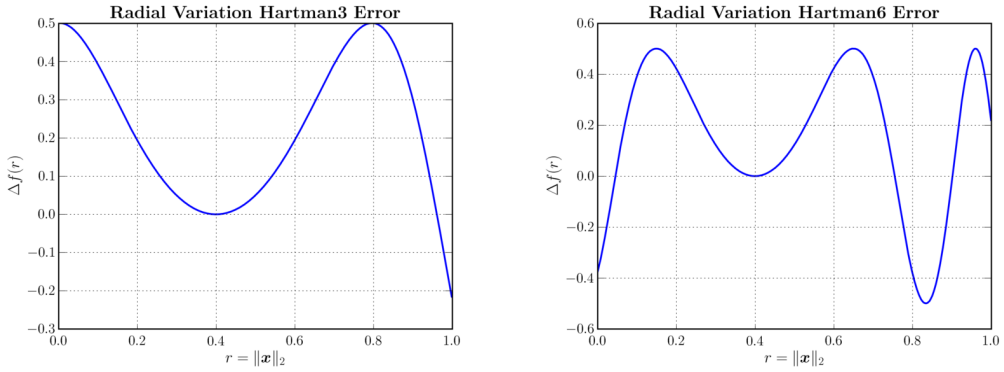
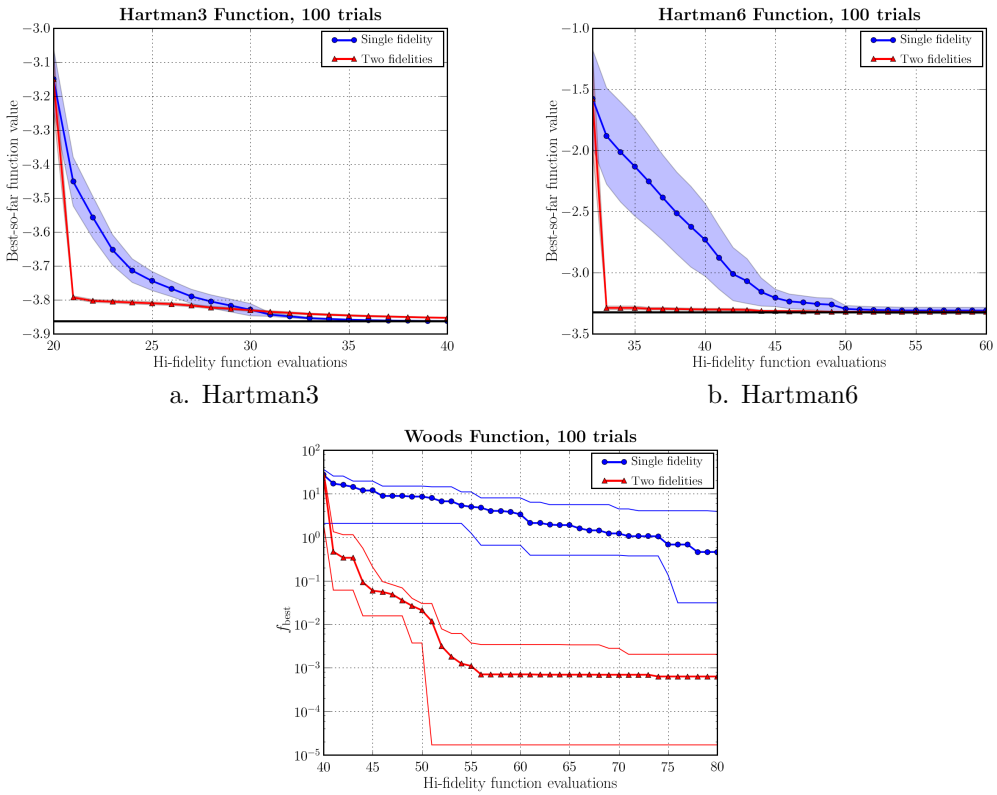


Figure 5.2. 1-D version of error applied to Hartman functions.

For the Woods problem, we used a relatively simple error model, given by the quadratic $10\sum_{i=1}^{4}(x_i - 0.5)^2$. In all cases, we ensured that the low-fidelity model did not share the optima of the high-fidelity model, thereby ensuring that direct optimization of this low-fidelity model would be misleading.

We now apply the method of maximization of expected improvement using the difference between the analyses of these two fidelities. To compare multiple runs of the algorithm, we specify a sample budget that determines termination. For Hartman3, we start with 20 initial samples and allow 20 additional high-fidelity evaluations. For Hartman6, we start with 32 initial samples and allow 60 total high-fidelity evaluations. For the Woods problem, we start with 40 initial samples and take 40 additional samples through the optimization process. In contrast, the maximum number of low-fidelity evaluations was fixed at 4000 for all problems.



a. Hartman3

b. Hartman6

c. Woods

Figure 5.3. Performance of two-stage expected improvement maximization on academic test problems.

Shown in Fig. 5.3 are summaries of 100 trials of the expected improvement maximization algorithm on these academic test problems. To remove effects of the initial sample set, we used the same set of initial Latin Hypercube samples for both algorithms on a given trial. The results are plotted by showing the best design found at each stage of the algorithm. The solid line represents the sample average performance, and the shaded regions represent the 95% confidence interval for this average. In the case of the Woods problem, a semilog plot is used, so we cannot plot the confidence intervals as before. Instead, we plot median performance, 10^{th} and 90^{th} percentiles of performance, as well as best and worst performance. The solid lines represent the true optima of the Hartman problems.

We can see that the presence of the low-fidelity model greatly reduces the variance of the algorithm from trial to trial. In other words, the dependence on the initial set of samples, and consequently on the data-fit constructed from them, is reduced. The discovery of optima is also much quicker, owing to the reduced uncertainty in the multifidelity function approximation. In the case of the Hartman problems, we note that the performance in the single-fidelity case, is comparable to that claimed by Jones et al. [2], indicating that our auxiliary optimizer is indeed functioning properly.

5.0.3 Constrained Multifidelity Optimization

In Sec. 5.0.1 we described the use of two-fidelity expected improvement maximization techniques for bound-constrained optimization. In this section, we review the application of a similar technique Schonlau et al. [24] to general nonlinear inequality constrained optimiza-

tion. The concept of expected improvement can be extended to problems with constraints as follows. A given point x_{test} has a continuum of possibilities for its objective values as well as constraints. If we use, say, Kriging fits for the objective function and constraints, we now have posterior probability distributions over all these functions. Now, note that this point yields an improvement over current designs if the true objective value is lower than the best objective value seen so far *and* it satisfies all of the constraints. In addition, we may consider the posterior probability distributions of objective function and constraints as independent: given the design variables x, the objective values and constraint values are indeed conditionally independent. Now, the computation of (constrained) expected improvement is an integral over all possible objective function values and constraint values, but the aforementioned independence enables us to split the integral into the product of two integrals, which turn out to be the original (unconstrained) expected improvement integral and the probability of feasibility of this new design. The derivation is as follows. Suppose the problem at hand is

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & g_i(x) \leq 0,\ i = 1, \ldots, m. \end{array} \tag{5.4}$$

The improvement of some set of values $(f, g_1, \ldots, g_m)$ is given by

$$I(f, g_1, \ldots, g_m) = \begin{cases} \max(f_{\text{best}} - f, 0), & g_i \leq 0,\ i = 1, \ldots, m, \\ 0, & \text{otherwise.} \end{cases} \tag{5.5}$$

Then, the expected improvement integral breaks down as follows

$$\begin{aligned} \mathbb{E}[I(x)] &= \int df\, dg_1 \ldots dg_m\, p(f, g_1, \ldots, g_m) I(f, g_1, \ldots, g_m), \\ &= \int df\, dg_1 \ldots dg_m\, p(f) \prod_{i=1}^{m} p(g_i) I(f, g_1, \ldots, g_m), \\ &= \left[\int_{-\infty}^{\infty} df\, I(f)\right] \prod_{i=1}^{m} \int_{-\infty}^{0} dg_i\, p(g_i) \end{aligned} \tag{5.6}$$

In practice, if the feasibility probability is generally very low, the second term in Eq. 5.6 is very small everywhere, leading to problems of numerical accuracy. We have found that modifying the feasibility probability above to reflect the probability of improving the feasibility of each constraint ameliorates most of these numerical problems, and therefore this is the approach we follow. In other words, the constrained expected improvement is given by

$$\mathbb{E}[I(x)] = \left[\int_{-\infty}^{\infty} df\, I(f)\right] \prod_{i=1}^{m} \int_{-\infty}^{g_{i_{\text{best}}}} dg_i\, p(g_i). \tag{5.7}$$

There is reason to believe that this multiplicative formulation for constraints will have poor numerical performance as the number of constraints grows. In addition, this method completely ignores the objective value in infeasible regions. This has been known to cause slow convergence. We are presently comparing this technique with the well-known exterior penalty methods for constraints.

6 Multiobjective Approaches to Surrogate-Based Optimization

This work was in collaboration with Geoffrey Bower, Ph. D. candidate at Stanford University, who is funded under NASA's Subsonic Fixed-Wing Program NRA titled *Multidisciplinary Optimization Techniques For Efficient Subsonic Aircraft Design*, a cooperative research effort between MIT, Stanford, Boeing and Purdue.

There are a number of drawbacks to the method described in Chapter 5. At each iteration, expected improvement chooses a single 'best point' for evaluation using the high fidelity analysis. It does this by synthesizing this figure of merit, expected improvement, from the surrogate prediction and uncertainty. It therefore relies heavily on the surrogate model and uncertainty being predicted well, and is closely linked with the assumptions made by them, such as Gaussianness. It is also known that the fit is a drastic approximation at the early stages of the algorithm [25].

In addition, we may wish evaluate to evaluate more than one high-fidelity design at each iteration. One such algorithm is presented by Schonlau et al. [24]. The motivation for picking multiple points is manifold: we may wish to exploit parallel computation for the high-fidelity simulation; or the auxiliary optimization involving the low-fidelity model may be sufficiently resource intensive that we want to restrict the number of these optimizations; or from a theoretical viewpoint, it may be beneficial to maintain a balanced portfolio of designs that trade performance and risk, and are thereby rendered relatively insensitive to modeling errors and assumptions.

In order to address these points, we propose an algorithm that formulates the auxiliary optimization as a multiobjective problem, one that balances uncertainty and performance, and computes an entire population of designs that form the risk-performance pareto frontier. This method is amenable to both multifidelity implementations and parallelization, and therefore seems quite promising. The following sections discuss the choice of objectives for this multiobjective approach, the selection of multiple points from the pareto front, and the results of some tests on analytic problems.

6.1 Trading Performance and Risk

During every iteration, a two-stage, surrogate-based optimization algorithm attempts to pick promising designs, based on both the predicted performance, and the uncertainty or risk in that prediction. This is indeed akin to balancing a stock portfolio: it is desirable to maintain a balance between high-risk, high-return stocks and low-risk, low-return stocks, and such an approach is known to perform well in practice. In the context of surrogate-based optimization, we first need to investigate some reasonable indicators of performance and risk.

In an ideal world, we would like excellent returns at zero risk, or, in the optimization context, very low objective values with almost no uncertainty. In other words, if we are almost certain of the location of the global optimum, and we haven't yet evaluated that point, we must do so forthwith. From this line of reasoning, two possible objectives to be simultaneously minimized are the mean prediction and a measure of the uncertainty such as the variance. More generally, we want one objective to be an optimistic estimate of performance that can only be achieved at the cost of hurting another pessimistic estimate,

which forms the second objective. In addition to these considerations, we seek points that are *informative*. In other words, points where the uncertainty is large may, in general, provide information about the location of minima. Intuitively, a good set of objectives will seek the competing goals of minimizing objective cost, reducing risk, and maximizing information gain. These considerations together have led to the set of objectives listed in Table 6.1. For now, we consider only two-objective problems for reasons outlined in subsequent discussions, but a better formulation may involve three or more objectives.

Type	Objective 1	Objective 2
I	μ	σ
II	$\mu - k\sigma$	σ
III	μ	$\mu - k\sigma$
IV	μ	$-\sigma$
V	$\mu - k\sigma$	$\mu + k\sigma$

Table 6.1. Possible objectives for multiobjective surrogate-based optimization.

A brief motivation for each of these formulations is presented below.

- Type I: Seeks good mean performance and small uncertainty. Among points with good mean performance, this metric prefers lower uncertainty.
- Type II: Seeks good optimistic performance and small uncertainty. Among points with good optimistic performance, this metric prefers lower uncertainty.
- Type III: Seeks good optimistic performance and good mean performance. Among points with good optimistic performance, this metric prefers good mean performance.
- Type IV: Seeks good mean performance and large uncertainty. Among points that have good performance, this metric prefers higher uncertainty as a surrogate for information. This is the most exploratory of the objective pairs.
- Type V: Seeks both good optimistic performance and good pessimistic performance. Among points that have good optimistic performance, this metric prefers those with better pessimistic performance.

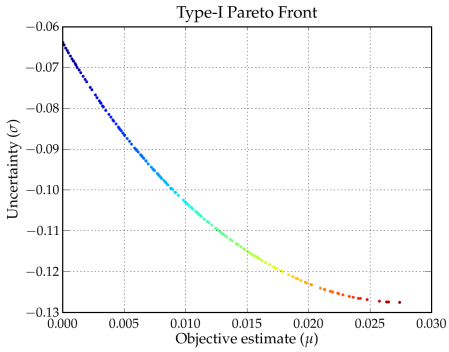
At every iteration of the algorithm, we perform a multiobjective minimization trading risk and performance, and this yields a pareto front of risk and performance. Among designs on this front, one can only trade one objective for another; in other words, no design on this front offers both lower risk *and* better performance than any other design on the front. For the Type I objectives, if we assume that the pareto fronts are convex, then we can use a tradeoff parameter and minimize $\mu - k\sigma$ for a wide range of values k. This is indeed suggested by methods in the literature [26], and has been shown by Jones [1] to be identical to maximizing probability of improvement for a wide range of k. In general, though, we expect the pareto front to be nonconvex, and needs to be computed using a true multiobjective optimization approach. We form our 'balanced portfolio' of designs by picking a few designs from this pareto surface. This overall algorithm is depicted in Algorithm 3. To further illustrate the idea, we consider the following 1-dimensional function

$$f(x) = x^2 + 0.2(1 - |x|)\sin(4\pi x + \pi/3) \text{ for } -1 \leq x \leq 1.$$

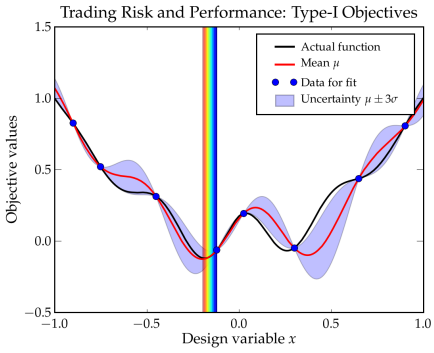
A few samples of this function, and the resulting fit are shown in Fig. 6.1(b). The pareto front corresponding to type-I objectives is shown in Fig. 6.1(a). The correspondence between designs on the front and their location in the design space is achieved by color-coding them.

Algorithm 3 Overview of multiobjective optimization generating multiple samples per iteration.

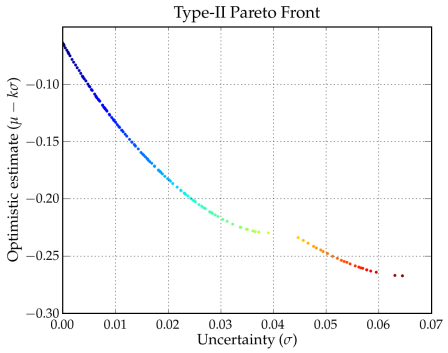
1: Initialize surrogate model
2: **repeat**
3: Perform multiobjective optimization (see Table 6.1 for details of the objectives themselves.)
4: Select promising designs from the pareto front
5: Evaluate these designs using the high-fidelity analysis
6: Use new designs to update surrogate model
7: **until** convergence



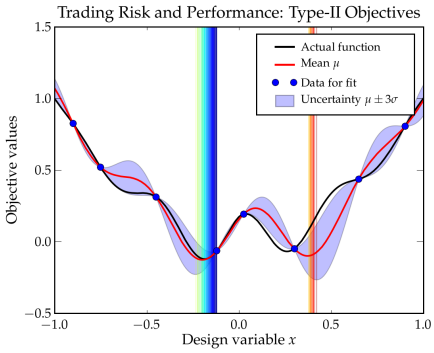
(a) Pareto front for type-I objectives



(b) Type-I objectives, μ and σ.

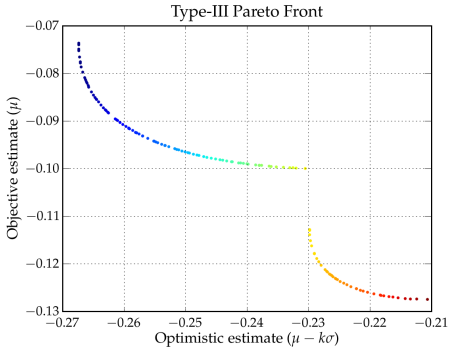


(c) Pareto front for type-II objectives

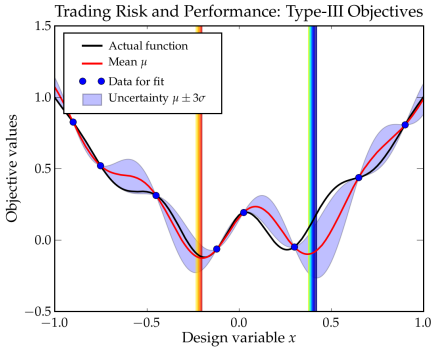


(d) Type II objectives, $\mu - k\sigma$ and σ.

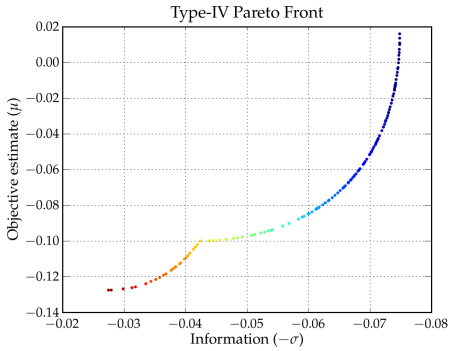
Figure 6.1. The risk-performance pareto front and corresponding designs in the design space.



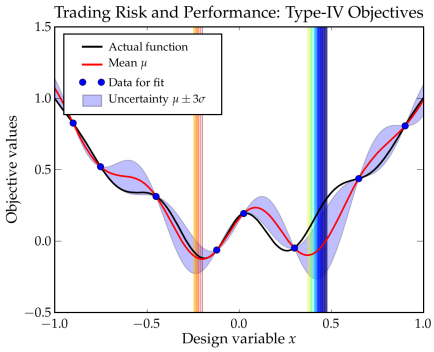
(a) Pareto front for type-III objectives



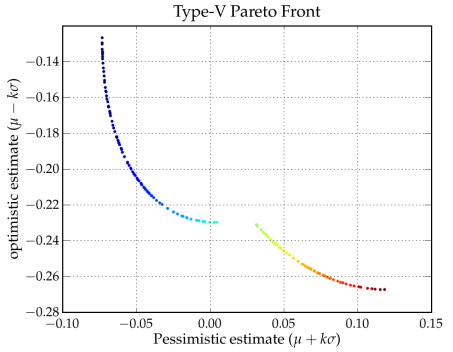
(b) Type III objectives, μ and $\mu - k\sigma$.



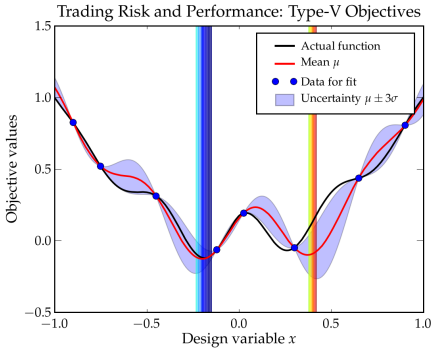
(c) Pareto front for type-IV objectives



(d) Type IV objectives, μ and $-\sigma$.



(e) Pareto front for type-V objectives



(f) Type V objectives, $\mu - k\sigma$ and $\mu + k\sigma$.

Figure 6.2. The risk-performance pareto front and corresponding designs in the design space.

Thus, the low-risk designs (dark blue) can be seen to be very close to existing designs, whereas the high-risk, high-performance designs (red) are further from existing designs. This is done for types III-V and is shown in Fig. 6.2

6.2 Selecting One or More Designs from the Pareto Front

Once the pareto front has been identified, we need an algorithm to pick one or more promising designs from it, for evaluation using the high-fidelity analysis. Typically, the pareto surface is populated by hundreds of points, making it impractical to evaluate all of them. We want to select a number large enough to 'cover' all interesting regions of this front, while at the same time not picking designs that are too similar. This section proposes some techniques to pick a fixed or variable number of points for evaluation, based on some reasonable heuristics.

A very simple approach is to select a fixed number of designs from the front, equally spaced on it. This is indeed one of the approaches we attempt. Nevertheless, when viewed in the space of design variables, it is often seen that points on the pareto front tend to 'bunch up' in a few distinct regions, as seen in Fig. 6.1. This suggests the following approach: cluster points on the front based on the design variable values. This yields a few distinct clusters, from each of which one point can be chosen for evaluation. This approach has been suggested by Jones [1] in the case of maximizing probability of improvement for a range of targets. In our experiments, we implemented Fuzzy c-means[27] and k-mediods[17] methods. The first method is a well-known clustering algorithm, and is the *de facto* standard in many machine learning applications. Fuzzy c-means clustering allows points to belong to each cluster with a varying degree of belonging. The degree of belonging is inversely proportional to the distance between the point and the cluster center. Thus points very close to a cluster center will have a high degree of belonging to that cluster and a very small degree of belonging to other clusters. The algorithm determines cluster centers by minimizing the sum of the distances of the data points to the cluster centers, where distances are weighted by the degree of belonging. The resulting cluster centers are not collocated with any of the data points and thus are not necessarily on the pareto front. In the second method, k-medoids, each data point belongs solely to the cluster with the closest center. As with Fuzzy c-means, this algorithm determines cluster centers by minimizing the sum of the distances between data points and cluster centers. However, unlike Fuzzy c-means, k-medoids requires that each cluster center be a member of the data set. This guarantees that the points sampled lie on the Pareto front. We are presently experimenting with these clustering approaches to pick multiple points from the pareto front.

One of the motivations for clustering is that we want to avoid redundant high-fidelity evaluations: points very close to each other in the design space will tend to have very similar performance, so little is gained by evaluating many points close together. Clustering provides one way to find distinct points, but a single cluster may be large enough that it may be worthwhile evaluating multiple points in it. One way to work around this problem is to specify a cluster size, but it is difficult to pick this size *a priori*. Instead, we propose using the information-theoretic measure of entropy to pick informative samples. In some sense, the most informative samples are those that result in the greatest drop in entropy upon evaluation. In other words, let Y be the unknown objective value of some design, and Z be the observed objective value upon evaluation, and let $H(\cdot)$ be the entropy of a random variable. We want to find designs with a value of $H(Y) - H(Y \mid Z)$ above some threshold. In the zero-noise case, Y and Z are the same, implying that $H(Y \mid Z) = 0$. Therefore we seek $H(Y)$ above a certain threshold. It turns out that in the case of Gaussian random

variables, such as in Kriging, entropy is directly related to variance:

$$Y \sim \mathcal{N}(\mu, \sigma) \Rightarrow H(Y) = \ln(\sigma\sqrt{2\pi e}).$$

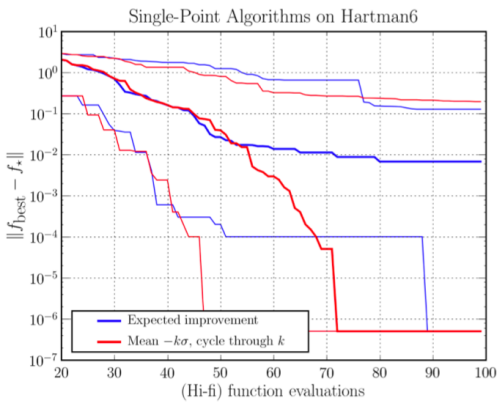
This indicates that we can just look for designs whose posterior variance is above a certain threshold. It turns out that this also allows us to specify the threshold in relative terms, as a fraction of the maximum uncertainty (prior variance) for the Kriging model.

Following these techniques, we propose the following adaptive method to pick mutually non-redundant designs from the front: we sort the designs in increasing order of risk. Starting from the low-risk end of the front, we seek the first design that exceeds an information threshold. This threshold can be specified in terms of relative uncertainty or correlation, so we can select a value in a reasonable manner. We proceed by adding designs that exceed this threshold to a set of candidate designs. Every time we do this, we update the uncertainty predictor alone. Whenever a design is added, other designs on the front that are nearby in the design space become redundant, and thereby do not exceed the information threshold. The search continues until all designs on the pareto front have been tested this way. If no designs exceed the threshold, then the point with the maximum uncertainty is selected. This is somewhat similar to a single-objective, multipoint approach proposed by Schonlau et al. [24], but in that approach, assumptions were made about the true performance of designs not yet evaluated. Besides, this approach requires multiple sequential auxiliary optimizations.

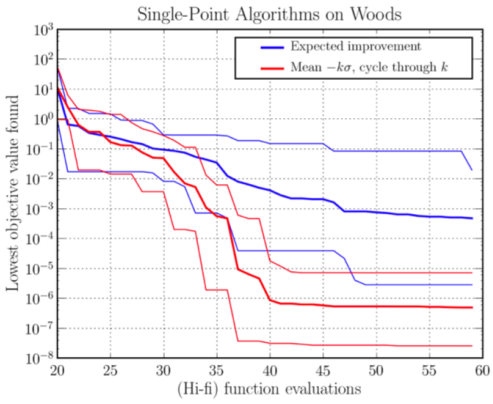
6.3 Preliminary Results

In this section we present the results of applying some of these algorithms to analytic test problems. Specifically, we compare the performance of the following 10 algorithms. Two methods that pick a single point per iteration — maximizing expected improvement and minimizing a statistical lower bound $\mu - k\sigma$ with k cyclically varied with each iteration; four multipoint algorithms that pick a fixed number of designs per iteration (this number was fixed at three in our experiments); and four multipoint algorithms that adaptively pick the number of samples per iteration based on relative uncertainty as described above. The multipoint algorithms used four of the objective pairs listed in Table 6.1, types I, II, III, and V. In our experiments, for types II, III and V, we used $k = 2$.

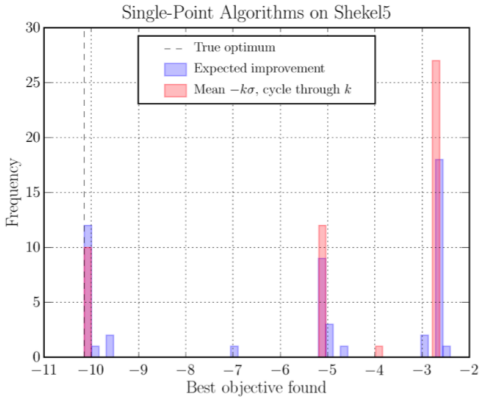
The three test functions used are: the 4-dimensional Woods function that exhibits poor scaling, the 4-dimensional Shekel-5 function with multiple local minima separated by relatively flat regions, and the 6-dimensional Hartman-6 function with multiple local minima. For the Shekel and Hartman problems, we used just a single-fidelity model, whereas for the Woods problem, we used a two-fidelity model by constructing a low-fidelity approximation using a radially symmetric nonlinear sinusoid, similar to that described in Sec. 5.0.2. We ran 50 trials for each of the ten algorithms on each problem. The initial surrogate for each trial was constructed using 20 Latin Hypercube samples, and to ensure a fair comparison, the same set of initial samples was provided to each algorithm on a given trial. Therefore, performance differences are due only to the choices made by the algorithm.



(a) Hartman-6

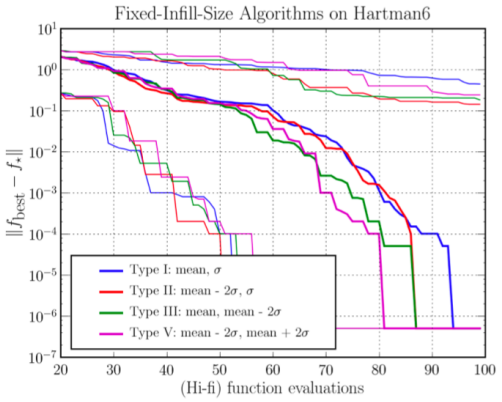


(b) Woods

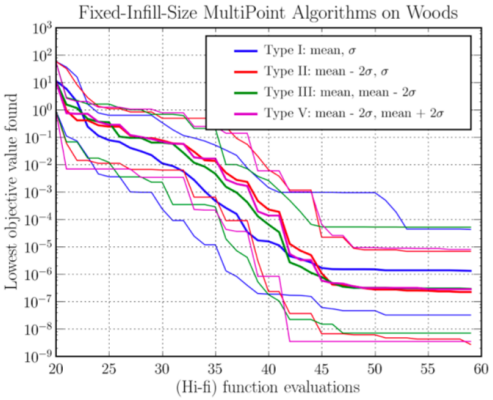


(c) Shekel-5

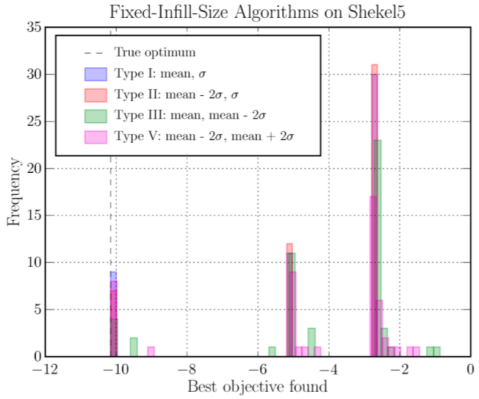
Figure 6.3. Best, worst and median performance for single-point algorithms. Note that minimizing a series of statistical lower bounds performs quite well, and that maximizing expected improvement exhibits slow convergence.



(a) Hartman-6

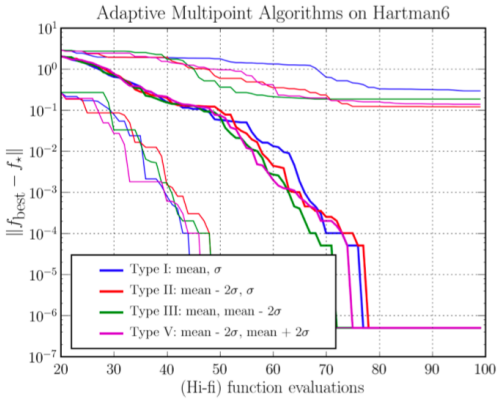


(b) Woods

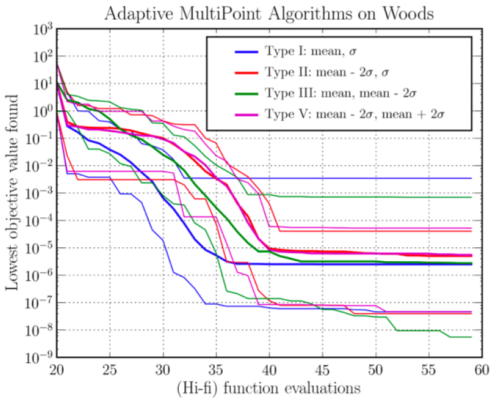


(c) Shekel-5

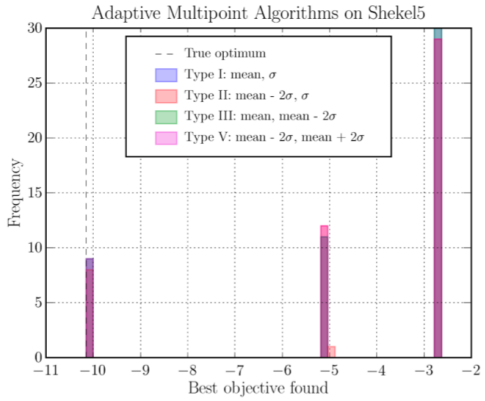
Figure 6.4. Best, worst and median performance for fixed-infill-size multipoint infill sample algorithms. The three points were selected to lie at the one-third, two-thirds, and maximum risk points of the front. The lowest risk point was never chosen because, at least for types I and III, it is always one of the already-evaluated designs.



(a) Hartman-6

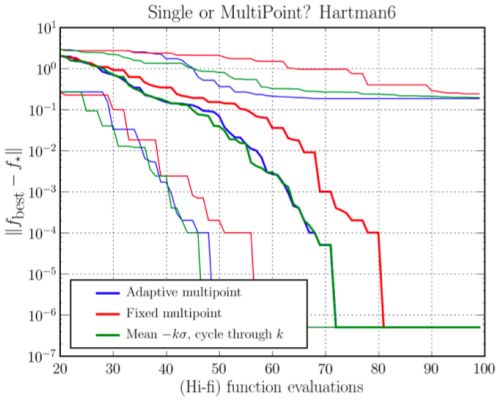


(b) Woods

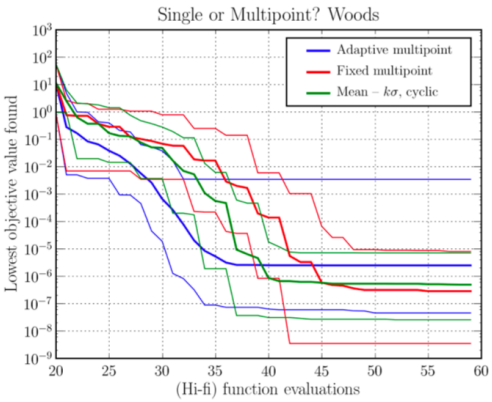


(c) Shekel-5

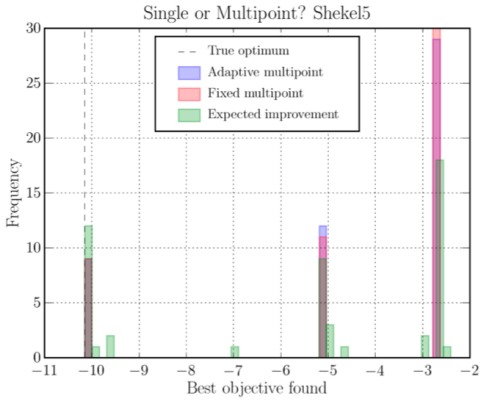
Figure 6.5. Best, worst and median performance for adaptive multipoint algorithms. Note that the performance is relatively insensitive to the choice of objectives, both in this case and the previous, fixed-infill-size case.



(a) Hartman-6



(b) Woods



(c) Shekel-5

Figure 6.6. Comparison of the best single-point algorithm, the best fixed-infill-size multipoint algorithm, and best adaptive multipoint algorithm. Note that the adaptive algorithms perform no worse than the best single-point algorithms, and at times do much better. As will be shown in a subsequent section, evaluating multiple designs per iteration can result in significant savings in wall-clock time when the high-fidelity analyses are truly time-consuming.

7 Bodies of Revolution with Minimum Wave Drag

As an intermediate step between analytic test problems and full aircraft design we minimize the wave drag on axissymmetric bodies of revolution using the multifidelity techniques described in Chapters 5 and 6. The body has a fixed length and either a fixed cross-sectional area or a fixed enclosed volume. The axial radius distribution is varied in order to minimize drag, subject to the constraint on cross-sectional area or enclosed volume. If the flow is assumed inviscid, then the only source of drag is wave drag. Further, assume that the shock waves created by the body are of negligible strength. Finally, the body is assumed to be sufficiently slender that it is valid to apply the flow-tangency boundary condition at the flow-aligned axis of revolution, rather than at the body surface. These assumptions lead to the area rule method and the bodies with minimum wave drag are the well-known Sears-Haack bodies.

For bodies with high fineness ratios predictions by A502 and CART3D agree well with the predictions made by the area rule method discussed in Sec. 2.2. As the fineness ratio decreases, however, the difference between wave drag predictions grows. This is to be expected: applying the boundary condition to the axis of revolution is less valid for bodies with low fineness ratios. At a fineness ratio of 20, the difference between the wave drag computed with the area rule and the higher fidelity methods is roughly 4%. But at a fineness ratio of 10, the difference has grown to 9%. This demonstrates the need for a multifidelity approach in such a problem.

7.0.1 Parametrization and Problem Formulation

The optimization problem can be posed in the following way.

$$\begin{array}{ll} \min & C_{d_w}(\delta r_i) \\ \text{w.r.t.} & \delta r_i \\ \text{s.t.} & c(\delta r_i) = c_0. \end{array} \tag{7.1}$$

Here the design variables, δr_i, are deviations from a nominal design, in this case, a parabolic radius distribution, specified at a given set of 13 locations along the axis of revolution. The radius for any value of x is found using an Akima spline fit to the specified radius distribution. Depending on the problem statement, either the maximum radius or the enclosed volume is constrained.

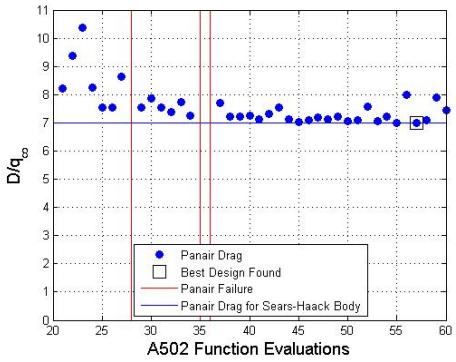
The optimization problem in Eq. 7.1 was originally stated using an equality constraint on either total volume or maximum radius, but using an inequality constraint does not alter the solution. Writing the constraint as $c(\delta r_i) \geq 0$ does not affect the solution because, for fixed body length, increasing maximum diameter or enclosed volume will always increase the minimum drag. So we pose the problem as follows.

$$\begin{array}{ll} \min & C_{d_w}(\delta r_i) \\ \text{w.r.t.} & \delta r_i, \\ \text{s.t.} & c(\delta r_i) \geq c_0. \end{array} \tag{7.2}$$

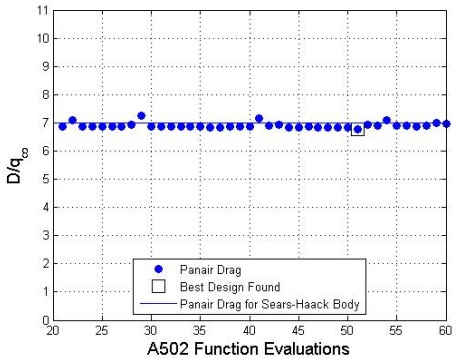
7.0.2 Results

The multifidelity techniques described in Chapters 5 and 6 were used to solve this problem with the area rule method as the low-fidelity analysis. Initially A502, as opposed to CART3D, was chosen as the high-fidelity analysis in the interests of low computational costs during the development of the multifidelity expected improvement method. Once the methods were working the high-fidelity analysis was switched to CART3D. The radii at the nose and tail are fixed to zero regardless of the constraint, while the radius at the middle of the body is fixed when the constraint is on maximum radius. The following results are for a body with a length of 100 ft, a maximum radius of 5 ft, or a total enclosed volume of 4200 ft^3, and $M_\infty = 1.5$

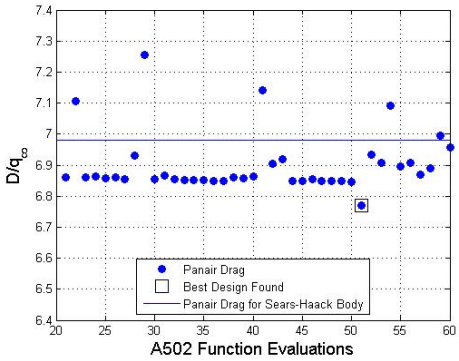
Figs. 7.1.a and 7.1.b show the progress of the single- and two-fidelity expected improvement methods, discussed in Sec. 5.0.2, for a representative run of each. Both methods were initialized with the same 20 samples drawn from a Latin Hypercube over the design space. A total of 40 additional design evaluations were allowed.



(a) Single fidelity expected improvement



(b) Two fidelity expected improvement



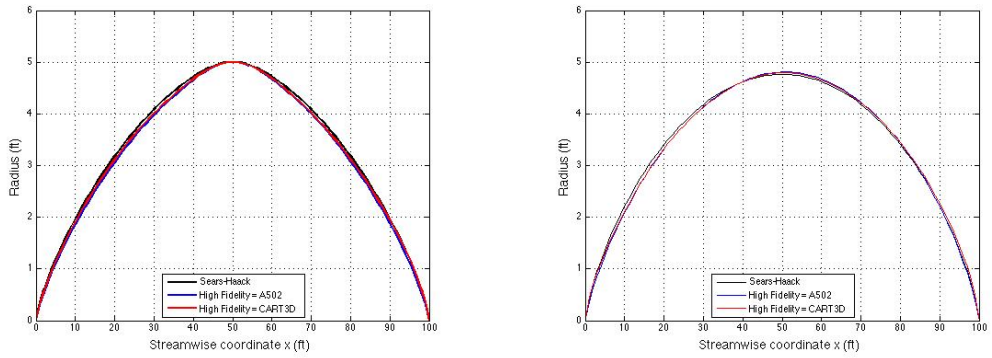
(c) Two fidelity expected improvement (note y-axis scale)

Figure 7.1. Progress of single-fidelity and two-fidelity expected improvement methods applied to wave drag minimization of of an axisymmetric body of revolution with fixed maximum radius.

Fig. 7.1 shows that the single-fidelity method explores relatively undesirable regions of the design space and evaluates many designs with relatively large drag. In fact, A502 could not evaluate three out of these 40 designs (these failures are denoted by vertical red

lines) due to the presence of super-inclined panels. This happens because, with 20 samples, the GP regression generates a rather poor model of a 12-dimensional design space, thereby prompting the algorithm to explore undesirable regions. In contrast, the two-fidelity method is guided toward good regions by the low-fidelity analysis. In fact, in the first two iterations it finds a design with a lower drag than the linear-theory optimum (Sears-Haack body). This happens because the high- and low-fidelity models are often in good agreement and thus 20 samples suffice to produce a reasonable model of the difference between them, even in 12 dimensions.

Having tested this optimization method, we replaced A502, which may be considered a low-fidelity analysis for many applications, with the Euler code CART3D. We than ran the two-fidelity expected improvement method, with no initial samples, for 40 iterations (40 high fidelity analyses) with CART3D as the high fidelity method. A comparison of the best designs found is shown in Fig. 7.2. For purposes of comparison they are plotted against the solution based on classic linear theory (Sears-Haack bodies). Table. 7.1 shows the drag of the bodies in Fig. 7.2 computed by each method.



(a) Maximum radius

(b) Total enclosed volume

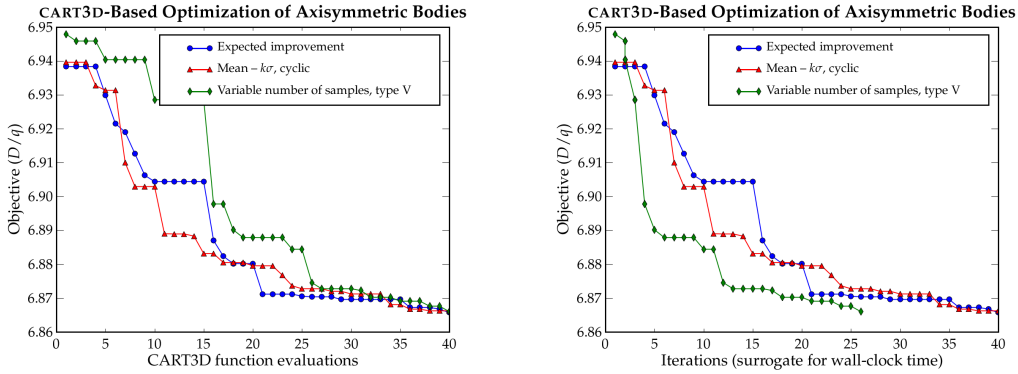
Figure 7.2. Radius distributions of optimized bodies found using the multifidelity expected improvement method with A502 or CART3D as the high-fidelity method and the area rule as the low-fidelity method.

Body	**Constraint**	**Area Rule** D/q_∞	**A502** D/q_∞	**CART3D** D/q_∞
Sears-Haack	Max radius	7.75157	6.9815	6.9354
Optimum found with A502	Max radius	7.8765	6.8536	6.88532
Optimum found with CART3D	Max radius	7.83137	6.89041	6.87266
Sears-Haack	Volume	7.18718	6.84677	6.58922
Optimum found with A502	Volume	7.24348	6.78318	6.64468
Optimum found with CART3D	Volume	7.30531	6.95656	6.58878

Table 7.1. Drag of the bodies shown in Fig. 7.2 computed with CART3D, A502, and the area rule method.

We then applied the adaptive multipoint search method, described in Chapter 6, to this problem and compared it with expected improvement and $\mu - k\sigma$ with k cyclically varied. Fig. 7.3a plots the improvement in drag against the number of high-fidelity function evaluations for a single run of each method, and it would appear that all the methods perform similarly. However, Fig. 7.3b plots the improvement in drag against iteration, which is representative of clock time assuming high fidelity analyses can be done in parallel,

which shows the reduction in overall time that the mulitpoint search method is capable of achieving.



(a) (b)

Figure 7.3. Comparison of adaptive multipoint search, expected improvement and $\mu - k\sigma$ with k cyclically varied for minimization of wave drag with fixed maximum radius. Note the range of the $y - axis$, which was selected so the differences in performance are clear even with the small variation in drag.

8 Drag Minimization of a Wing-Body Configuration

In this chapter the multifidelity, multiobjective optimization approach described in Chapter 6 is applied to the optimization of a wing-body configuration based on a supersonic business jet. The baseline design is described by Wintzer and Sturdza [28] and is shown in Fig. 8.1. According to Wintzer and Sturdza [28], this design was optimized using analyses for conceptual design of aircraft. The aircraft was required to satisfy several operational requirements including cruise performance, low-speed performance, stability and control, cabin geometry restrictions, payload requirements, and others. The design variables defined the overall planform geometry and a few sectional properties of the fuselage and wings.

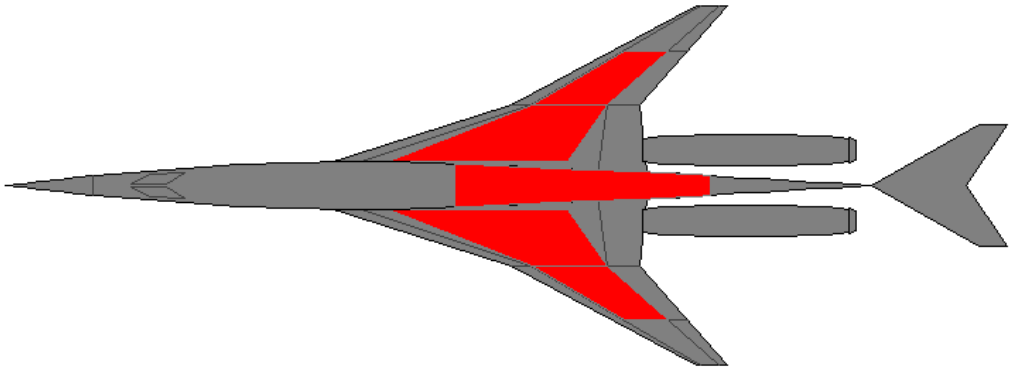


Figure 8.1. Baseline planform for supersonic wing-body, Wintzer and Sturdza [28].

8.1 Problem Definition

The procedure followed by Wintzer and Sturdza [28] to arrive at this baseline design assumed optimal sectional geometry for the wing and fuselage. We now attempt to find good values for these parameters: we tune these sectional properties to optimize cruise aerodynamics. At the same time, there are several crucial constraints from other disciplines such as structures and stability and control, that we ignore in this aerodynamic optimization but were incorporated in the original design problem. In an attempt to minimize the effect of our modifications on these other disciplines, we fix the planform geometry and impose constraints on the fuselage and wing. In particular, we stipulate that the volume of the fuel tanks in the wing and fuselage (shown by the red patches in Fig. 8.1) should be at least as large as that of the baseline design. In general, we would expect these constraints to be tight, because increasing the thickness of the wing or the volume of the fuselage will result in higher drag. With these constraints in place, one could envision our optimization as being part of a multidisciplinary approach, albeit one that we do not explicitly describe here.

Our design variables include the fuselage radii at seven axial stations (Akima spline used to loft the fuselage), the wing thickness and camber at three stations, and the wing twist

at two sections. The wing planform is held fixed. These design variables are depicted in Fig 8.2.

(a)

Fixed parameter	Value
Wing reference area (ft^2)	1007
Aspect ratio	3.0
Taper ratio	0.15
Quarter-chord sweep (deg)	57.4
Chord extension span	0.45
Leading-edge extension	0.49
Trailing-edge extension	0.38
Fuselage length (ft)	135

(b)

Weights and flight conditions	Value
Max. takeoff weight (lb)	84700
Weight at initial cruise (lb)	77036
Total fuel weight (lb)	40404
Mach number	1.7
Altitude (ft)	45,000

(c)

Design variable	Min.	Nominal	Max.
Root t/c	1%	2 %	5%
Break t/c	1%	2.8 %	6%
Tip t/c	1%	4.0 %	6%
Root camber/c	0	0	2%
Break camber/c	0	0	3%
Tip camber/c	0	0	3%
Break twist (deg)	-2	0	10
Tip twist (deg)	-2	0	10
Fuselage radius 1 (ft)	0.675	1.585	2.025
Fuselage radius 2 (ft)	2.500	2.746	3.375
Fuselage radius 3 (ft)	3.510	3.510	4.725
Fuselage radius 4 (ft)	3.510	3.510	4.725
Fuselage radius 5 (ft)	2.025	2.735	3.375
Fuselage radius 6 (ft)	2.025	1.910	3.375
Fuselage radius 7 (ft)	0.675	1.089	2.025

Table 8.1. Design variables and bounds for supersonic wing-body. (a) shows values for a few fixed parameters that define planform geometry, while (b) shows design variables and bounds for our aerodynamic optimization problem. **Note:** as defined, a positive value of twist indicates washout.

The objective is to minimize the drag of the wing-body configuration, subject to constraints on the volume of fuel that can be accommodated in the wing and the fuselage. These values are constrained to be no less than those for the baseline design. The optimization problem can now be stated as

$$\begin{array}{ll} \text{minimize} & C_D, \\ \text{w.r.t} & \text{design variables } \vec{x}, \\ \text{subject to} & \text{wing fuel volume } V_w(\vec{x}) >= \text{baseline value } w_{\text{req}}, \\ & \text{fuselage fuel volume } V_f(\vec{x}) >= \text{baseline value } f_{\text{req}}, \\ & \vec{x}_{\min} \leq \vec{x} \leq \vec{x}_{\max}. \end{array} \tag{8.1}$$

Using a linear exterior penalty method, we convert this inequality-constrained problem to a bound-constrained one as follows.

$$\begin{array}{ll} \text{minimize} & C_D + 0.1 \max(0, f_{\text{req}} - V_f(\vec{x})) + 0.3 \max(0, w_{\text{req}} - V_w(\vec{x})), \\ \text{w.r.t} & \text{design variables } \vec{x}, \\ \text{subject to} & \vec{x}_{\min} \leq \vec{x} \leq \vec{x}_{\max}. \end{array} \tag{8.2}$$

8.2 Analysis

Inviscid aerodynamic analysis is performed using the multifidelity drag method described in Sec. 2.3.1 with CART3D as the high-fidelity method and A502 as the low-fidelity method. For our configurations, the surface geometry is described using slightly less than 900,000 triangles. This results in a total of about 1.7 million cells. The paneling used to represent aircraft geometry for A502 is fairly coarse, with 10 spanwise panels, 10 chordwise panels, 4 meridian panels along each half of the fuselage, and 10 panels along the fuselage length. Parasite drag is estimated using the total wetted area of the configuration. The weight of the aircraft is held fixed, and includes the weight of tails, nacelles, and so on, from the baseline configuration. These other components, however, are not used in the drag estimate. Because of this the analysis method from Sec. 2.3.1 was slightly modified to only perform C_L matching (the effect was that each routine, A502 or CART3D, was run three times instead of five times per function call).

For both high- and low-fidelity analyses, the fuel volume computation is fairly straightforward: we assume the fuselage tank to be of circular cross section, and occupying a certain allowable fraction (0.9) of the fuselage volume at each station. The wing tank is situated in the main wing box, occupying a fraction of the wing's chordwise extent. Again, the cross-section is assumed to match that of the wing, with a given fraction (0.8) being available for fuel.

The total time taken for a high-fidelity analysis was about 30 minutes, when run over 12 cores across 3 nodes of a computer cluster running Rocks Linux on Intel hardware @2.0GHz, with 12.0 GB of RAM per node. On the same computer, the low-fidelity analysis runs on a single node in about 4 seconds. In order to speed up the design process, the large number of low-fidelity evaluations required by the auxiliary optimizer (a genetic algorithm), were evaluated in parallel. The auxiliary optimizer NSGA-II was allowed 400 generations of a 100-member population. The total time required for a single auxiliary optimization was about 4 hours, justifying the evaluation of up to 8–10 high-fidelity evaluations per auxiliary optimization.

8.3 Multifidelity Design Procedure

As with the analytic test problems, we model the difference between the high-fidelity and low-fidelity values of the drag coefficient C_D. This is done using a Kriging model with a squared exponential covariance function that guarantees smooth predictors. The process begins by optimizing the low-fidelity analysis alone. A few of the best designs from that process are used to 'seed' the difference model for the multifidelity procedure. The difference model is calibrated using maximum likelihood (ML-II), and this calibration is performed after every auxiliary optimization. This means that, at every iteration of the multifidelity design procedure, not only is the correction model improved by adding new data, but it is also recalibrated, further improving its accuracy.

Type-I objectives of mean prediction and standard deviation are used to perform the multiobjective optimization. The auxiliary optimizer used is NSGA-II, using a population size of 100 and 400 generations. At the end of each auxiliary optimization, we allow up to ten high-fidelity evaluations, based on a relative uncertainty threshold of 0.25. Every iteration must include at least one low-risk sample, and this sample is selected by sorting the pareto front in increasing order of uncertainty, and selecting the design one-tenth of the way down this sorted set.

8.4 Results

For the given planform, the lift-coefficient required for lift at the given flight conditions to equal weight is $C_{L_{\text{trim}}} = 0.1227$. We perform C_L-matching within both the high- and low-fidelity analysis, and for the baseline, this yields a total drag coefficient of $C_D = 0.014836$ at an angle of attack of 3.05°. Some details of the Cartesian mesh used in the analysis are shown in Fig. 8.3. For the same configuration, the low-fidelity analysis predicts a drag-coefficient of $C_D = 0.1554$. A comparison between the pressure distributions computed by the two fidelities at five spanwise stations is shown in Fig. 8.4. We see that CART3D captures shocks that are present on the outboard sections, whereas A502 does not.

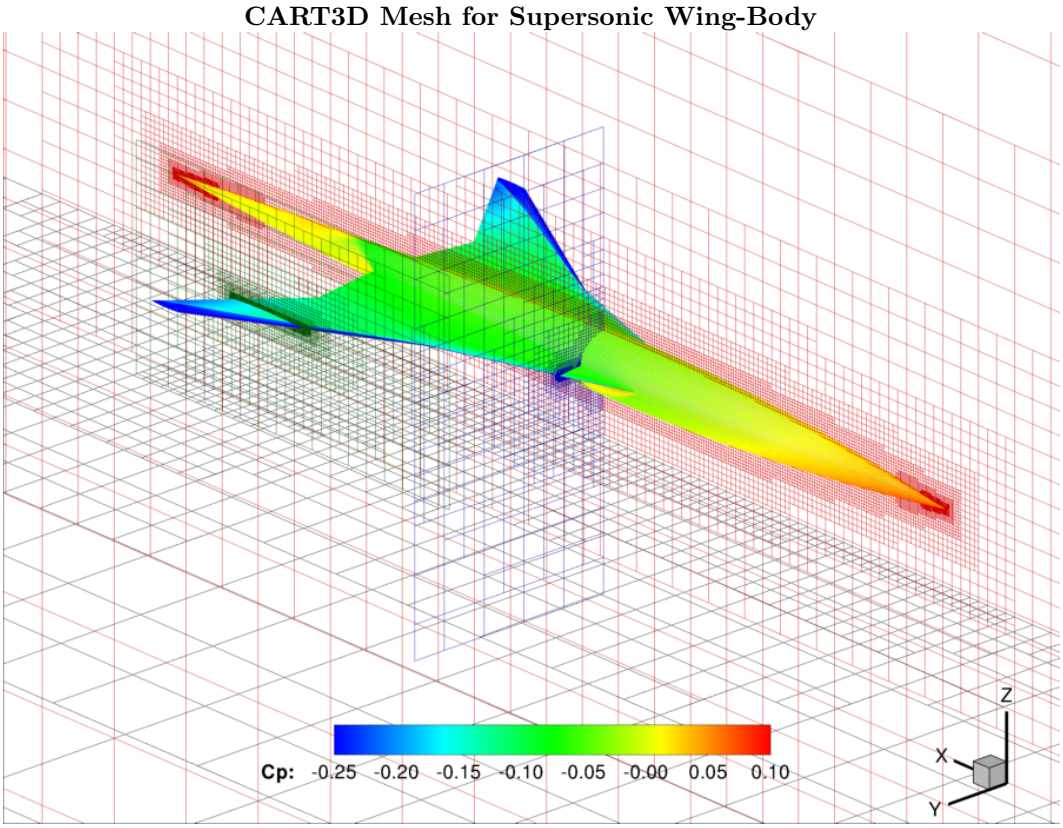


Figure 8.3. Details of CART3D mesh for wing-body analysis. Baseline C_p shown.

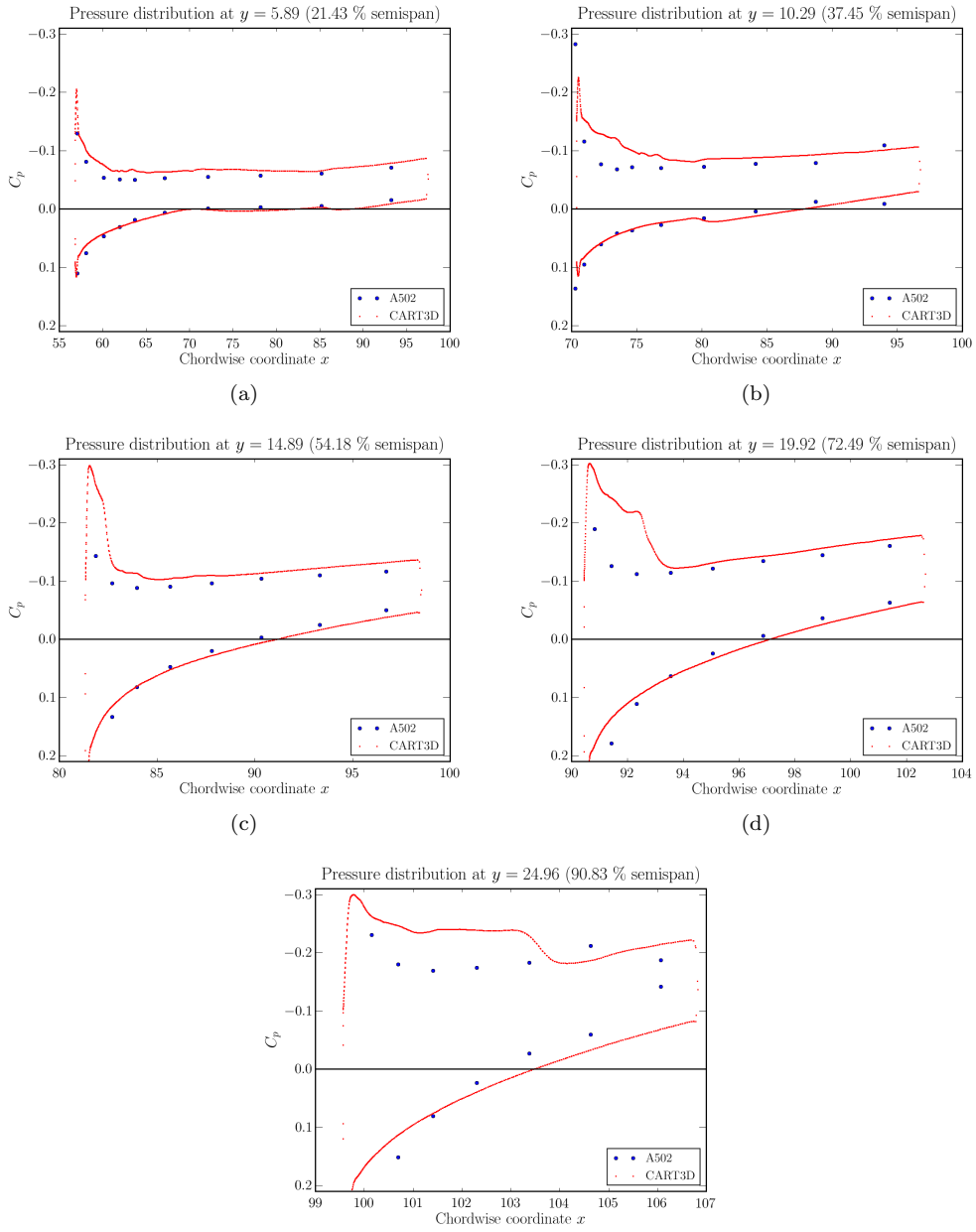


Figure 8.4. C_p cuts, baseline configuration. We can see that there are shocks on the outboard cuts, and while these are captured by CART3D, they are not by $A502$.

The sample budget used was 10 initial high-fidelity samples, followed by 30 further samples during optimization. We show the progress of the multifidelity optimization algorithm in three ways in Fig. 8.5. First, we plot high-fidelity drag values against the number of high-fidelity function evaluations. Since multiple high-fidelity evaluations are performed during every iteration, we also plot high-fidelity drag values against the total time elapsed in hours. This shows the time taken for auxiliary optimizations based on the corrected low-fidelity model, as well as that taken for high-fidelity evaluations. Finally, we plot the predicted

values of the low-fidelity model, along with the 3-sigma error bars in that prediction. First, we display the baseline and optimized values for the design variables, objective function and constraints.

Parameter	Baseline	Optimized
Root t/c	2.0%	3.4%
Break t/c	2.8%	1.0%
Tip t/c	4.0%	1.2%
Root camber/c	0	0.16%
Break camber/c	0	0.78%
Tip camber/c	0	0.52%
Break twist (deg)	0	1.40
Tip twist (deg)	0	6.36
Fuselage radius 1 (ft)	1.585	1.591
Fuselage radius 2 (ft)	2.746	2.594
Fuselage radius 3 (ft)	3.510	3.510
Fuselage radius 4 (ft)	3.510	3.510
Fuselage radius 5 (ft)	2.375	2.631
Fuselage radius 6 (ft)	1.910	2.266
Fuselage radius 7 (ft)	1.089	1.360
Cruise C_D (CART3D)	0.01484	0.01383
Fuselage fuel volume margin (cu. ft.)	0	32.4
Wing fuel volume margin (cu. ft.)	0	0

Table 8.2. Baseline and optimized design variables for supersonic wing-body. We see that the break thickness is essentially at its lower bound. The sections have a small amount of camber, but the break and tip sections are severely washed out to move the lift inboard. Note that positive twist indicates washout. The fuselage radii show a mild 'coke-bottle' effect due to area-ruling, most clearly seen in the figures on the following pages.

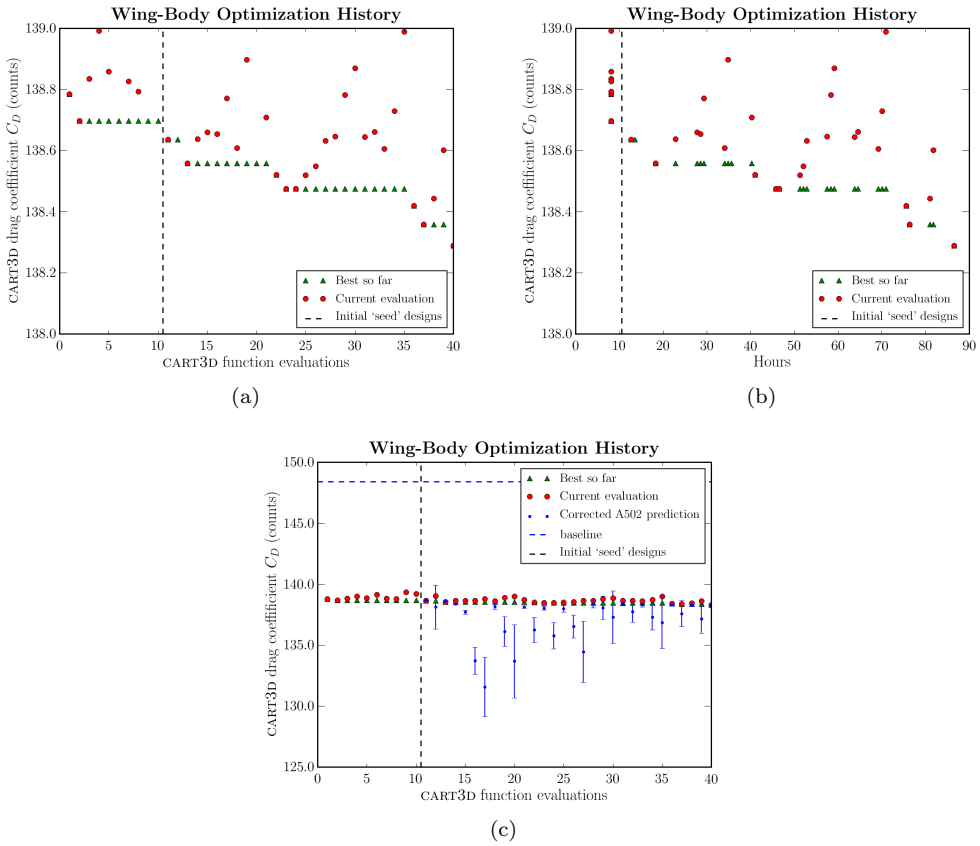
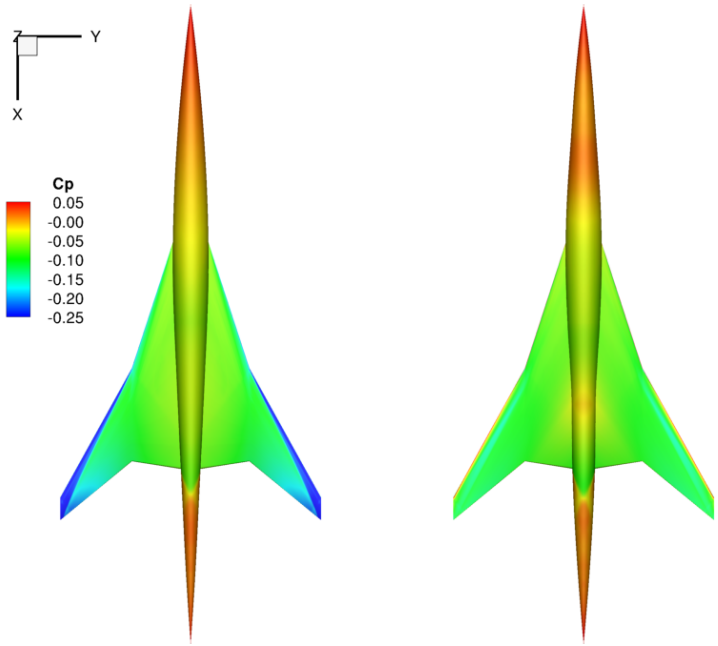
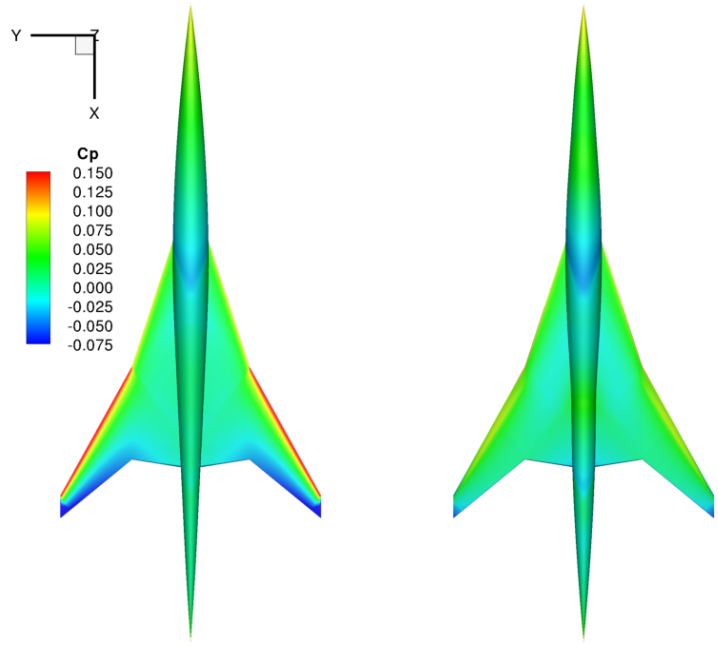


Figure 8.5. Iteration history for supersonic wing-body optimization.



(a) Baseline, top surface

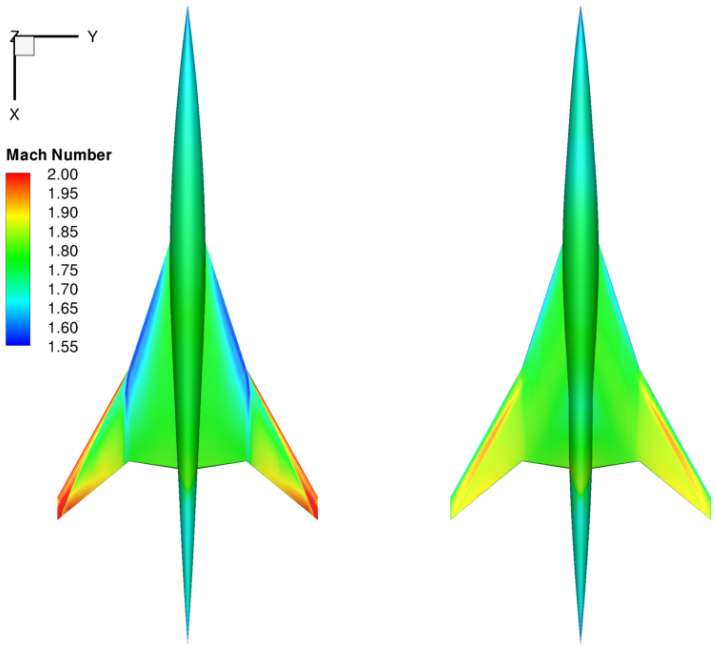
(b) Optimized, top surface



(c) Baseline, bottom surface

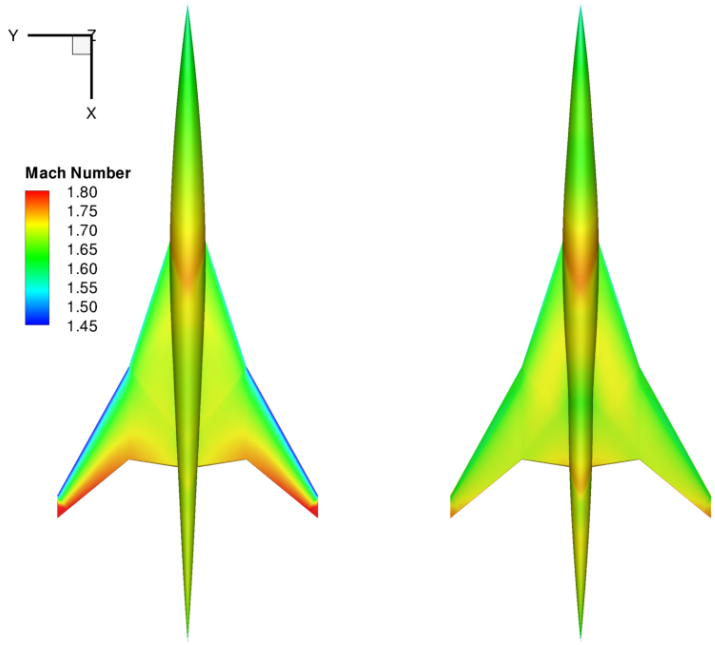
(d) Optimized, bottom surface

Figure 8.6. Surface pressure distributions for baseline and optimized configurations.



(a) Baseline, top surface

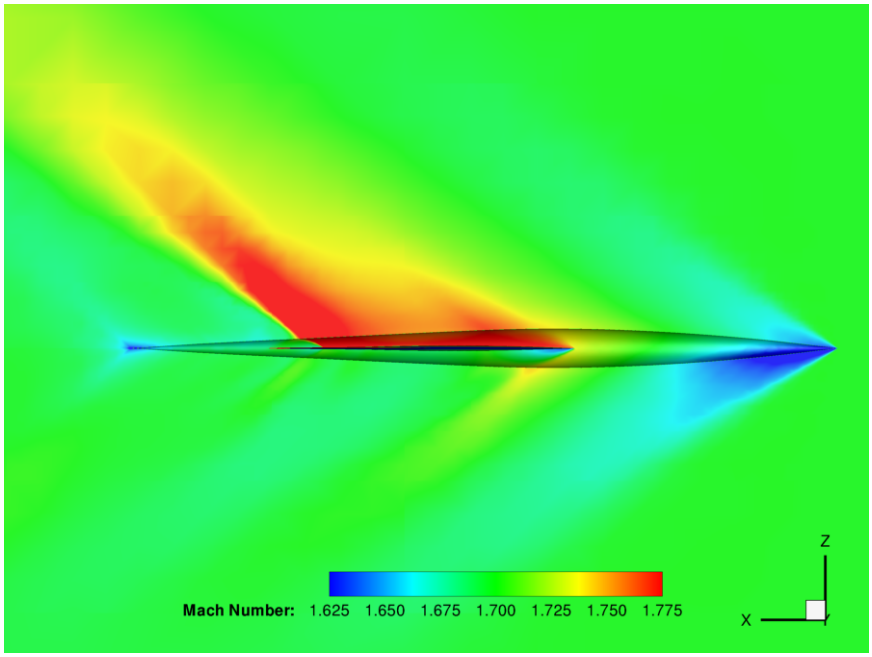
(b) Optimized, top surface



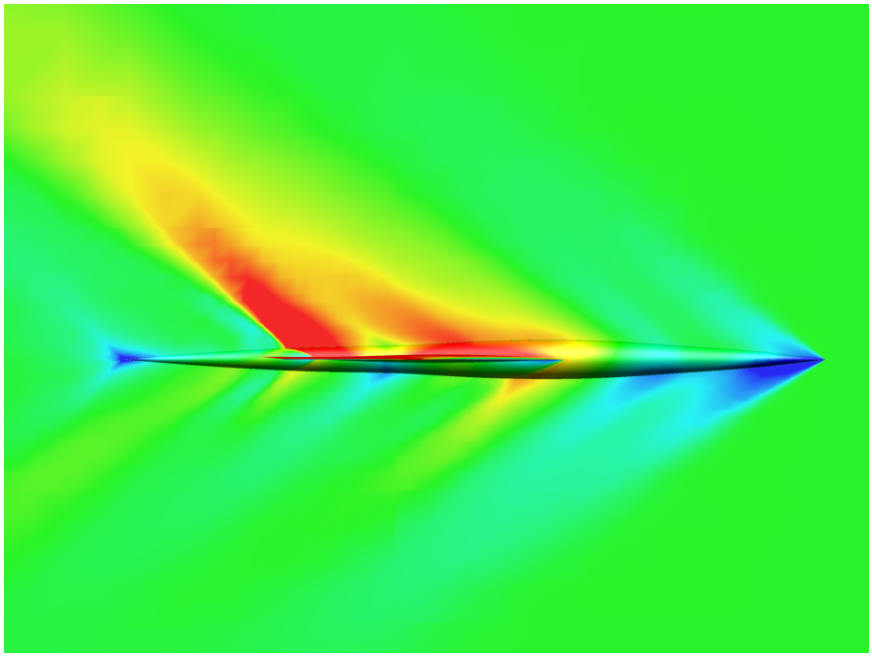
(c) Baseline, bottom surface

(d) Optimized, bottom surface

Figure 8.7. Mach number distributions for baseline and optimized configurations.

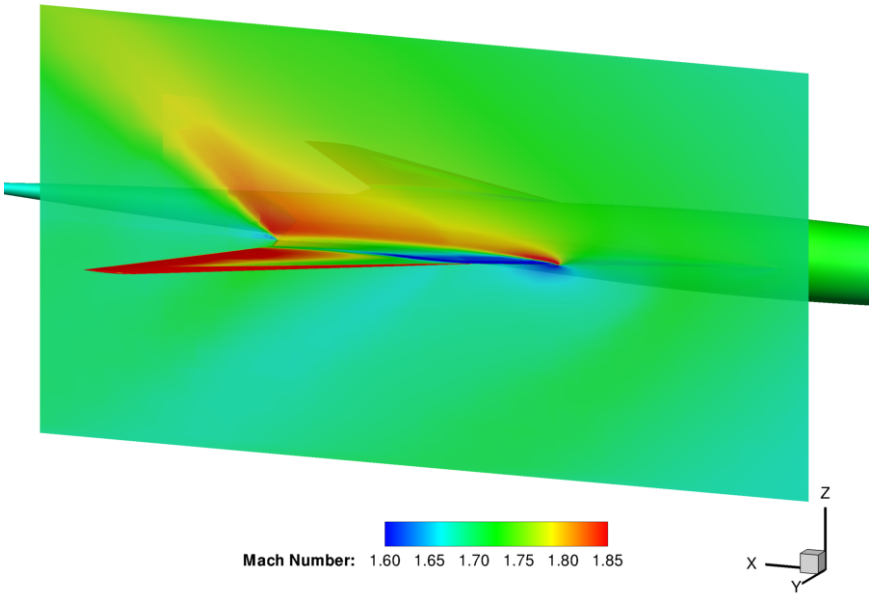


(a)

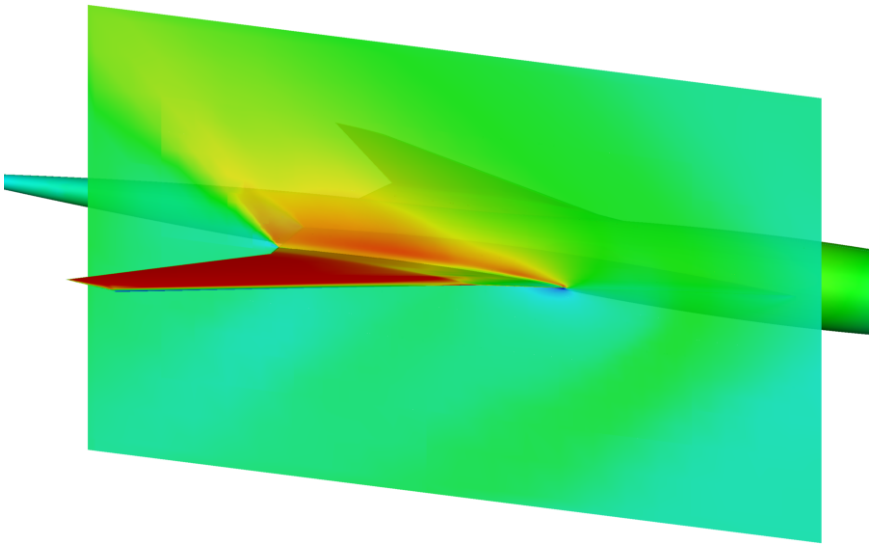


(b)

Figure 8.8. Near-field Mach number variation for supersonic wing-body over a cut plane defined by the plane of symmetry $y = 0$.

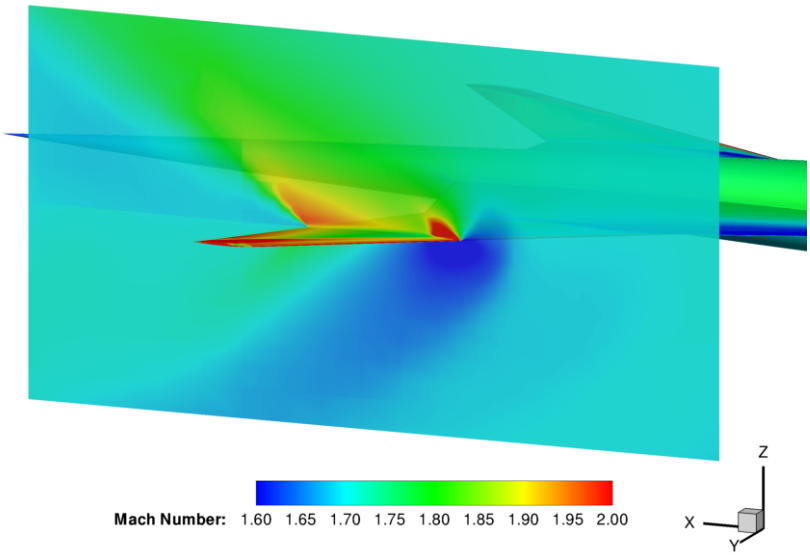


(a) Baseline

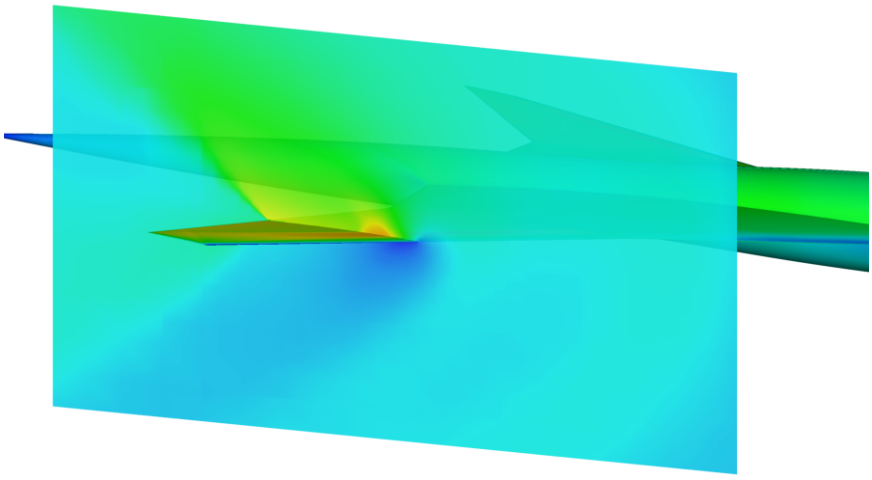


(b) Optimized

Figure 8.9. Near-field Mach number variation over inboard cut plane.



(a) Baseline



(b) Optimized

Figure 8.10. Near-field Mach number variation over outboard cut plane.

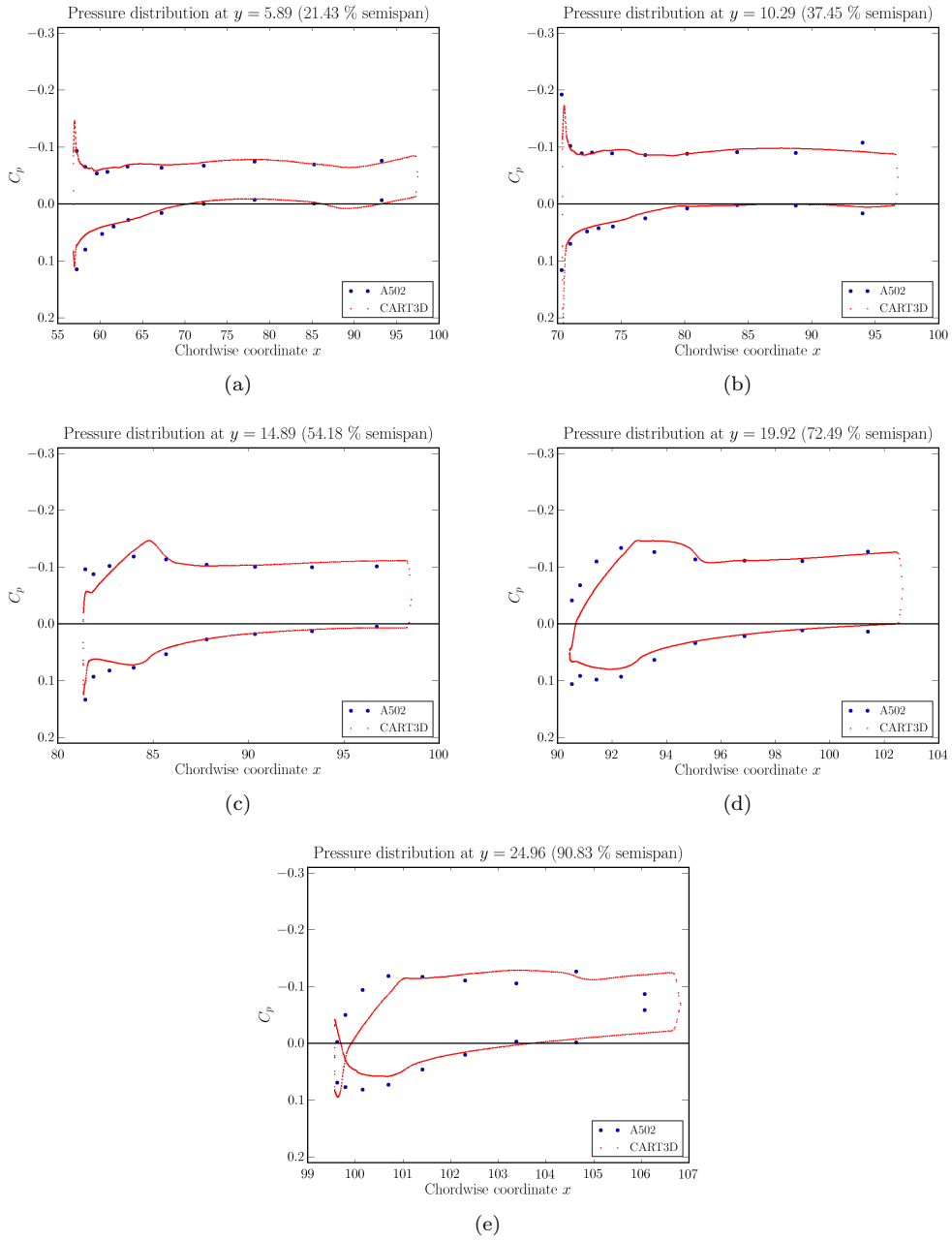
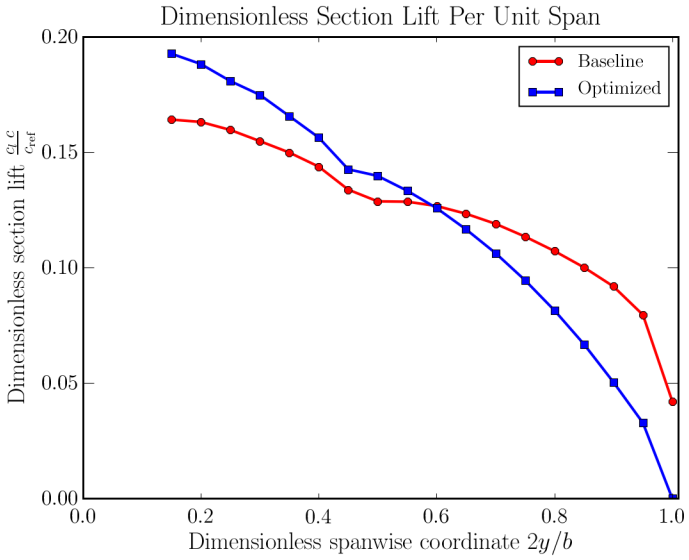
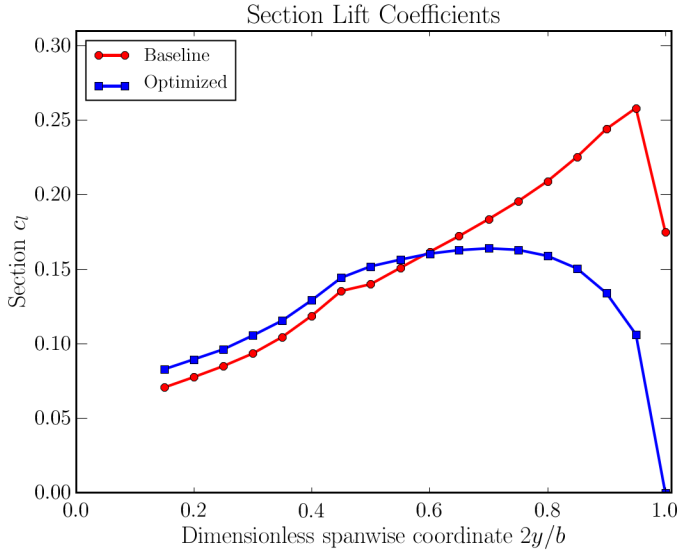


Figure 8.11. C_p cuts, optimized configuration. We see that the previous suction peaks on the outboard sections are eliminated, and with them the shocks that caused high drag.



(a)

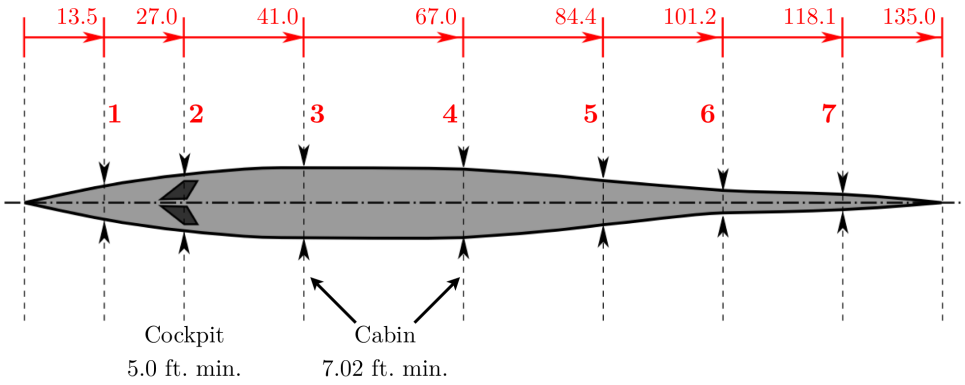


(b)

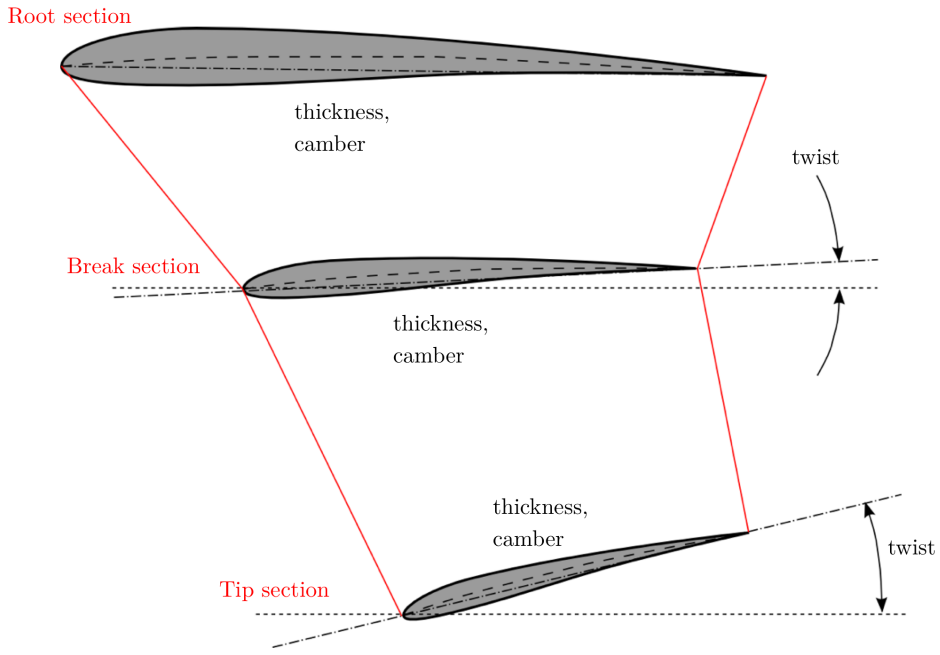
Figure 8.12. Spanwise lift distributions for supersonic wing-body, computed using CART3D. We see that the optimized configuration has moved the lift significantly inboard, to reduce the wave drag due to lift on the outboard sections, where there are shocks. This is also reflected in the reduced C_l values over the outboard sections.

8.5 Discussion

In this section, we presented a multifidelity aerodynamic optimization problem for a supersonic wing-body configuration based on a conceptual design by Wintzer and Sturdza [28]. That study used simple conceptual design tools to analyze multiple disciplines including aerodynamics, structures, propulsion, stability and control, etc. In contrast, we attempted



(a)



(b)

Figure 8.2. Design variables for supersonic wing-body. (a) shows the fuselage parametrization using radii at seven stations, while (b) shows the wing parametrization using three thicknesses, three cambers, and two twists. Lines and text in red denote fixed values.

multifidelity optimization of the aerodynamics alone, with simplistic constraints on the fuel volume required in the wings and fuselage. We presented this as a believable surrogate for a subspace optimization in a multidisciplinary optimization framework, without specifying that framework in its entirety. Therefore, any results must be interpreted cautiously.

For instance, the current optimized value of a 1% thickness at the break section will clearly lead to an unacceptably large weight penalty. The use of a reasonable structural model will increase the thicknesses at the break and tip sections to more reasonable values. The optimization of cruise aerodynamics alone restricted our choice of design variables. Including low-speed constraints will permit meaningful modification of planform geometry.

From Fig. 8.5(c), we observe that the true values are often outside the 3-sigma intervals, indicating that the model needs recalibration. This indicates that it is not advisable to use this possibly erroneous error estimate in termination criteria. We also see that the corrected model progressively improves as the algorithm progresses.

Also note that the low-fidelity model in this case was reasonably accurate: it provided a reduction of tens of drag counts, whereas the use of the high-fidelity model yielded less than one drag count of further improvement. A promising next step would be to solve this problem in two stages, using A502 and CART3D as multiple high-fidelity models. Nevertheless, even in the current restrictive setting, this example demonstrates a viable algorithm for the use of multifidelity models in optimization.

9 High-Fidelity-Gradient-Free Local Optimization

This chapter presents a summary of the provably convergent unconstrained high-fidelity-gradient-free multifidelity optimization algorithm presented at the 6^{th} Multidisciplinary Design Optimization Specialist Conference, Ref. [29]. The method uses a Bayesian model calibration to create a surrogate model of the high-fidelity function from a low-fidelity function. The calibration is done by interpolating the error between the high- and low-fidelity functions with a Gaussian process. When appropriately distributed spatial calibration points are used, the low-fidelity function and radial basis function interpolation generate a fully linear model. This condition is sufficient to prove convergence in a trust-region framework. In the case when there are multiple lower-fidelity models, the predictions of all calibrated lower-fidelity models can be combined with a maximum likelihood estimator constructed using Kriging variance estimates from the radial basis function models. This procedure allows for flexibility in sampling lower-fidelity functions, does not alter the convergence proof of the optimization algorithm, and is shown to be robust to poor low-fidelity information. The algorithm is compared with an unconstrained single-fidelity quasi-Newton algorithm and two first-order consistent multifidelity trust-region algorithms. For simple functions the quasi-Newton algorithm uses slightly fewer high-fidelity function evaluations; however, for more complex supersonic airfoil design problems it uses significantly more. In all cases tested, our radial basis function calibration approach uses fewer high-fidelity function evaluations when compared with first-order consistent trust-region schemes.

9.1 Motivation

There are several different multifidelity optimization strategies that optimize a high-fidelity function using a lower-fidelity surrogate. One class of approaches uses trust regions. These methods are provably convergent to a local optimum of the high-fidelity function, if at the center of the trust region the low-fidelity function value and derivative are scaled or shifted to be equal to the high-fidelity function and gradient[30, 31, 32]. Another multifidelity approach is to combine a pattern-search with conformal space mapping, where the least squares difference between a low-fidelity function and high-fidelity function is minimized at a collection of points by mapping the low-fidelity design space to the high-fidelity design space. This method is also provably convergent to an optimum of the high-fidelity function[33]. A third general approach is Efficient Global Optimization (EGO) developed by Jones *et al* [2]. In this method, a Bayesian uncertainty approach is used to find regions in the design space with a high likelihood of having an optimal solution. EGO uses a combination of a regression model and an uncertainty estimate based on distance from known points as a way to find better points. An improvement to Jones' approach is to use model calibration techniques to model the difference or quotient between a high- and low-fidelity function as opposed to modeling the high-fidelity function itself. In this way, a low-fidelity model can increase the efficiency of finding an optimum of a high-fidelity function in situations where using only a regression surface requires a considerable number of function evaluations for calibration [34, 35, 36]. These model calibration techniques are generally based on heuristic methods and are not provably convergent to an optimum of the high-fidelity function.

In this chapter, we present a provably convergent multifidelity optimization algorithm based on model calibration. The first-order-consistent trust-region methods mentioned

above can be thought of as employing model calibration; however, the calibration is only local and temporary, since sample points from previous iterations are not re-used. The challenge we address here is to produce a surrogate model that captures local function behavior sufficiently well to prove convergence, while capturing global function behavior to speed convergence. Carter proved that a trust-region algorithm is convergent provided the error between the gradient of the function and the gradient of surrogate model is bounded by a constant times the gradient of the function [37]. Oeuvray showed that a radial basis function interpolation satisfies this criterion from Carter, provided the interpolation points satisfy certain conditions [38]. Conn *et al.* then showed that both the error between a function and a smooth interpolation model as well as the error between the function's derivative and the interpolation model's derivative can be bounded by appropriately selecting interpolation points [39]. Conn *et al.* also proved that any interpolation model that can locally be made fully linear (defined in the next section) can be used in a provably convergent trust-region framework [40]. Wild *et al.* then developed an algorithm to produce fully linear radial basis function interpolation models and showed that his method could be used within Conn's provably convergent optimization framework [41, 42, 43].

The method in this chapter combines the provably convergent optimization frameworks of Wild *et al.* and Conn *et al.* with Bayesian model calibration ideas to result in a provably convergent multifidelity optimization approach that does not require high-fidelity gradient information. Section 9.2 provides an overview of the derivative-free trust-region algorithm using fully linear models proposed by Conn *et al.* [40]. Section 9.3 discusses the approach of Wild *et al.* [41] to build a fully linear model using RBF functions, and presents our extension to the case of multifidelity model calibration. Section 9.4 provides an overview of the computational implementation of the method and suggests a way to incorporate the method of generating fully linear models from Wild *et al.* [43] with flexible Bayesian model calibration techniques. Section 9.5 demonstrates the multifidelity optimization algorithm on an analytical example and a supersonic airfoil design problem. Section 9.6 then develops the extension of our approach to the case when there are multiple lower-fidelity models.

9.2 Trust-Region-Based Multifidelity Optimization

We consider a setting where we have two (or more) models that represent the physical system of interest: a high-fidelity function that accurately estimates system metrics of interest but is expensive to evaluate, and a low-fidelity function with lower accuracy but cheaper evaluation cost. We define our high-fidelity function as $f_{\text{high}}(\mathbf{x})$ and our low-fidelity function as $f_{\text{low}}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$ is the vector of n design variables. Our goal is to solve the unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f_{\text{high}}(\mathbf{x}), \tag{9.1}$$

using information from evaluations of $f_{\text{low}}(\mathbf{x})$ to reduce the required number of evaluations of $f_{\text{high}}(\mathbf{x})$.

We use the derivative-free trust-region algorithm of Conn *et al.* [40] to solve (9.1). From an initial design vector $\mathbf{x}_0$, the trust-region method generates a sequence of design vectors that each reduce the high-fidelity function value, where we denote $\mathbf{x}_k$ to be this design vector on the kth trust-region iteration. Following the general Bayesian calibration approach in Ref. [35], we define $e_k(\mathbf{x})$ to be a model of the error between the high- and low-fidelity functions on the kth trust-region iteration, and we construct a surrogate model $m_k(\mathbf{x})$ for $f_{\text{high}}(\mathbf{x})$ as

$$m_k(\mathbf{x}) = f_{\text{low}}(\mathbf{x}) + e_k(\mathbf{x}). \tag{9.2}$$

We define the trust region at iteration k, $\mathcal{B}_k$, to be the region centered at $\mathbf{x}_k$ with size Δ_k,

$$\mathcal{B}_k = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}, \tag{9.3}$$

where any norm can be used, provided there exist constants c_1 and c_2 such that

$$\|\cdot\|_2 \leq c_1\|\cdot\| \qquad \text{and} \qquad \|\cdot\| \leq c_2\|\cdot\|_2. \tag{9.4}$$

If the high-fidelity function $f_{\text{high}}(\mathbf{x})$ and the surrogate models $m_k(\mathbf{x})$ satisfy certain conditions, this framework provides a guarantee of convergence to a local minimum of the high-fidelity function $f_{\text{high}}(\mathbf{x})$. Specifically, the convergence proof requires that the high-fidelity function $f_{\text{high}}(\mathbf{x})$ be (i) continuously differentiable, (ii) have a Lipschitz continuous derivative, and (iii) be bounded from below within a region of a relaxed level-set, $\mathcal{L}(\mathbf{x}_0)$, defined as

$$\begin{aligned} L(\mathbf{x}_0) &= \{\mathbf{x} \in \mathbb{R}^n : f_{\text{high}}(\mathbf{x}) \leq f_{\text{high}}(\mathbf{x}_0)\} && (9.5)\\ B(\mathbf{x}_k) &= \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_{\max}\} && (9.6)\\ \mathcal{L}(\mathbf{x}_0) &= L(\mathbf{x}_0) \bigcup_{\mathbf{x_k} \in L(\mathbf{x}_0)} B(\mathbf{x}_k), && (9.7) \end{aligned}$$

where $\Delta_{\max}$ is the maximum allowable trust-region size. The relaxed level-set is required because the trust-region algorithm may attempt to evaluate the high-fidelity function at points outside of the level set at $\mathbf{x}_0$. The convergence proof further requires that the surrogate models $m_k(\mathbf{x})$ are *fully linear*, where the following definition of a fully linear model is from Conn *et al.* [40]:

Definition 1. *Let a function $f_{high}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ that satisfies the conditions (i)–(iii) above, be given. A set of model functions $\mathcal{M} = \{m : \mathbb{R}^n \to \mathbb{R}, m \in C^1\}$ is called a fully linear class of models if the following occur:*

There exist positive constants κ_f, κ_g and κ_{blg} such that for any $\mathbf{x} \in L(\mathbf{x}_0)$ and $\Delta_k \in (0, \Delta_{\max}]$ there exists a model function $m_k(\mathbf{x})$ in $\mathcal{M}$ with Lipschitz continuous gradient and corresponding Lipschitz constant bounded by k_{blg}, and such that the error between the gradient of the model and the gradient of the function satisfies

$$\|\nabla f_{high}(\mathbf{x}) - \nabla m_k(\mathbf{x})\| \leq \kappa_g \Delta_k \quad \forall \mathbf{x} \in \mathcal{B}_k \tag{9.8}$$

and the error between the model and the function satisfies

$$|f_{high}(\mathbf{x}) - m_k(\mathbf{x})| \leq \kappa_f \Delta_k^2 \quad \forall \mathbf{x} \in \mathcal{B}_k. \tag{9.9}$$

Such a model $m_k(\mathbf{x})$ is called fully linear on $\mathcal{B}_k$ [40].

At iteration k, the trust-region algorithm solves the subproblem

$$\begin{aligned} \min_{\mathbf{s}_k} \quad & m_k(\mathbf{x}_k + \mathbf{s}_k) \\ \text{s.t.} \quad & \|\mathbf{s}_k\| \leq \Delta_k \end{aligned} \tag{9.10}$$

to determine the trust-region step $\mathbf{s}_k$. The steps found in the trust-region subproblem must satisfy a sufficient decrease condition. At iteration k, we require that the model $m_k(\mathbf{x})$ have a finite upper bound on the 2-norm of its Hessian matrix evaluated at $\mathbf{x}_k$: $\|H_k(\mathbf{x}_k)\| \leq \kappa_{bhm} < \infty$. This bound on the Hessian may be viewed as a bound on the Lipschitz constant of the gradient of $m_k(\mathbf{x}_k)$ [40]. The sufficient decrease condition requires the step to satisfy the fraction of Cauchy decrease. As given in Ref. [40] and Ref. [42], this requires that for some constant, $\kappa_{FCD} \in (0, 1)$, the step $\mathbf{s}_k$ satisfies

$$m_k(\mathbf{x}_k) - m_k(\mathbf{x}_k + \mathbf{s}_k) \geq \frac{\kappa_{FCD}}{2}\|\nabla m_k(\mathbf{x}_k)\| \min\left[\frac{\|\nabla m_k(\mathbf{x}_k)\|}{\kappa_{bhm}}, \frac{\|\nabla m_k(\mathbf{x}_k)\|_2}{\|\nabla m_k(\mathbf{x}_k)\|}\Delta_k\right]. \tag{9.11}$$

The high-fidelity function f_{high} is then evaluated at the new point, $\mathbf{x}_k + \mathbf{s}_k$. We compare the actual improvement in the function value with the improvement predicted by the model by defining

$$\rho_k = \frac{f_{\text{high}}(\mathbf{x}_k) - f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{x}_k) - m_k(\mathbf{x}_k + \mathbf{s}_k)}. \tag{9.12}$$

The trial point is accepted or rejected according to

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k & \text{if } \rho_k > 0 \\ \mathbf{x}_k & \text{otherwise.} \end{cases} \tag{9.13}$$

If the step is accepted, then the trust region is updated to be centered on the new iterate $\mathbf{x}_{k+1}$. The size of the trust region, Δ_k, must now be updated based on the quality of the surrogate model prediction. The size of the trust region is increased if the surrogate model predicts the change in the function value well and the trust region is contracted if the model predicts the function change poorly. Specifically, we update the trust region size using

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta \\ \gamma_0 \Delta_k & \text{if } \rho_k < \eta, \end{cases} \tag{9.14}$$

where $0 < \eta < 1$, $0 < \gamma_0 < 1$, and $\gamma_1 > 1$.

A new fully linear model, $m_{k+1}(\mathbf{x})$, is then built using the radial basis function interpolation approach described in the next section. That surrogate model will be fully linear on a region $\mathcal{B}_{k+1}$ having center $\mathbf{x}_{k+1}$ and size Δ_{k+1}.

To check for algorithm termination, the gradient of the model is computed at $\mathbf{x}_{k+1}$. If $\|\nabla m_{k+1}(\mathbf{x}_{k+1})\| > \epsilon$ for a small ϵ, the trust-region algorithm will continue to iterate, solving the next subproblem on the new trust region, $\mathcal{B}_{k+1}$, with the updated model, $m_{k+1}(\mathbf{x})$. However, if $\|\nabla m_{k+1}(\mathbf{x}_{k+1})\| \leq \epsilon$, we need to confirm that the algorithm has reached a stationary point of $f_{\text{high}}(\mathbf{x})$. If gradients of the high-fidelity function are available, one could evaluate if $\|\nabla f_{\text{high}}(\mathbf{x}_{k+1})\| \leq \epsilon$ directly. In the general derivative-free case, we use the condition in Eq. 9.8, and show that if $\Delta_{k+1} \to 0$ then $\|\nabla f_{\text{high}}(\mathbf{x}_{k+1}) - \nabla m_{k+1}(\mathbf{x}_{k+1})\| \to 0$. In practice we achieve this by updating the model to be fully linear on a trust region with size some fraction, $0 < \alpha < 1$, of Δ_{k+1}. This process continues until either $\|\nabla m_{k+1}(\mathbf{x}_{k+1})\| > \epsilon$, in which case the trust-region algorithm will continue with the updated model and updated Δ_{k+1}, or $\Delta_{k+1} \leq \epsilon_2$, for a small ϵ_2, which terminates the algorithm. This process of checking for convergence is referred to as the criticality check in Conn *et al* [40].

9.3 Interpolation-Based Multifidelity Models

In this section we discuss a method of creating surrogate models that satisfy the conditions for provable convergence presented in Section 9.2. This section first presents an overview of the radial basis function (RBF) interpolation approach of Wild *et al.*[41], where the interpolation points are chosen so that the resulting model is fully linear. Next, we present an extension of this approach to the case of multifidelity models.

Define $\mathbf{d}_j$ to be the jth point in the set of designs at which the high-fidelity and low-fidelity functions have been sampled. Define $\mathbf{y}_i$ to be the vector from the current iterate (i.e., center of the current trust region), $\mathbf{x}_k$, to any sample point inside or within the vicinity of the current trust region, $\mathbf{d}_i$, that is selected to be an interpolation point. Also define $\mathcal{Y}$ to be the set of the zero vector and all of the vectors $\mathbf{y}_i$. This notation is shown graphically in Figure 9.1.

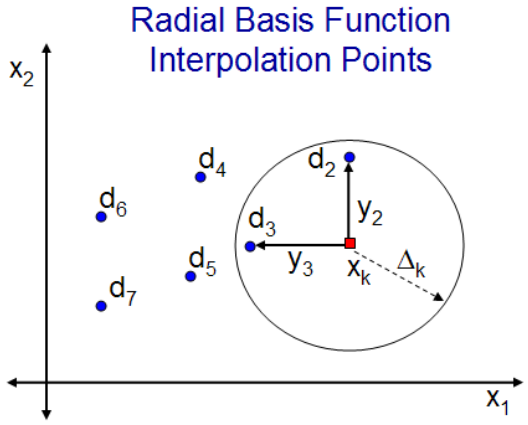


Figure 9.1. Graphical representation of the notation used to define points and vectors in and around the trust region.

The RBF interpolation is defined so that by construction the surrogate model is equal to the high-fidelity function at all interpolation points. That is, the error between the high- and low-fidelity functions is interpolated exactly for all points defined by the vectors within $\mathcal{Y}$,

$$e_k(\mathbf{x}_k + \mathbf{y}_i) = f_{\text{high}}(\mathbf{x}_k + \mathbf{y}_i) - f_{\text{low}}(\mathbf{x}_k + \mathbf{y}_i) \quad \forall \mathbf{y}_i \in \mathcal{Y}. \tag{9.15}$$

The RBF interpolation has the form

$$e_k(\mathbf{x}) = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \phi\left(\|\mathbf{x} - \mathbf{x}_k - \mathbf{y}_i\|\right) + \sum_{i=1}^{n+1} \nu_i \pi_i(\mathbf{x} - \mathbf{x}_k), \tag{9.16}$$

where ϕ is any positive definite, twice continuously differentiable RBF with $\phi'(0) = 0$, and the second term in (9.16) represents a linear tail, where π_i denotes the i^{th} component of the vector $\Pi(\mathbf{x}-\mathbf{x}_k) = [1\ (\mathbf{x}-\mathbf{x}_k)]^T$. The coefficients λ_i and ν_i represent the RBF interpolation, and are found by the QR-factorization technique of Wild *et al* [41]. In order for the model to be fully linear, the RBF coefficients λ_i and ν_i must be bounded in magnitude. This is achieved by using the interpolation point selection method in Wild *et al* [41]. The process can be summarized as follows. First, the existing high-fidelity sample points, $\mathbf{d}_j$, in the vicinity of the trust region are tested for affine independence. If fewer than $n+1$ affinely independent points are found, additional high-fidelity function evaluations are required to generate them. Second, we test all other points $\mathbf{d}_j$ at which the high-fidelity function value is known, by measuring the impact of their addition as interpolation points on the RBF coefficients λ_i and ν_i. Those points that ensure the RBF coefficients remain bounded are used as additional interpolation points to update the model. Wild proved that this RBF interpolation model construction algorithm produces a fully linear model for a function satisfying conditions (i) and (ii) above [42].

In order for Wild's interpolation approach to be applicable in our Bayesian calibration setting, we require that the error function defined by $f_{\text{high}}(\mathbf{x}) - f_{\text{low}}(\mathbf{x})$ satisfies conditions (i) and (ii) above. Condition (i), that the function is continuously differentiable, is satisfied if both $f_{\text{high}}(\mathbf{x})$ and $f_{\text{low}}(\mathbf{x})$ are continuously differentiable. To establish condition (ii), that the derivative of $f_{\text{high}}(\mathbf{x}) - f_{\text{low}}(\mathbf{x})$ is Lipschitz continuous, we require that both $\nabla f_{\text{high}}(\mathbf{x})$ and $\nabla f_{\text{low}}(\mathbf{x})$ be Lipschitz continuous in the relaxed level set defined in Eq. (9.7). For the

high-fidelity function we require

$$\frac{\|\nabla f_{\text{high}}(\mathbf{x}_1) - \nabla f_{\text{high}}(\mathbf{x}_2)\|}{\|\mathbf{x}_1 - \mathbf{x}_2\|} \leq \kappa_{\text{high}} \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{L}(\mathbf{x}_0), \tag{9.17}$$

and for the low-fidelity function,

$$\frac{\|\nabla f_{\text{low}}(\mathbf{x}_1) - \nabla f_{\text{low}}(\mathbf{x}_2)\|}{\|\mathbf{x}_1 - \mathbf{x}_2\|} \leq \kappa_{\text{low}} \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{L}(\mathbf{x}_0), \tag{9.18}$$

with Lipschitz constants κ_{high} and κ_{low}, respectively. Since the difference between any two functions with Lipschitz continuous first derivatives is also Lipschitz continuous, we obtain

$$\frac{\|\nabla[f_{\text{high}}(\mathbf{x}_1) - f_{\text{low}}(\mathbf{x}_1)] - \nabla[f_{\text{high}}(\mathbf{x}_2) - f_{\text{low}}(\mathbf{x}_2)]\|}{\|\mathbf{x}_1 - \mathbf{x}_2\|} \leq \kappa_{\text{high}} + \kappa_{\text{low}} \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{L}(\mathbf{x}_0) \tag{9.19}$$

where the Lipschitz constant of the difference is bounded by $\kappa_{\text{high}} + \kappa_{\text{low}}$. Accordingly, the convergence proof for the trust-region algorithm used in Conn *et al.* [40] holds, and this multifidelity algorithm is provably convergent to an optimum of the high-fidelity function.

9.4 Numerical Implementation of Algorithms

This section presents an overview of the numerical implementation of the multifidelity optimization algorithm and suggests a manner in which the method of Wild *et al.* [43] to generate fully linear models can be used in a flexible Bayesian calibration setting. The first subsection, Section 9.4.1, implements the the trust region based optimization algorithm presented in Section 9.2. Whenever creation of a new fully linear model is needed, the method discussed in Section 9.3 is implemented using the algorithm presented in Section 9.4.2.

9.4.1 Trust Region Implementation

Algorithm 4 provides an overview of the numerical implementation of the trust-region optimization method presented in Section 9.2. For each trust-region iteration, the algorithm guarantees that a step is found that satisfies the fraction of Cauchy decrease, Eq. 9.11. The algorithm only samples the high-fidelity function when necessary for convergence, and it stores all high-fidelity function evaluations in a database so that design points are never re-evaluated. Whenever an updated surrogate model is needed, the model generation method described in the following subsection creates a surrogate model using this database of high-fidelity function evaluations together with new high-fidelity evaluations when necessary. The parameters of the trust-region optimization algorithm were defined in Section 9.2, while recommended values and sensitivity of results to those values will be presented in Section 9.5.

9.4.2 Fully Linear Bayesian Calibration Models

Algorithm 5 presents the numerical implementation of the method to generate fully linear surrogate models, allowing for a Bayesian maximum likelihood estimate of the RBF correlation length. The RBF models used in Bayesian model calibration have a length scale parameter that provides flexibility. For instance, in the Gaussian RBF model, $\phi(r) = e^{-r^2/\xi^2}$, the parameter ξ is a variable length scale that can alter the shape of the correlation structure. If the interpolation errors are assumed to have a Gaussian distribution, then a maximum likelihood estimate can be used to estimate the value of ξ that best represents the data [3, 4]. Therefore, our process to generate a fully linear surrogate model uses the method of Wild *et*

Algorithm 4: Trust-Region Algorithm for Iteration k

1: Solve the trust-region subproblem using nonlinear programming techniques to find the $\mathbf{s}_k$ that solves,

$$\begin{aligned} \min_{\mathbf{s}_k} \quad & m_k(\mathbf{x}_k + \mathbf{s}_k) \\ \text{s.t.} \quad & \|\mathbf{s}_k\| \leq \Delta_k. \end{aligned}$$

1a: If the subproblem solution fails to satisfy the fraction of Cauchy decrease, Eq. 9.11, the simple line search from Conn *et al.* is used [40].

2: If $f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ has not been evaluated previously, evaluate the high-fidelity function at that point.

2a: Store $f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ in database.

3: Compute the ratio of actual improvement to predicted improvement,

$$\rho_k = \frac{f_{\text{high}}(\mathbf{x}_k) - f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{x}_k) - m_k(\mathbf{x}_k + \mathbf{s}_k)}.$$

4: Accept or reject the trial point according to ρ_k,

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k & \text{if } \rho_k > 0 \\ \mathbf{x}_k & \text{otherwise.} \end{cases}$$

5: Update the trust region size according to ρ_k,

$$\Delta_{k+1} \quad = \quad \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \rho_k \geq \eta \\ \gamma_0 \Delta_k & \text{if } \rho_k < \eta. \end{cases}$$

5: Create a new model $m_{k+1}(\mathbf{x})$ that is fully linear on $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_{k+1}\| \leq \Delta_{k+1}\}$ using Algorithm 5.

6: Check for convergence: if $\|\nabla m_{k+1}(\mathbf{x}_{k+1})\| > \epsilon$, algorithm is not converged—go to step 1. Otherwise,

6a: While $\|\nabla m_{k+1}(\mathbf{x}_{k+1})\| \leq \epsilon$ and $\Delta_{k+1} > \epsilon_2$,

6b: Reduce the trust region size, $\alpha \Delta_{k+1} \to \Delta_{k+1}$.

6c: Update model $m_{k+1}(\mathbf{x})$ to be fully linear on $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_{k+1}\| \leq \Delta_{k+1}\}$ using Algorithm 9.2.

al. [41] on a set of candidate length scales, $\xi_i \in \{\xi_1, \ldots, \xi_n\}$. A fully linear model is constructed for each candidate length scale, and the likelihood of each length scale is computed. The trust region algorithm then uses the surrogate model constructed with ξ^*, where ξ^* is chosen as the value of ξ corresponding to the maximum likelihood. This maximum likelihood approach can improve the model calibration, and also provides flexibility in selecting sample points in the extension to the case when there are multiple lower-fidelity models (as will be discussed in Section 9.6).

Algorithm 5: Create Fully Linear Models Allowing Maximum Likelihood Correlation Lengths

1: Compute the likelihood for all RBF correlation lengths, $\xi_i \in \{\xi_1, \ldots, \xi_n\}$ with steps 2-5.
2: Generate a set of $n+1$ affinely independent points in the vicinity of the trust region:
  2a: Set $\mathbf{y}_1 = \mathbf{0}$, and add $\mathbf{y}_1$ to the set of calibration vectors $\mathcal{Y}$.
  2b: Randomly select any high-fidelity sample point, $\mathbf{d}_2$, within the current trust region and add the vector $\mathbf{y}_2 = \mathbf{d}_2 - \mathbf{x}_k$ to $\mathcal{Y}$.
  2c: For all unused high-fidelity sample points within the current trust region, add the vector $\mathbf{y} = \mathbf{d}_j - \mathbf{x}_k$ to $\mathcal{Y}$ if the projection of $\mathbf{y}$ onto the nullspace of the span of the vectors in the current $\mathcal{Y}$ is greater than $\theta_1 \Delta_k$, $0 < \theta_1 < 1$.
  2d: If fewer than $n+1$ vectors are in calibration set, repeat step 2c allowing a larger search region of size $\theta_3 \Delta_k$, $\theta_3 > 1$.
  2e: While fewer than $n+1$ vectors are in $\mathcal{Y}$,
  2f: Evaluate the high-fidelity function at a point within the nullspace of the span of the vectors in $\mathcal{Y}$ and add $\mathbf{y} = \mathbf{d} - \mathbf{x}_k$ to $\mathcal{Y}$.
  2g: Store the results of all high-fidelity function evaluations in the database.
3: Consider the remaining unused high-fidelity sample points within a region centered at the current iterate with size $\theta_4 \Delta_k$, $\theta_4 > 1$. Add points so that the total number of interpolation points does not exceed p_{max}, the RBF coefficients remain bounded, and the surrogate model is fully linear (using, for example, the AddPoints algorithm of Wild *et al.* [43]).
4: Compute the RBF coefficients using the QR factorization technique of Wild *et al.* [41].
5: If only $n+1$ vectors are in the calibration set, $\mathcal{Y}$, assign the likelihood of the current correlation length, ξ_i, to $-\infty$. Otherwise compute the likelihood of the RBF interpolation using standard methods[3, 4].
6: Select the ξ_i with the maximum likelihood.
  6a: If the maximum likelihood is $-\infty$ choose the largest ξ_i. This model corresponds to a linear regression of the high-fidelity function at the calibration points included in $\mathcal{Y}$, but still satisfies conditions for convergence.
7: Return the set of calibration vectors $\mathcal{Y}$, RBF coefficients, and updated database of high-fidelity function evaluations.

9.5 Multifidelity Optimization Examples

This section demonstrates the multifidelity optimization scheme for two examples. The first is an analytical example considering the Rosenbrock function and the second is a supersonic airfoil design problem.

9.5.1 Rosenbrock Function

The first example multifidelity optimization example is the Rosenbrock function,

$$\min_{\mathbf{x} \in \mathbb{R}^2} f_{\text{high}}(\mathbf{x}) = \left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2 . \tag{9.20}$$

The minimum of the Rosenbrock function is at $x^* = (1, 1)$ and $f(x^*) = 0$. Table 9.2 presents the number of high-fidelity function evaluations required to optimize the Rosenbrock function using a variety of low-fidelity functions. All of the low-fidelity functions have a different minimum than the Rosenbrock function, with the exception of the case when the low-fidelity function is set equal to the Rosenbrock function, corresponding to a

perfect low-fidelity function. Convergence results are presented for the case when the low-fidelity function is parabolic, $f_{\text{low}}(\mathbf{x}) = x_1^2 + x_2^2$, in Figure 9.2(a), and a surface plot of the Rosenbrock function and this low-fidelity function is shown in Figure 9.2(b). For all of the examples in this section the optimization parameters used are given in Table 9.1 and are discussed in the remainder of this subsection.

Parameter	Description	value
$\phi(r)$	RBF Correlation	e^{-r^2/ξ^2}
ξ	RBF spatial correlation length	See Table 9.2
Δ_0	Initial trust region size	$\max[10, \|\mathbf{x}_0\|_\infty]$
$\Delta_{\max}$	Maximum trust region size	$10^3\Delta_0$
ϵ, ϵ_2	Termination tolerances	5×10^{-4}
γ_0	Trust region contraction ratio	0.5
γ_1	Trust region expansion ratio	2
η	Trust region expansion criterion	0.2
α	Trust region contraction ratio used in convergence check	0.9
κ_{FCD}	Fraction of Cauchy decrease requirement	10^{-4}
$p_{\max}$	Maximum number of calibration points	50
θ_1	Minimum projection into null-space of calibration vectors	10^{-3}
θ_2	RBF coefficient conditioning parameter	10^{-4}
θ_3	Expanded trust-region size to find basis, $\theta_3\Delta_k$	10
θ_4	Maximum calibration region size, $\theta_4\Delta_k$	10
δx	Finite difference step size	10^{-6}

Table 9.1. Optimization parameters used in the Rosenbrock function demonstration.

We use a Gaussian RBF, $\phi(r) = e^{-r^2/\xi^2}$, to build the RBF error interpolation and two methods of selecting the spatial correlation length, ξ. The first method is to fix a value of ξ, and the second approach is based on Kriging methods, which assume interpolation errors are normally distributed and maximize the likelihood that the RBF surface predicts the function [3, 4]. To save computation time, the maximum likelihood correlation length is estimated by examining 10 correlation lengths between 0.1 and 5.1, and the correlation length that has the maximum likelihood is chosen. If all correlation lengths have the same likelihood, the maximum correlation length is used. The results in Table 9.2 show that the correlation length has a moderate impact on the convergence rate of the method. For this problem, using either $\xi = 2$ or ξ^*, the correlation length that maximizes the likelihood at each trust-region iteration, leads to the best result.

The results in Table 9.2 show that using a multifidelity framework can reduce the number of high-fidelity function calls. As a baseline, the average number of function calls for a quasi-Newton method directly optimizing the Rosenbrock function is 69. The Bayesian calibration approach uses between 5 and 180 high-fidelity function evaluations depending on the quality of the low-fidelity model. The worst case, 180 high-fidelity function evaluations, corresponds to not having a lower-fidelity model and simply approximating the function with a RBF interpolation. The best case, 5 high-fidelity evaluations, corresponds to the case when the low-fidelity function exactly models the high-fidelity function. With a moderately good low-fidelity function (e.g., a 4^{th} degree polynomial), the multifidelity method performs similarly to the quasi-Newton method. Clearly the performance of this method compared to conventional optimization methods depends considerably on the quality of the low-fidelity function used. However, when this method is compared with other multifidelity methods such as the first-order consistent trust-region approaches of Alexandrov *et al.*[32], it uses fewer high-fidelity function evaluations. Results for two first-order consistent trust-region methods are presented in Table 9.2 along with the results of the Bayesian calibration method.

Low-Fidelity Function	$\xi = 1$	$\xi = 2$	$\xi = 3$	$\xi = 5$	ξ^*	Mult.-Corr.	add-Corr.
$f_{\text{low}}(\mathbf{x}) = 0$	148	107	177	223	178	289	503
$f_{\text{low}}(\mathbf{x}) = x_1^2 + x_2^2$	129	77	106	203	76	312	401
$f_{\text{low}}(\mathbf{x}) = x_1^4 + x_2^2$	74	74	73	87	65	171	289
$f_{\text{low}}(\mathbf{x}) = f_{\text{high}}(\mathbf{x})$	5	5	5	5	7	7	6
$f_{\text{low}}(\mathbf{x}) = -x_1^2 - x_2^2$	195	130	132	250	100	352	fail

Table 9.2. Table of average number of function evaluations required to minimize the Rosenbrock function, Eq. 9.20, from a random initial point on $x_1, x_2 \in [-5, 5]$. Results for a selection of Gaussian radial bases function spatial parameters, ξ, are shown. ξ^* corresponds to optimizing the spatial parameter according to a maximum likelihood criteria [4]. Also included are the number of function evaluations required using first-order consistent trust region methods with a multiplicative correction and an additive correction. For a standard quasi-Newton method the average number of function evaluations is 69.

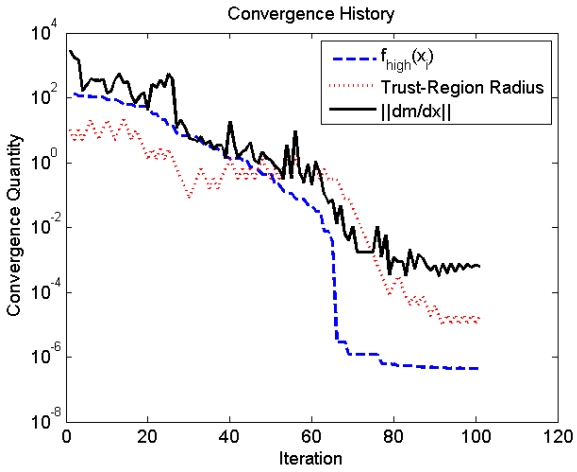
The first uses a multiplicative correction defined in Ref. [32], while the second uses an additive correction. The first-order consistent methods require $n + 1$ high-fidelity function evaluations to estimate the high-fidelity gradient at each $\mathbf{x}_k$.

For this simple high-fidelity function, the first-order consistent trust-region methods and the quasi-Newton method require less than half the wall-clock time that the Bayesian calibration method requires. Building the RBF models requires multiple matrix inversions, each of which requires $\mathcal{O}(p_{\max}(p_{\max} + n + 1)^3)$ operations, where n is the number of design variables and $p_{\max}$ is the user-set maximum number of calibration points allowed in a model. Accordingly, the Bayesian calibration method is only recommended for high-fidelity functions that are expensive compared to the cost of repeatedly solving for RBF coefficients.

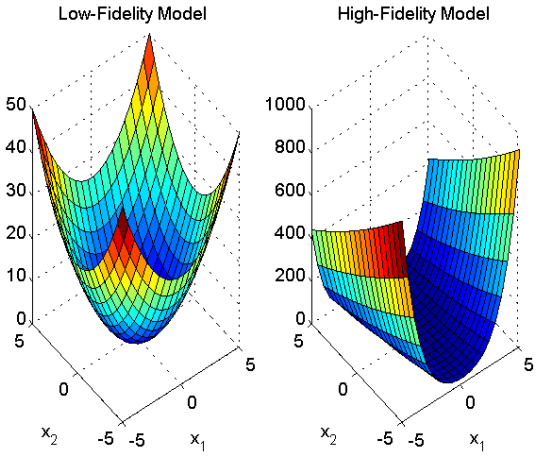
As with any optimization algorithms, tuning parameters can affect performance significantly; however, the best choices for these tuning parameters can be highly problem dependent. A sensitivity study measured the impact of algorithm parameters on the number of high-fidelity function evaluations for the Rosenbrock example using $f_{\text{low}}(\mathbf{x}) = x_1^2 + x_2^2$ as the low-fidelity function (Figure 9.2). For all of these tests, one parameter is varied and the remainder are all set to the values in Table 9.1. The conclusions drawn are based on the average of at least ten runs with random initial conditions on the interval $x_1, x_2 \in [-5, 5]$. While these conclusions may provide general useful guidance for setting algorithm parameters, similar sensitivity studies are recommended for application to other problems.

The parameter η is the trust region expansion criterion, where the trust region expands if $\rho_k \geq \eta$ and contracts otherwise. The sensitivity results show that lower values of η have the fewest high-fidelity function calls, and any value $0 \leq \eta \leq 0.2$ performs well. For the trust region expansion ratio, γ_1, the best results are at $\gamma_1 \approx 2$, and high-fidelity function evaluations increase substantially for other values. Similarly, for the contraction ratio, γ_0, the best results are observed at $\gamma_0 \approx 0.5$, with a large increase in high-fidelity function evaluations otherwise. For the fraction of Cauchy decrease, κ_{FCD}, the results show the number of high-fidelity evaluations is fairly insensitive to any value $0 < \kappa_{FCD} < 10^{-2}$. Similarly, for the trust-region contraction ratio used in the algorithm convergence check, α, the number of high-fidelity function evaluations is insensitive to any value $0.5 < \alpha < 0.95$.

The method of Wild *et al.* to generate fully linear models requires four tuning parameters, θ_1, θ_2, θ_3, and θ_4[41, 43]. The parameter θ_1 $(0 < \theta_1 < 1)$ determines the acceptable points when finding the affinely independent basis in the vicinity of the trust region in Algorithm 5. As θ_1 increases, the calibration points added to the basis must have a larger projection into the null-space of the current basis, and therefore fewer points are admitted to the basis. We find for the Rosenbrock example that the fewest function evaluations occurs with $\theta_1 \approx 10^{-3}$; however, for any value of θ_1 within two orders of magnitude of



(a) Convergence plot.



(b) Parabolic low-fidelity function and the Rosenbrock function.

Figure 9.2. Rosenbrock function and a similar low-fidelity model.

this value, the number of function evaluations increases by less than 50%. The second parameter, θ_2 ($0 < \theta_2 < 1$), is used in the AddPoints algorithm of Wild *et al.* [43] to ensure that the RBF coefficients remain bounded when adding additional calibration points. The number of allowable calibration points increases as θ_2 decreases to zero; however, the matrix used to compute the RBF coefficients also becomes more ill-conditioned. For our problem, we find that $\theta_2 \approx 10^{-4}$ enables a large number of calibration points while providing acceptable matrix conditioning. The two other parameters, θ_3 and θ_4, used in the calibration point selection algorithm, are significant to the algorithm's performance. The parameter θ_3 ($\theta_3 > 1$) is used if $n+1$ affinely independent previous high-fidelity sample points do not exist within the current trust region. If fewer than $n+1$ points are found, the calibration algorithm allows a search region of increased size $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_k\| \leq \theta_3 \Delta_k\}$ in order to find $n+1$ affinely independent points prior to evaluating the high-fidelity function in additional locations. The results show that the number of function calls is insensitive to θ_3 for $1 < \theta_3 \leq 10$, with $\theta_3 \approx 3$ yielding the best results. The parameter θ_4 ($\theta_4 > 1$) represents the balance between global and local model calibration, as it determines how far points can be from the current iterate, $\mathbf{x}_k$, and still be included in the RBF interpolation. Points that lie within a region $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_k\| \leq \theta_4 \Delta_k\}$ are all candidates to be added to the interpolation. Calibration points outside of the trust region will affect the shape of the model within the trust region, but the solution to the subproblem must lie within the current trust region. The results of our analysis show that $\theta_4 \approx 10$ is the best value, with the number of high-fidelity function calls increasing substantially if $\theta_4 < 5$ or $\theta_4 > 15$.

9.5.2 Supersonic Airfoil Optimization

As an engineering example, the supersonic airfoil problem presented in Section 2.1 is used to demonstrate how the algorithm performance scales with the number of design variables. The optimization will seek to find the angle of attack and set of surface splines points to minimize drag at Mach 1.5. The upper and lower surfaces will each have the same number of spline points. In this test case, the linear supersonic panel method is used as the low-fidelity function and shock-expansion theory is used as the high-fidelity function. For supersonic flow, a zero thickness airfoil will have the minimum drag, so the airfoil must be constrained to have a thickness to chord ratio greater than 5%. This is accomplished by adding a penalty function, so that if the maximum thickness of the airfoil is less than 5%, the penalty term $1000(t/c - 0.05)^2$ is added to the drag. A similar penalty is added if the thickness anywhere on the airfoil is less than zero.

The optimization parameters used by this method are the same as in Table 9.1, with the exception that the RBF correlation length is either $\xi = 2$ or optimized at each iteration. A consecutive step size of less than 5×10^{-6} is an additional termination criteria for all of the multifidelity methods compared. The number of high-fidelity function evaluations required to optimize the airfoil for each of the methods using a different number of design variables is presented in Figure 9.3. The airfoil optimization shows that both the first-order consistent methods and the RBF calibration method perform significantly better than the quasi-Newton method. This is largely because the multifidelity methods have a significant advantage over the single fidelity methods in that the physics-based low-fidelity model is a reasonable representation of the high-fidelity model. However, the RBF calibration approach uses less than half the number of function evaluations than the multiplicative-correction approach. In addition, the additive correction outperforms the multiplicative correction for this problem, but the RBF calibration outperformed it. The method of maximizing the likelihood of the RBF calibration performs slightly better than just using a fixed correlation length.

As a second test case, the panel method is used as a low-fidelity function to minimize the drag of an airfoil with Cart3D as the high-fidelity function. Cart3D has an adjoint-

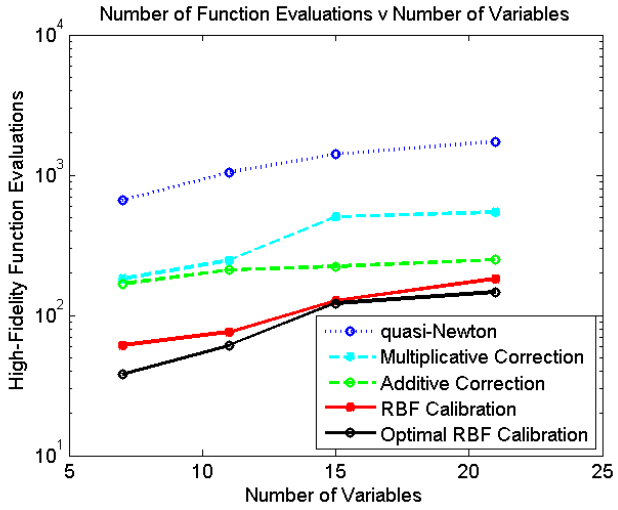


Figure 9.3. Number of shock-expansion theory evaluations required to minimize the drag of a supersonic airfoil verse the number of parameters. The low-fidelity model is the supersonic panel method.

based mesh refinement, which ensures the error caused by the discretization is less than a tolerance. Accordingly, the drag computed by Cart3D is not Lipschitz continuous because there is finite precision and a finite difference estimate of the derivative only measures numerical noise. However, in the calibration algorithm the trust region radius converges to zero, which forces a small step size and this is a supplemental termination criteria. So the method is not provably convergent in this case, but it still does converge to the correct solution. On average, the airfoil parameterized with 11 variables requires 88 high-fidelity (Cart3D) function evaluations. A comparison of the optimum airfoils from the panel method, shock-expansion theory, and Cart3D is presented in Figure 9.4.

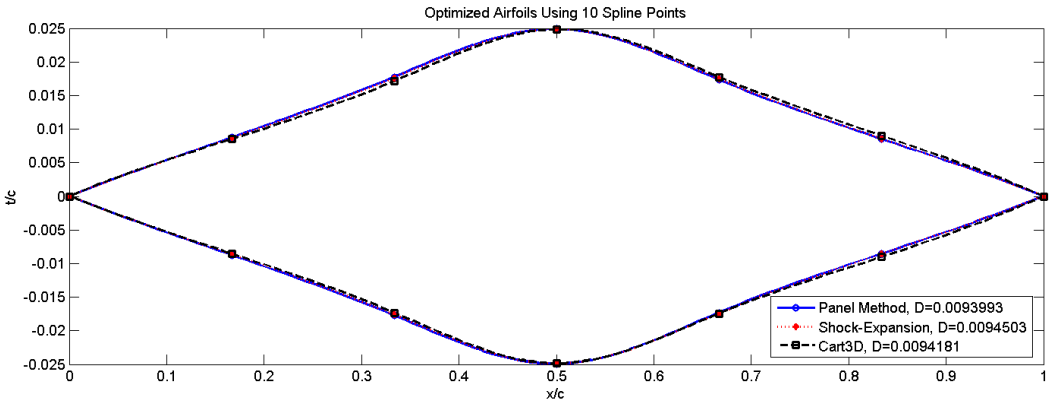


Figure 9.4. Minimum drag airfoils from each of the three analysis models. The panel method airfoil is generated by a quasi-Newton method, but the shock-expansion and Cart3D airfoils are generated with this RBF calibration method using the panel method as a low-fidelity model.

9.6 Combining Multiple Fidelity Levels

This section addresses how the radial basis function interpolation technique can be extended to optimize a function when there are multiple lower-fidelity functions. For instance, consider the case when our goal is to find the $\mathbf{x}^*$ that minimizes $f_{\text{high}}(\mathbf{x})$, and there exists two or more lower-fidelity functions, an intermediate-fidelity, $f_{\text{med}}(\mathbf{x})$, and a low-fidelity, $f_{\text{low}}(\mathbf{x})$. The typical approach to solve this problem is to nest the lower-fidelity function; that is, to use the intermediate fidelity function as the low-fidelity model of the high-fidelity function, and to use the lowest-fidelity function as the low-fidelity model of the intermediate-fidelity function. To do this, two calibration models are needed,

$$f_{\text{high}}(\mathbf{x}) \approx f_{\text{med}}(\mathbf{x}) + e_{\text{med}}(\mathbf{x}) \tag{9.21}$$

$$f_{\text{med}}(\mathbf{x}) \approx f_{\text{low}}(\mathbf{x}) + e_{\text{low}}(\mathbf{x}). \tag{9.22}$$

In the nested approach, the high-fidelity optimization is performed on the approximate high-fidelity function, which is the medium-fidelity function plus the calibration model e_{med}. However, to determine the steps in that optimization, another optimization is performed on a lower-fidelity model. This low-fidelity optimization is performed on the model

$$m(\mathbf{x}) \approx f_{\text{low}}(\mathbf{x}) + e_{\text{low}}(\mathbf{x}) + e_{\text{med}}(\mathbf{x}), \tag{9.23}$$

but only the low-fidelity calibration model e_{low} is adjusted.

A problem with the nested approach is that on the low-fidelity optimization, a constrained optimization that uses model calibration techniques must be performed due to the trust region at the higher level. Moreover, this method likely requires a considerable reduction in the resources required to run each of the lower-fidelity models. The reason for this is that in order to take one step in the high-fidelity space, an optimization is required on the medium-fidelity function. However, for each step in medium-fidelity space, an optimization is required on the lower-fidelity function. So, if an optimization routine requires 50 function evaluations to converge, then for one high-fidelity step, 50 intermediate-fidelity evaluations will be required, and 2,500 lower-fidelity evaluations will be required. If the number of optimization iterations is of the same order at each level, then there will be an exponential scaling in the number of function evaluations required between fidelity levels.

An alternative to nesting multiple lower-fidelity functions is to use a maximum likelihood estimator to estimate the high-fidelity function. Since the multifidelity optimization method discussed in this chapter uses radial basis function interpolants, a variance estimate of the interpolation error can be created using standard Gaussian process techniques [3, 4]. In the case of multiple fidelity levels, the calibration of $f_{\text{high}}(\mathbf{x}) \approx f_{\text{low}}(\mathbf{x}) + e(\mathbf{x})$ is modified so that the error model, $e(\mathbf{x})$ is treated as the mean of a Gaussian process, and the error model also includes a variance model. In this case, the error model, normally distributed with mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$, is written as $\mathcal{N}\left(\mu(\mathbf{x}), \sigma^2(\mathbf{x})\right)$. In the case of multiple lower-fidelity functions there are multiple estimates of the high-fidelity function, for example,

$$f_{\text{high}}(\mathbf{x}) \approx f_{\text{med}}(\mathbf{x}) + \mathcal{N}\left(\mu_{\text{med}}(\mathbf{x}), \sigma^2_{\text{med}}(\mathbf{x})\right) \tag{9.24}$$

$$f_{\text{high}}(\mathbf{x}) \approx f_{\text{low}}(\mathbf{x}) + \mathcal{N}\left(\mu_{\text{low}}(\mathbf{x}), \sigma^2_{\text{low}}(\mathbf{x})\right). \tag{9.25}$$

From these two or more models, a maximum likelihood estimate of the high-fidelity function weights each prediction according to a function of the variance estimates. The high-fidelity maximum likelihood estimate has a mean f_{est}, given by

$$\begin{aligned} f_{\text{est}}(\mathbf{x}) = \left(f_{\text{med}}(\mathbf{x}) + \mu_{\text{med}}(\mathbf{x})\right) &\left[\frac{\sigma^2_{\text{low}}(\mathbf{x})}{\sigma^2_{\text{low}}(\mathbf{x}) + \sigma^2_{\text{med}}(\mathbf{x})}\right] \\ &+ \left(f_{\text{low}}(\mathbf{x}) + \mu_{\text{low}}(\mathbf{x})\right)\left[\frac{\sigma^2_{\text{med}}(\mathbf{x})}{\sigma^2_{\text{low}}(\mathbf{x}) + \sigma^2_{\text{med}}(\mathbf{x})}\right] \end{aligned} \tag{9.26}$$

The estimate of the high-fidelity function also has a variance, σ^2_{est}, which is less than either of the variances of the lower-fidelity models since

$$\frac{1}{\sigma^2_{\text{est}}(\mathbf{x})} = \frac{1}{\sigma^2_{\text{low}}(\mathbf{x})} + \frac{1}{\sigma^2_{\text{med}}(\mathbf{x})}. \tag{9.27}$$

A schematic of the behavior of this maximum likelihood estimate is shown in Figure 9.5. In the first case with two similar models, the combined estimate has a similar mean with a reduced variance. In the second case with two dissimilar estimates, the combined estimate has the average mean of the two models again with lower variance. In the third case when one model has a considerably smaller variance than the other model, the combined estimate has a similar mean and slightly reduced variance than the model with the lower variance. Accordingly, the maximum likelihood estimate is the best probabilistic guess of the high-fidelity function at a non-calibrated point.

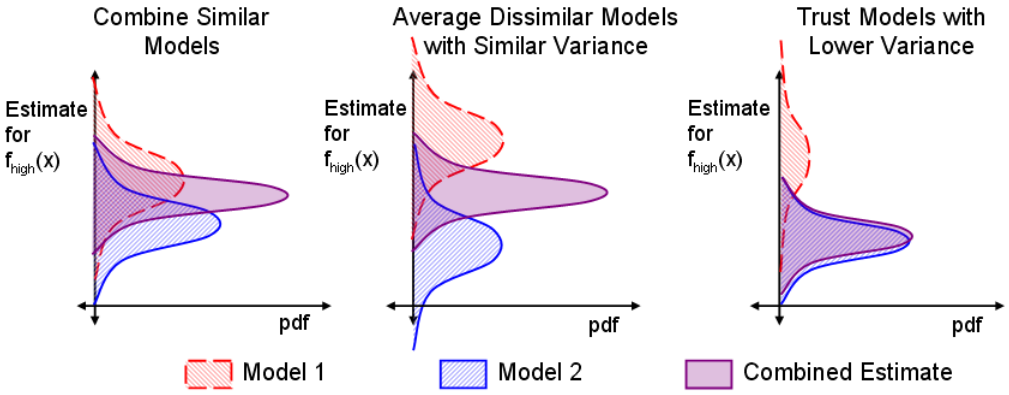


Figure 9.5. Behavior of the combined maximum likelihood estimate given the behavior of the individual estimates.

This method provides flexibility while still being provably convergent to a high-fidelity optimum using our multifidelity optimization approach. The requirements for convergence are that the surrogate model upon which the optimization is performed be smooth and exactly interpolate the function at the necessary calibration points. Using this maximum likelihood estimator, only one of the lower-fidelity functions needs to be sampled at the calibration points because at a calibration point an individual Gaussian process model has zero variance. Accordingly, at that calibration point the model is known to be correct, so that prediction is trusted implicitly and the other lower-fidelity information is not used. Also, with a smooth Gaussian process covariance function, the variance estimate will be a smooth function. This makes the model of the high-fidelity function, $f_{\text{est}}(\mathbf{x})$, a smooth model that satisfies the optimization algorithm requirements for convergence to a high-fidelity optimum. Therefore, the user may choose any method of selecting which lower-fidelity models are calibrated at a required calibration point, as only one needs to be. For example, the calibration procedure could choose a ratio, such as one intermediate-fidelity update for every three low-fidelity updates, or simply update both the intermediate-fidelity and low-fidelity models each time a new calibration point is needed.

Optimization results show that the nesting approach suffers from poor scaling between fidelity levels and that the maximum likelihood approach speeds convergence of our multifidelity optimization method even if the lowest-fidelity function is a poor representation of the high-fidelity function. In all examples presented, the calibration strategy employed for the maximum likelihood method is to update all lower-fidelity models whenever the optimization method requires a new calibration point.

The first example is an optimization of the Rosenbrock function with two parabolic lower-fidelity functions. The number of required function evaluations for each fidelity level is presented in Table 9.3. Using the maximum likelihood approach, the number of high-fidelity function evaluations has been reduced by 34%, and the number of combined lower-fidelity evaluations has been reduced by 27%. However, combining the multiple lower-fidelity functions through nesting leads to a large increase in the number of function evaluations at each level.

Method	$(x_2 - x_1^2)^2 + (1 - x_1)^2$	$(x_1 - 1)^2 + x_2^2$	$x_1^2 + x_2^2$
Two-Fidelities	87	0	6975
Max. Likelihood	57	2533	2533
Nested	137	4880	50455

Table 9.3. Number of function calls required to optimize the Rosenbrock function using multiple lower-fidelity functions. The maximum likelihood approach requires the least high-fidelity function evaluations to converge and the nested approach the most.

The second example is to optimize a supersonic airfoil for minimum drag with respect to an Euler code, Cart3D. Two lower-fidelity methods are used: shock-expansion theory and a panel method. These results also show that the maximum likelihood approach converges faster and with fewer calibration points than the original multifidelity method using only the panel method. The nesting approach failed to converge as the step size required in the intermediate-fidelity optimization became too small. The likely cause of this is that the adjoint-based mesh refinement used in Cart3D allows numerical oscillations in the output functional at a level that is still significant in the optimization, and this makes the necessary calibration surface non-smooth. The lack of smoothness violates the convergence criteria of this method.

Method	Cart3D	Shock-expansion	Panel Method
Two-Fidelities	88	0	47679
Max. Likelihood	66	23297	23297
Nested	66*	7920*	167644*

Table 9.4. Number of function calls required to optimize an airfoil for minimum drag using the Euler equations (Cart3D) with multiple lower-fidelity models. An asterisk indicates that solution was not converged due to numerical limitations.

The final example demonstrates that the maximum likelihood approach can still benefit from a poor low-fidelity model. The results in Table 9.5 are for minimizing the drag of a supersonic airfoil using shock-expansion theory, with the panel method as an intermediate-fidelity function; however, unlike the preceding example, the lowest-fidelity model is quite poor and uses the panel method only on the camberline of the airfoil. Using this method, any symmetric airfoil at zero angle of attack has no drag and many of the predicted trends are incorrect compared to the panel method or shock-expansion theory. The optimization results show an important benefit of this maximum likelihood approach: even adding this additional bad information, the number of high-fidelity function calls has been reduced by 33%, and the number of intermediate-fidelity function calls has decreased by 31%. An additional point of note is the magnitude to which the nested approach suffers by adding poor low-fidelity information. In most test problems, the nested optimization was terminated due to an exceptionally large number of function evaluations. The results presented are the minimum number of function evaluations the nested approach required to converge.

Method	Shock-expansion	Panel Method	Camberline
Two-Fidelities	126	43665	0
Max. Likelihood	84	30057	30057
Nested	212*	59217*	342916*

Table 9.5. Number of function calls required to optimize an airfoil for minimum drag using shock-expansion theory with multiple lower-fidelity models. An asterisk indicates a minimum number of function evaluations as opposed to an average value from random starting points.

9.7 Summary

The method discussed in this chapter is a provably convergent multifidelity optimization method that does not require computation of derivatives of the high-fidelity function. The optimization results show that this method reduces the number of high-fidelity function calls required to find a local minimum compared with other state-of-the-art methods. The method creates surrogate models that retain accurate local behavior while also capturing some global behavior of the high-fidelity function. However, a downfall of the method is that the overhead increases dramatically with the number of design variables and the number of calibration points used to build the radial basis function model. Accordingly, this approach is only recommended for high-fidelity functions that require a considerable wall-clock time. This chapter has also shown that a multifidelity optimization method based on a maximum likelihood estimator is an effective way of combining many fidelity levels to optimize a high-fidelity function. The maximum likelihood estimator permits flexible sampling strategies among the low-fidelity models and is robust with respect to poor low-fidelity estimates.

10 High-Fidelity-Gradient-Free Constrained Optimization

This chapter will extend the unconstrained local optimization multifidelity optimization method presented in Chapter 9 to address constrained multifidelity optimization problems. The method presented is a summary of the method that will be presented at the 13^{th} AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference [44]. The algorithm minimizes a high-fidelity objective function subject to a high-fidelity constraint and other simple constraints. The algorithm never computes the gradient of a high-fidelity function; however, it demonstrates convergence using sensitivity information from the calibrated low-fidelity models, which are constructed to have negligible error in a neighborhood around the solution. The method is demonstrated for aerodynamic shape optimization using the airfoil design problem in Section 2.1 and shows an 80% reduction in the number of high-fidelity analyses compared with a single-fidelity sequential quadratic programming formulation and a similar number of high-fidelity analyses compared with a multifidelity trust-region algorithm that estimates the high-fidelity gradient using finite differences.

10.1 Motivation

A powerful technique commonly used for solving constrained multifidelity optimization problems is trust-region model management, where at all design iterates the gradients of low-fidelity objective function and constraints are either scaled or shifted such that they match the value and gradient of their high-fidelity counterpart [31, 32]. This technique is provably convergent to a locally optimal high-fidelity design and in practice provides a 2–3 fold reduction in the number of high-fidelity function calls [31]. However, it is common in engineering design problems that the gradient of a high-fidelity function is unavailable and estimating the gradient is unreliable or too expensive. In these cases, multifidelity methods such as Efficient Global Optimization (EGO) [1] or Surrogate Management Framework (SMF) [45] can be used. In EGO a Gaussian process regression model is fit to the high-fidelity objective function. The mean of the Gaussian process interpolates the value of the high-fidelity function, whereas, the mean square error of the Gaussian process models the uncertainty in the high-fidelity function value. This error is zero at all locations where the value of the high-fidelity function is known and increases with distance away from sample points. Optimization is then performed on the Gaussian process model, and the high-fidelity function is sampled at locations likely to reduce to the value of the function over the current observed minimum. This technique works well in practice, can be made globally convergent [1], can be used in a multifidelity setting using Bayesian model calibration methods[34, 35, 36], and does not require a high-fidelity derivative estimate. However, the method may be globally biased and attempt to explore the entire design space as opposed to simply reducing the objective function. In addition, the method has been shown to be sensitive to both the initial high-fidelity samples [2] and to the exact metric of selecting points likely to improve the high-fidelity function value [46]; moreover, methods to handle constraints are still somewhat heuristic [1, 18, 46]. SMF is a derivative-free pattern-search method augmented with a prediction of a locally optimal design from a surrogate model. The underlying pattern-search method ensures convergence, so a broad range of surrogate models are allowable. Of specific interest is conformal space mapping where a low-fidelity function is calibrated to the high-fidelity function at locations where the value is known [33]. In SMF, constraints may be

handled directly on the surrogate models either with gradient information or with penalties, and for the gradient-free high-fidelity pattern-search can be handled with an augmented Lagrangian[47, 48, 49], exact penalty method[50], or constraint filtering[51].

In many cases, sensitivity information is very useful and can accelerate convergence of a derivative-based optimization method towards a local optimum. Accordingly, for multifidelity optimization of a function where the gradient is known, trust-region model management is likely the best method to find a locally optimal design. However, if high-fidelity gradient information is unknown, trust-region model management requires finite differences to produce a surrogate model that satisfies first-order requirements at each design iterate. Finite differences may not always be a viable choice due to noise in the function evaluations, number of design variables, design locations where the objective function fails to exist, or simply computational requirements. In these situations gradient-free multifidelity techniques such as EGO or SMF could be considered, since they converge quickly if the model calibration methods are robust. This chapter discusses the question: can a low-fidelity surrogate model be calibrated in a sufficiently robust manner as to use low-fidelity sensitivity information to optimize a high-fidelity system with a limited number of high-fidelity evaluations? The answer to this question requires finding a surrogate modeling technique that can capture enough high-fidelity behavior to prove convergence to a high-fidelity optimum, using as few high-fidelity function evaluations as a finite difference based calibration approach.

Creating a fully linear model is one possible surrogate modeling technique to ensure low-fidelity sensitivity information predicts high-fidelity sensitivities well. A fully linear model, formally defined in the next section, establishes Lipschitz-type error bounds between the high-fidelity function and the surrogate model. This ensures that the error between the high-fidelity gradient and surrogate model gradient is locally bounded without ever calculating the high-fidelity gradient. Conn *et al.* showed polynomial interpolation models can be made fully linear, provided the interpolating set satisfied certain geometric requirements [39], and further developed an unconstrained gradient-free optimization technique using fully linear models [40, 52]. Wild *et al.* demonstrated that a radial basis function interpolation could satisfy the requirements for a fully linear model and be used in Conn's derivative-free optimization framework [41, 42, 43]. March and Willcox generalized this method to the cases with arbitrary low-fidelity functions or multiple low-fidelity functions using Bayesian model calibration methods typically associated with EGO [29]. This chapter demonstrates how fully linear models can be used for constrained optimization of computationally expensive functions when their derivatives are not available. The constraints are partitioned into constraints with and without available derivatives. The constraints without available derivatives are approximated with multifidelity methods; whereas the other constraints are handled either implicitly with a penalty method or explicitly. Two constrained multifidelity methods are presented, the first optimizes a high-fidelity objective function subject to constraints with available derivatives. The second formulation optimizes a high-fidelity objective function subject to high-fidelity constraints and constraints with available derivatives.

Section 10.2 of this chapter presents the derivative-free method to optimize a high-fidelity objective function subject to constraints with available derivatives. Fully linear surrogate models of the objective function are minimized within a trust-region setting until convergence to a high-fidelity optimum is demonstrated. Section 10.3 presents a technique for minimizing a high-fidelity objective function subject to both constraints with available derivatives and computationally expensive constraints with unavailable derivatives. This algorithm finds a feasible point using the first formulation to reduce the value of the high-fidelity constraint subject to the constraints with available derivatives. After a feasible point is found, the high-fidelity objective is minimized using fully linear surrogate models for both the high-fidelity objective and high-fidelity constraint. Only points that maintain feasibility are accepted. Section 10.4 presents an aerodynamic shape optimization problem to

demonstrate the proposed multifidelity optimization techniques and compares the number of high-fidelity function evaluations with other methods. The summar in Section 10.5 discusses extensions of the method to the cases when there are multiple lower-fidelity models or when constraints are hard (when the objective function fails to exist if the constraints are violated).

10.2 Constrained Optimization of a Multifidelity Objective Function

We consider a setting where we have two (or more) models that represent the physical system of interest: a high-fidelity function that accurately estimates system metrics of interest but is expensive to evaluate, and a low-fidelity function with lower accuracy but cheaper evaluation cost. We define our high-fidelity function as $f_{\text{high}}(\mathbf{x})$ and our low-fidelity function as $f_{\text{low}}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$ is the vector of n design variables. Our goal is to solve a constrained optimization problem without ever directly computing or estimating the gradient of the high-fidelity function, but generating sensitivity information from the low-fidelity function to reduce the required number of high-fidelity evaluations. The minimization problem considered is to minimize the high-fidelity objective function subject to equality constraints, $h(\mathbf{x})$, and inequality constraints $g(\mathbf{x})$,

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f_{\text{high}}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0, \end{aligned} \tag{10.1}$$

where we assume gradient information from $h(\mathbf{x})$ and $g(\mathbf{x})$ is available or can be estimated accurately. To establish convergence it is necessary to assume that all of the functions in Eq. 10.1 and the low-fidelity models are twice continuously differentiable, are Lipschitz continuous, and have Lipschitz continuous first-derivatives. In addition we will assume that the solution to Eq. 10.1, $\mathbf{x}^*$, is feasible and is a regular point (satisfies strict linear independent constraint qualification).

10.2.1 Trust-region Model Management

From an initial design vector $\mathbf{x}_0$, the trust-region method generates a sequence of design vectors that each reduce a merit function consisting of the high-fidelity function value and penalized constraint violation, where we denote $\mathbf{x}_k$ to be this design vector on the kth trust-region iteration. Following the general Bayesian calibration approach in Ref. [35], we define $e_k(\mathbf{x})$ to be a model of the error between the high- and low-fidelity functions on the kth trust-region iteration, and we construct a surrogate model $m_k(\mathbf{x})$ for $f_{\text{high}}(\mathbf{x})$ as

$$m_k(\mathbf{x}) = f_{\text{low}}(\mathbf{x}) + e_k(\mathbf{x}). \tag{10.2}$$

We define the trust region at iteration k, $\mathcal{B}_k$, to be the region centered at $\mathbf{x}_k$ with size Δ_k,

$$\mathcal{B}_k = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}, \tag{10.3}$$

where any norm can be used, provided there exist constants c_1 and c_2 such that

$$\|\cdot\|_2 \leq c_1\|\cdot\| \qquad \text{and} \qquad \|\cdot\| \leq c_2\|\cdot\|_2. \tag{10.4}$$

To solve the constrained optimization problem presented in Eq. 10.1 we define a merit function, $\Phi(\mathbf{x}_k, \sigma_k)$, where σ_k is a parameter that must go to infinity as the iteration number k goes to infinity and serves to increase the penalty placed on the constraint violation. We

also must assume that the merit function and the trust-region iterates satisfy the following four properties. First, the merit function with the initial penalty, σ_0, must be bounded from below within a relaxed level-set, $\mathcal{L}(\mathbf{x}_0, \sigma_0)$, defined as

$$\begin{aligned} L(\mathbf{x}_0, \sigma_0) &= \{\mathbf{x} \in \mathbb{R}^n : \Phi(\mathbf{x}, \sigma_0) \leq \Phi(\mathbf{x}_0, \sigma_0)\} && (10.5) \\ B(\mathbf{x}_k) &= \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_{\max}\} && (10.6) \\ \mathcal{L}(\mathbf{x}_0, \sigma_0) &= L(\mathbf{x}_0, \sigma_0) \bigcup_{\mathbf{x_k} \in L(\mathbf{x}_0, \sigma_0)} B(\mathbf{x}_k), && (10.7) \end{aligned}$$

where $\Delta_{\max}$ is the maximum allowable trust-region size and the relaxed level-set is required because the trust-region algorithm may attempt to evaluate the high-fidelity function at points outside of the level set at $\mathbf{x}_0$. Second, the level sets of $\Phi(\mathbf{x}_k, \sigma_k > \sigma_0)$ must be contained within $L(\mathbf{x}_0, \sigma_0)$. Third, $L(\mathbf{x}_0, \sigma_0)$ must be a compact set, and fourth, all design iterates $\mathbf{x}_k$ must remain within $L(\mathbf{x}_0, \sigma_0)$.

Although other merit functions are possible, we restrict our attention to merit functions based on quadratic penalty functions because it is trivial to show that they are bounded from below if the objective function obtains a finite global minimum. The merit function used in this method is the objective function plus the scaled sum-squares of the constraint violation, where $\mathbf{g}^+(\mathbf{x})$ are the values of the violated inequality constraints,

$$\Phi(\mathbf{x}, \sigma_k) = f_{\text{high}}(\mathbf{x}) + \frac{\sigma_k}{2}\mathbf{h}(\mathbf{x})^T\mathbf{h}(\mathbf{x}) + \frac{\sigma_k}{2}\mathbf{g}^+(\mathbf{x})^T\mathbf{g}^+(\mathbf{x}). \quad (10.8)$$

The parameter σ_k is a penalty weight, which must go to $+\infty$ as the iteration k goes to $+\infty$. Note that when using a quadratic penalty function for constrained optimization, the sequence of iterates generated, $\{\mathbf{x}_k\}$, can either terminate at a feasible regular point at which the Karush-Kuhn-Tucker (KKT) conditions are satisfied, or at a point that minimizes the squared norm of the constraint violation, $\mathbf{h}(\mathbf{x})^T\mathbf{h}(\mathbf{x}) + \mathbf{g}^+(\mathbf{x})^T\mathbf{g}^+(\mathbf{x})$ [53, 54]. Accordingly, we assume that the iterates generated by minimizing the merit function are such that the sequence $\{\mathbf{x}_k\}$ always remains within $L(\mathbf{x}_0, \sigma_0)$ and terminates at a limit point, $\mathbf{x}^*$ that is both regular and feasible.

We now define a surrogate merit function, $\hat{\Phi}(\mathbf{x}, \sigma_k)$, which replaces the objective function with a surrogate model of the expensive objective function.

$$\hat{\Phi}(\mathbf{x}, \sigma_k) = m_k(\mathbf{x}) + \frac{\sigma_k}{2}\mathbf{h}(\mathbf{x})^T\mathbf{h}(\mathbf{x}) + \frac{\sigma_k}{2}\mathbf{g}^+(\mathbf{x})^T\mathbf{g}^+(\mathbf{x}). \quad (10.9)$$

Optimization is performed on this function, and updates to the trust-region are based on how changes in this surrogate merit function compare with changes in the original merit function, $\Phi(\mathbf{x}, \sigma_k)$.

We further require that the surrogate models $m_k(\mathbf{x})$ are *fully linear*, where the following definition of a fully linear model is from Conn *et al.*:

Definition 1. *Let a function $f_{high}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ that is continuously differentiable and has a Lipschitz continuous derivative, be given. A set of model functions $\mathcal{M} = \{m : \mathbb{R}^n \to \mathbb{R}, m \in C^1\}$ is called a fully linear class of models if the following occur:*

There exist positive constants κ_f, κ_g and κ_{blg} such that for any $\mathbf{x} \in L(\mathbf{x}_0, \sigma_0)$ and $\Delta_k \in (0, \Delta_{\max}]$ there exists a model function $m_k(\mathbf{x})$ in $\mathcal{M}$ with Lipschitz continuous gradient and corresponding Lipschitz constant bounded by k_{blg}, and such that the error between the gradient of the model and the gradient of the function satisfies

$$\|\nabla f_{high}(\mathbf{x}) - \nabla m_k(\mathbf{x})\| \leq \kappa_g \Delta_k \quad \forall \mathbf{x} \in \mathcal{B}_k \quad (10.10)$$

and the error between the model and the function satisfies

$$|f_{high}(\mathbf{x}) - m_k(\mathbf{x})| \leq \kappa_f \Delta_k^2 \quad \forall \mathbf{x} \in \mathcal{B}_k. \quad (10.11)$$

Such a model $m_k(\mathbf{x})$ is called fully linear on $\mathcal{B}_k$ [40].

10.2.2 Trust-region Subproblem

At each trust-region iteration a point likely to decrease the merit function is found by solving one of two minimization problems on the fully linear model for a step $\mathbf{s}_k$, on a trust region of size Δ_k:

$$\begin{aligned} \min_{\mathbf{s}_k \in \mathbb{R}^n} \quad & m_k(\mathbf{x}_k + \mathbf{s}_k) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_k + \mathbf{s}_k) = 0 \\ & \mathbf{g}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ & \|\mathbf{s}_k\| \leq \Delta_k, \end{aligned} \tag{10.12}$$

or

$$\begin{aligned} \min_{\mathbf{s}_k \in \mathbb{R}^n} \quad & \hat{\Phi}_k(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) \\ \text{s.t.} \quad & \|\mathbf{s}_k\| \leq \Delta_k. \end{aligned} \tag{10.13}$$

Ideally the first trust-region subproblem, Eq. 10.12, will be used for optimization; however, typical nonlinear programming techniques do not allow a user to specify which constraints must be satisfied, so if some of the constraints are violated it may not be possible to reduce the value of the merit function and guarantee that the trust-region constraint is satisfied. Accordingly, when the current iterate is infeasible, the second trust-region subproblem, Eq. 10.13, must be used. A problem with this subproblem is that the norm of the objective function Hessian grows without bound due to the penalty increasing to infinity. Therefore, a fundamental assumption in trust-region algorithms, that the subproblem have a bounded norm for the Hessian, can become violated. So, the algorithm must be structured in a way that once the design iterate is "close" to feasible, the subproblem in Eq. 10.12 must be used, since the Hessian of this subproblem remains bounded.

We require that for whichever trust-region subproblem is used, the subproblem must be solved such that the 2-norm of the first-order optimality conditions is less than a constant τ_k. This requirement is stated as $\|\nabla_x \mathfrak{L}_k\| \leq \tau_k$, where $\mathfrak{L}_k$ is the Lagrangian for the trust-region subproblem used. There are two requirements for τ_k. First $\tau_k < \epsilon$, where ϵ is the desired termination tolerance for the optimization problem in Eq. 10.1. Second, τ_k must decrease to zero as the number of iterations goes to infinity. Accordingly, we define $\tau_k = \min[\beta\epsilon, c\Delta_k]$, with a constant $\beta \in (0, 1)$ to satisfy the overall tolerance criteria, and a constant $c \in (a, 1)$ multiplying Δ_k to assure that τ_k goes to zero. The constant a will be defined as part of a sufficient decrease condition that forces the size of the trust-region to decrease to zero in the next section.

10.2.3 Trust-region Updating

Without using the high-fidelity function gradient, the trust-region update scheme must ensure the size of the trust-region decreases to zero to establish convergence. To do this, we use a requirement similar to the fraction of Cauchy decrease requirement in an the unconstrained trust-region formulation, see for example Ref. [40]. We require that the improvement in our merit function is at least a small constant a, $0 < a \leq \epsilon$, multiplying Δ_k,

$$\hat{\Phi}(\mathbf{x}_k, \sigma_k) - \hat{\Phi}(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) \geq a\Delta_k. \tag{10.14}$$

The sufficient decrease condition is enforced through the trust region update parameter, ρ_k. The update parameter is the ratio of the actual reduction in the merit function to the

predicted reduction in the merit function unless the sufficient decrease condition is not met,

$$\rho_k = \begin{cases} 0 & \hat{\Phi}(\mathbf{x}_k, \sigma_k) - \hat{\Phi}(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) < a\Delta_k \\ \frac{\Phi(\mathbf{x}_k,\sigma_k) - \Phi(\mathbf{x}_k+\mathbf{s}_k,\sigma_k)}{\hat{\Phi}(\mathbf{x}_k,\sigma_k) - \hat{\Phi}(\mathbf{x}_k+\mathbf{s}_k,\sigma_k)} & \text{otherwise.} \end{cases} \tag{10.15}$$

The size of the trust region, Δ_k, must now be updated based on the quality of the surrogate model prediction. The size of the trust region is increased if the surrogate model predicts the change in the function value well, kept constant if the prediction is fair, and the trust region is contracted if the model predicts the change poorly. Specifically, we update the trust region size using

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \eta_1 \leq \rho_k \leq \eta_2, \\ \gamma_0 \Delta_k & \text{if } \rho_k \leq \eta_0, \\ \Delta_k & \text{otherwise,} \end{cases} \tag{10.16}$$

where $0 < \eta_0 < \eta_1 < 1 < \eta_2$, $0 < \gamma_0 < 1$, and $\gamma_1 > 1$. Regardless of whether or not a sufficient decrease has been found, the trust-region location will be updated if the trial point has decreased the value of the merit function,

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k & \text{if } \Phi(\mathbf{x}_k, \sigma_k) > \Phi(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) \\ \mathbf{x}_k & \text{otherwise.} \end{cases} \tag{10.17}$$

A new surrogate model, $m_{k+1}(\mathbf{x})$, is then built such that it is fully linear on a region $\mathcal{B}_{k+1}$ having center $\mathbf{x}_{k+1}$ and size Δ_{k+1}. The new fully linear model is constructed using the procedure of Wild *et al.* [41] with the calibration technique of March and Willcox [29].

10.2.4 Termination

For termination, we must establish that the first-order KKT conditions,

$$\|\nabla f_{\text{high}}(\mathbf{x}_k) + A(\mathbf{x}_k)^T \lambda(\mathbf{x}_k)\| \leq \epsilon, \tag{10.18}$$

$$\| \left[\mathbf{h}(\mathbf{x}_k), \mathbf{g}^+(\mathbf{x}_k)\right] \| \leq \epsilon \tag{10.19}$$

are satisfied at $\mathbf{x}_k$, where $A(\mathbf{x}_k)$ is defined to be the Jacobian of all active constraints at that point,

$$A(\mathbf{x}_k) = \left[\nabla \mathbf{h}(\mathbf{x}_k), \nabla \mathbf{g}^+(\mathbf{x}_k)\right]^T. \tag{10.20}$$

The constraint violation criteria, Eq. 10.19, can be evaluated directly. However, the first-order condition, Eq. 10.18, cannot be verified directly in the derivative-free case because the gradient $\nabla f_{\text{high}}(\mathbf{x}_k)$ is unknown. Therefore, we must use the property of a fully linear model given in Eq. 10.10, which shows that as $\Delta_k \to 0$, then $\|\nabla f_{\text{high}}(\mathbf{x}) - \nabla m_k(\mathbf{x})\| \to 0$. We know that when the constraint violation termination condition is satisfied the trust-region problem in Eq. 10.12 can be solved. Therefore if the trust-region constraint is inactive, we know that

$$\|\nabla m(\mathbf{x}_k) + A(\mathbf{x}_k)^T \hat{\lambda}(\mathbf{x}_k)\| \leq \min\left[\beta\epsilon, c\Delta_k\right], \tag{10.21}$$

where $\hat{\lambda}$ are the appropriate Lagrange multipliers computed using the surrogate model as opposed to the high-fidelity function. Therefore a Δ_k sufficiently small, for example $\Delta_k \leq \epsilon_2$ for a small constant ϵ_2 is sufficient to show $\|\nabla f_{\text{high}}(\mathbf{x}) - \nabla m_k(\mathbf{x})\| \approx 0$, $\|\hat{\lambda} - \lambda\| \approx 0$, and $\|\nabla f_{\text{high}}(\mathbf{x}_k) + A(\mathbf{x}_k)^T \hat{\lambda}(\mathbf{x}_k)\| \leq \epsilon$.

10.2.5 Convergence Discussion

To demonstrate convergence of this algorithm we must first show that the trust-region subproblem that handles the constraints explicitly, Eq. 10.12, can be solved in a finite number of iterations, secondly that the size of the trust-region must go to zero, and finally that the algorithm cannot stop before the KKT conditions are satisfied for the original optimization problem, Eq. 10.1. To demonstrate that the trust-region subproblem handling the constraints directly can be solved we must show that a feasible point (or point feasible to within numerical tolerance) must exist within a trust region that will be encountered by solving the surrogate penalty trust-region subproblem, Eq. 10.13.

We begin by establishing a bound on the constraint violation in terms of the penalty parameter σ_k. We use the definition of $A^+(\mathbf{x})$ from Conn *et al.* [55],

$$A^+(\mathbf{x}) = A(\mathbf{x})^T(A(\mathbf{x})A(\mathbf{x})^T)^{-1}, \tag{10.22}$$

so that least square Lagrange multipliers λ which approximate,

$$\nabla f_{\text{high}}(\mathbf{x}) + A(\mathbf{x})^T\lambda = 0, \tag{10.23}$$

can be found by the matrix-vector product,

$$\lambda(\mathbf{x}) = -A^+(\mathbf{x})^T\nabla f_{\text{high}}(\mathbf{x}). \tag{10.24}$$

Given the assumption that minimizing a quadratic penalty function terminates at a point that is both feasible and regular, Conn *et al.* shows that if the objective function is used exactly, then there exists an iteration k after which all $A^+(\mathbf{x}_k)$ will exist, be bounded such that $\|A^+(\mathbf{x}_k)\| \leq \kappa_1$ for a positive constant κ_1, and converge to $A^+(\mathbf{x}^*)$ [55]. Conn *et al.* then establishes the bound for the constraint violation,

$$\| \left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T\right] \| \leq \frac{\kappa_1\omega_k + \|\lambda(\mathbf{x}^*)\| + \kappa_2\|\mathbf{x}_k - \mathbf{x}^*\|}{\sigma_k}, \tag{10.25}$$

where κ_2 is a positive constant and $\|\nabla\Phi(\mathbf{x}_k,\sigma_k)\| = \|\nabla f_{\text{high}}(\mathbf{x}_k) + \sigma_k A(\mathbf{x}_k)^T\left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T\right]^T\| \leq \omega_k$, for a convergence tolerance ω_k[55]. Using the surrogate model, for any solution to Eq. 10.13 where the trust-region is an inactive constraint, we have that, $\|\nabla\hat{\Phi}(\mathbf{x}_k,\sigma_k)\| = \|\nabla m_k(\mathbf{x}_k) + \sigma_k A(\mathbf{x}_k)^T\left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T\right]^T\| \leq \tau_k$. Accordingly, when the trust-region constraint is inactive, using the definitions of τ_k and a fully linear model we have that,

$$\| \left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T\right] \| \leq \frac{\kappa_1(c+\kappa_g)\Delta_k + \|\lambda(\mathbf{x}^*)\| + \kappa_2\|\mathbf{x}_k - \mathbf{x}^*\|}{\sigma_k}. \tag{10.26}$$

Now, using the fact that all $\mathbf{x}_k$ lie within the compact set $L(\mathbf{x}_0,\sigma_0)$, and that $\mathbf{x}^*$ is a regular point we can say that there exists a finite constant that bounds the numerator. Therefore by increasing the penalty parameter σ_k to infinity we may force the constraint violation to zero. Moreover, if we examine the constraint violation bound of Eq. 10.26 further, we see that the bound has the form $\kappa_3\Delta_k/\sigma_k + \kappa_4/\sigma_k$, with κ_3, κ_4 being arbitrary positive constants. It will be shown shortly that $\{\Delta_k\}$ converges to zero, therefore we must have that $\{\sigma_k\}$ increases to infinity strictly faster than $\{\Delta_k\}$ converges to zero, or that the series $\sigma_k\Delta_k$ diverges, in order to mitigate the term κ_4/σ_k. This criteria is enforced by construction and ensures that an iteration k exists such that a feasible point will reside in the interior of a trust region of size Δ_k and the subproblem using the constraints explicitly, Eq. 10.12, can be solved.

We now use our sufficient decrease condition, that $\rho_k = 0$ unless $\hat{\Phi}(\mathbf{x}_k,\sigma_k) - \hat{\Phi}(\mathbf{x}_k + \mathbf{s}_k,\sigma_k) \geq a\Delta_k$, to establish the proposition,

$$\lim_{k\to+\infty}\Delta_k = 0. \tag{10.27}$$

From the sufficient decrease condition, we know that the trust region size decreases unless the change in the merit function $\Phi(\mathbf{x}_k, \sigma_k) - \Phi(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) \geq \eta_0 a \Delta_k$. Therefore we must show that the total number of times in which the size of the trust region is kept constant or increased must be bounded. To demonstrate this, we have assumed a priori that the merit function is bounded from below and also that the trust-region algorithm remains within a level-set of the function from where it initiated. Let us now consider the merit function written in an alternate form,

$$\Phi(\mathbf{x}_k, \sigma_k) = f_{\text{high}}(\mathbf{x}_k) + \frac{\sigma_k}{2} \| \left[\mathbf{h}(\mathbf{x}_k), \mathbf{g}^+(\mathbf{x}_k) \right] \|^2. \tag{10.28}$$

From Eq. 10.26 we know that if σ_k is large enough such that the bound on the constraint violation is less than unity, then we have established that at each subsequent iteration,

$$\| \left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T \right] \|^2 \leq \left[\frac{\kappa_1 (c + \kappa_g) \Delta_k + \|\lambda^*\| + \kappa_2 \|\mathbf{x}_k - \mathbf{x}^*\|}{\sigma_k} \right]^2. \tag{10.29}$$

Now combining Eqs. 10.28 and 10.29, we can show that at an iteration k an upper bound in the total remaining change in the merit function is

$$f_{\text{high}}(\mathbf{x}_k) - \min_{\mathbf{x} \in L(\mathbf{x}_0, \sigma_0)} f_{\text{high}}(\mathbf{x}) + \frac{\left[\kappa_1 (c + \kappa_g) \Delta_k + \|\lambda(\mathbf{x}^*)\| + \kappa_2 \|\mathbf{x}_k - \mathbf{x}^*\| \right]^2}{\sigma_k}. \tag{10.30}$$

Given that that Δ_k is always bounded from above by $\Delta_{\max}$, $\|\lambda(\mathbf{x}^*)\|$ is bounded because $\mathbf{x}^*$ is a regular point, and $\|\mathbf{x}_k - \mathbf{x}^*\|$ is bounded because we've assumed $L(\mathbf{x}_0, \sigma_0)$ is a compact set. Therefore, if the series $\{1/\sigma_k\}$ has a finite sum, then the total improvement in the merit function is finite. Accordingly, the sum of the series $\{\Delta_k\}$ must be finite, and $\Delta_k \to 0$ as $k \to \infty$.

We have currently placed two restrictions on the series $\{\sigma_k\}$, that it grow faster than $1/\Delta_k$ and that $\{1/\sigma_k\}$ has a finite sum. A third restriction comes from the fact that we may not grow σ_k arbitrarily quickly for two reasons, first the solution $\mathbf{x}_{k+1}$ can move significantly far from $\mathbf{x}_k$ such as to cause problems in the trust region framework and secondly, the error between the surrogate model and high-fidelity function may cause arbitrarily poor estimates for $\mathbf{x}_{k+1}$. To place a conservative upper bound on the growth rate for the series $\{\sigma_k\}$ we compare the value of the merit function at a point $\Phi(\mathbf{x}_k + \mathbf{p}, \sigma_k)$ with its linearized prediction based on $\Phi(\mathbf{x}_k, \sigma_k)$. From the mean value theorem, we know there exists a $t \in (0, 1)$ such that,

$$\Phi(\mathbf{x}_k + \mathbf{p}, \sigma_k) = \Phi(\mathbf{x}_k, \sigma_k) + \nabla \Phi(\mathbf{x}_k + t\mathbf{p}, \sigma_k)^T \mathbf{p}. \tag{10.31}$$

We may also create the linearized prediction of $\Phi(\mathbf{x}_k + \mathbf{p}, \sigma_k)$, $\tilde{\Phi}(\mathbf{x}_k + \mathbf{p}, \sigma_k)$ as,

$$\tilde{\Phi}(\mathbf{x}_k + \mathbf{p}, \sigma_k) = \Phi(\mathbf{x}_k, \sigma_k) + \nabla \Phi(\mathbf{x}_k, \sigma_k)^T \mathbf{p}, \tag{10.32}$$

which provides us the bound,

$$\|\Phi(\mathbf{x}_k + \mathbf{p}, \sigma_k) - \tilde{\Phi}(\mathbf{x}_k + \mathbf{p}, \sigma_k)\| \leq \|\nabla \Phi(\mathbf{x}_k + t\mathbf{p}, \sigma_k) - \nabla \Phi(\mathbf{x}_k, \sigma_k)\| \|\mathbf{p}\|. \tag{10.33}$$

If κ_{fg} is the Lipschitz constant for $\nabla f_{\text{high}}(\mathbf{x})$, κ_c is the maximum Lipschitz constant for the constraints, and κ_{cd} is the maximum Lipschitz constant for a constraint gradient we can show that

$$\begin{aligned} &\|\Phi(\mathbf{x}_k + \mathbf{p}, \sigma_k) - \tilde{\Phi}(\mathbf{x}_k + \mathbf{p}, \sigma_k)\| \leq \\ &\qquad \left[\kappa_{fg} + \sigma_k \left(\kappa_c \| \left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T \right] \| + \kappa_{cd} \|A(\mathbf{x}_k)\| \|\mathbf{p}\| + \kappa_c \kappa_{cd} \|\mathbf{p}\|^2 \right) \right] \|\mathbf{p}\|^2. \end{aligned} \tag{10.34}$$

We have a similar result for $\hat{\Phi}(\mathbf{x}, \sigma_k)$ by replacing κ_{fg} with the sum $\kappa_{fg} + \kappa_g$ using the definition of a fully linear model, and by bounding $\|\mathbf{p}\|$ by Δ_k. Therefore, we may show the error in a linearized prediction of the surrogate model will go to zero provided that the series $\{\sigma_k \Delta_k^2\}$ converges to zero. $\{\Delta_k\}$ will converge R-linearly to zero with constant γ_0, therefore $\{\sigma_k\}$ must grow strictly slower than $\{1/\gamma_0^{2k}\}$. Accordingly, if the penalty parameter $\{\sigma_k\}$ grows at a rate exceeding both $\{k\}$ and $\{1/\Delta_k\}$ then $\{\Delta_k\}$ will converge to zero and an iterate $\mathbf{x}_k$ will exist for a finite k such that the trust-region problem handling the constraints explicitly will be feasible. In addition, if $\{\sigma_k\}$ grows slower than $\{1/\Delta_k^2\}$ then the error between the predicted values of the merit function and the actual values of the merit function will converge to zero within the trust region. It also suggests that a good choice for $\{\sigma_k\}$ has the form $\sigma_k = \max\left[e^{k/10}, 1/\Delta_k^{1.1}\right]$.

To demonstrate that the sequence of iterates generated by this algorithm terminates at a point $\mathbf{x}^*$ where the KKT conditions must be satisfied to within a tolerance of ϵ, we will assume for contradiction that the algorithm terminates at a regular and feasible point $\mathbf{x}^*$ at which the KKT conditions are not satisfied. We have established for these assumptions that the algorithm will always terminate by solving the trust-region subproblem in Eq. 10.12. If the KKT conditions are not satisfied, then as the trust-region size decreases to zero the KKT conditions will not be satisfied for the surrogate model. Therefore, the possible solutions to Eq. 10.12 are the current iterate or a point on the trust-region boundary. If the solution to the trust-region subproblem is the current iterate, then $\|\nabla m_k(\mathbf{x}_k)\| \leq \beta\epsilon$ because the error between the surrogate model and high-fidelity function will decrease to zero. This shows that $\|\nabla f_{\text{high}}(\mathbf{x})\| \leq \epsilon$ is a necessary contradiction. If the solution to Eq. 10.12 is always on the trust-region boundary and the trust-region continues to shrink, then the decrease in the surrogate model value must be less than $a\Delta_k$. For any sufficiently small trust-region size, the fully linear model ensures that the error between the surrogate model and high-fidelity function decreases to zero. Accordingly, for sufficiently small trust-region size, this shows that $\|\nabla f_{\text{high}}(\mathbf{x})\| \leq a$, and since $a < \epsilon$ then we have established a contradiction and the KKT conditions are satisfied to a tolerance of ϵ.

10.2.6 Implementation

The numerical implementation of the multifidelity optimization algorithm, which does not compute the gradient of the high-fidelity objective function, is presented as Algorithm 6. A set of possible parameters that may be used in this algorithm are listed in Table 10.1 in Section 10.4. A key element of this algorithm is the logic to switch from the penalty function trust-region subproblem, Eq. 10.13, to the subproblem that uses the constraints explicitly, Eq. 10.12. Handling the constraints exactly will generally lead to faster convergence and fewer function evaluations; however, a feasible solution to this subproblem likely does not exist at early iterations. If either the constraint violation is sufficiently small, $\| \left[\mathbf{h}(\mathbf{x}_k)^T \ \mathbf{g}^+(\mathbf{x}_k)^T\right] \| \leq \epsilon$, or the linearized steps, δ_h, satisfying $h(\mathbf{x}) + \nabla h(\mathbf{x})^T \delta_h = 0$ for all equality and inequality constraints are all smaller than the size of the trust region, then the subproblem with the explicit constraints is attempted. If the optimization fails, then the penalty function subproblem is solved.

This method may be accelerated with the use of multiple lower-fidelity models. March and Willcox [29] suggest a multifidelity filtering technique to combine estimates from multiple low-fidelity functions into a single maximum likelihood estimate of the high-fidelity function value. That technique will work unmodified within this multifidelity optimization framework and will likely improve performance.

Algorithm 6: Multifidelity Objective Trust-Region Algorithm

1: Update tolerance, $\tau_k = \min\left[\beta\epsilon, c\Delta_k\right]$.

2: Choose and solve a trust-region subproblem:

2a: If the maximum linearized step to constraint feasibility for all active constraints is smaller than the current trust region size, Δ_k then solve:

$$\begin{aligned} \min_{\mathbf{s}_k \in \mathbb{R}^n} \quad & m_k(\mathbf{x}_k + \mathbf{s}_k) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_k + \mathbf{s}_k) = 0 \\ & \mathbf{g}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ & \|\mathbf{s}_k\| \leq \Delta_k, \end{aligned}$$

to convergence tolerance τ_k.

2b: If 2a is not used or fails to converge to the required tolerance, solve the trust-region subproblem:

$$\begin{aligned} \min_{\mathbf{s}_k \in \mathbb{R}^n} \quad & \hat{\Phi}_k(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) \\ \text{s.t.} \quad & \|\mathbf{s}_k\| \leq \Delta_k. \end{aligned}$$

to convergence tolerance τ_k.

3: If $f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ has not been evaluated previously, evaluate the high-fidelity function at that point.

3a: Store $f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ in database.

4: Compute the merit function $\Phi(\mathbf{x}_k, \sigma_k)$, $\Phi(\mathbf{x}_k + \mathbf{s}_k, \sigma_k)$ and the surrogate merit function, $\hat{\Phi}(\mathbf{x}_k + \mathbf{s}_k, \sigma_k)$.

5: Compute the ratio of actual improvement to predicted improvement,

$$\rho_k = \begin{cases} 0 & \hat{\Phi}(\mathbf{x}_k, \sigma_k) - \hat{\Phi}(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) < a\Delta_k \\ \frac{\Phi(\mathbf{x}_k, \sigma_k) - \Phi(\mathbf{x}_k + \mathbf{s}_k, \sigma_k)}{\hat{\Phi}(\mathbf{x}_k, \sigma_k) - \hat{\Phi}(\mathbf{x}_k + \mathbf{s}_k, \sigma_k)} & \text{otherwise.} \end{cases}$$

6: Update the trust region size according to ρ_k,

$$\Delta_{k+1} \quad = \quad \begin{cases} \min\{\gamma_1 \Delta_k, \Delta_{\max}\} & \text{if } \eta_1 \leq \rho_k \leq \eta_2 \\ \gamma_0 \Delta_k & \text{if } \rho_k \leq \eta_0, \\ \Delta_k & \text{otherwise,} \end{cases}$$

7: Accept or reject the trial point according to improvement in the merit function,

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k & \Phi(\mathbf{x}_k, \sigma_k) - \Phi(\mathbf{x}_k + \mathbf{s}_k, \sigma_k) > 0 \\ \mathbf{x}_k & \text{otherwise.} \end{cases}$$

8: Create new model $m_{k+1}(\mathbf{x})$ fully linear on $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_{k+1}\| \leq \Delta_{k+1}\}$.

9: Increment the penalty, $\sigma_{k+1} = \max\left[e^{k+1/10}, 1/\Delta_{k+1}^{1.1}\right]$.

10: Check for convergence, if the trust region constraint is inactive, $\|\nabla m_{k+1}(\mathbf{x}_{k+1}) + A(\mathbf{x}_{k+1})^T \hat{\lambda}\| \leq \beta\epsilon$, $\|\left[\mathbf{h}(\mathbf{x}_k), \mathbf{g}^+(\mathbf{x}_k)\right]\| \leq \epsilon$, and $\Delta_k \leq \epsilon_2$ the algorithm is converged, otherwise go to step 1.

10.3 Multifidelity Objective and Constraint Optimization

We now consider a more general constrained optimization problem with a computationally expensive objective function and computationally expensive constraints. We assume for both the expensive objective and expensive constraints that the gradients are either unavailable, unreliable or expensive to estimate. Accordingly, we augment the multifidelity optimization problem in Eq. 10.1 with the high-fidelity constraint, $c_{\text{high}}(\mathbf{x}) \leq 0$. In addition, we have a low-fidelity estimate of this constraint, $c_{\text{low}}(\mathbf{x})$, which estimates the same metric as the high-fidelity constraint, but with unknown error. Therefore, our goal is to find the vector $\mathbf{x} \in \mathbb{R}^n$ of n design variables that solves the nonlinear constrained optimization problem,

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f_{\text{high}}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0 \\ & c_{\text{high}}(\mathbf{x}) \leq 0, \end{aligned} \tag{10.35}$$

where $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ represent vectors of inexpensive equality and inequality constraints with derivatives that are either known or may be estimated cheaply. The same assumptions for the expensive objective function formulation are made for the functions presented in this formulation, including that a quadratic penalty function with the new high-fidelity constraint is bounded from below within an initial expanded level-set. A point of note is that multiple high-fidelity constraints can be used if an initial point $\mathbf{x}_0$ is given that is feasible with respect to all constraints; however, due to the effort required to construct approximations of the multiple high-fidelity constraints, it is recommended that all of the high-fidelity constraints be combined into a single high-fidelity constraint through, as an example, a discriminant function [56, 57]. This optimization problem will be solved using the multifidelity optimization method presented in Section 10.2 to find a feasible point, and then an interior point formulation is presented in Section 10.3.2 to solve the optimization problem in Eq. 10.35. Convergence of the algorithm is discussed in Section 10.3.3, and the numerical implementation is presented in Section 10.3.4.

10.3.1 Finding a Feasible Point

This algorithm begins by finding a point that is feasible with respect to all of the constraints and then minimizes the high-fidelity objective function subject to all of the constraints in a manner similar to an interior-point method. In order to find an initial feasible point, the algorithm presented in Section 10.2 iterates on the optimization problem,

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & c_{\text{high}}(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0, \end{aligned} \tag{10.36}$$

until a point that is feasible with respect to Eq. 10.35 is found. If this optimization problem is unconstrained then the trust-region algorithm of Conn *et al.* [40] is used with the multifidelity calibration method of March and Willcox [29]. The optimization problem in Eq. 10.36 may violate one of the assumptions of the multifidelity objective function method in that $c_{\text{high}}(\mathbf{x})$ may not be bounded from below. This issue will be addressed in the numerical implementation of the method in Section 10.3.4.

10.3.2 Interior Point Trust-region Method

To minimize the high-fidelity objective function subject to the constraints, this formulation again uses the general Bayesian calibration approach in Ref. [35] to create fully linear surrogate models for both the high-fidelity objective function and the high-fidelity constraint. The surrogate model for the objective function is $m_k(\mathbf{x})$ as defined in Eq. 10.2. For the constraint, the surrogate model, $\bar{m}_k(\mathbf{x})$, is defined as

$$\bar{m}_k(\mathbf{x}) = c_{\text{low}}(\mathbf{x}) + \bar{e}_k(\mathbf{x}). \tag{10.37}$$

From the fact that $\bar{m}(\mathbf{x})$ is a fully linear model for $c_{\text{high}}(\mathbf{x})$, we know

$$\|\nabla c_{\text{high}}(\mathbf{x}) - \nabla \bar{m}_k(\mathbf{x})\| \leq \kappa_{cg}\Delta_k \quad \forall \mathbf{x} \in \mathcal{B}_k \tag{10.38}$$

$$|c_{\text{high}}(\mathbf{x}) - \bar{m}_k(\mathbf{x})| \leq \kappa_{cf}\Delta_k^2 \quad \forall \mathbf{x} \in \mathcal{B}_k. \tag{10.39}$$

In addition, we require that our procedure to construct fully linear models ensures that at the current design iterate, the fully linear models have the exact value of the function they are modeling. Accordingly, we have that

$$f_{\text{high}}(\mathbf{x}_k) = m_k(\mathbf{x}_k) \tag{10.40}$$

$$c_{\text{high}}(\mathbf{x}_k) = \bar{m}_k(\mathbf{x}_k). \tag{10.41}$$

This is required so that every trust-region subproblem is feasible at the initial point $\mathbf{x}_k$.

The trust-region subproblem used is to minimize the surrogate high-fidelity objective function subject to the easy constraints, the surrogate high-fidelity constraint, and the trust-region constraint,

$$\begin{aligned} \min_{\mathbf{s}_k \in \mathbb{R}^n} \quad & m_k(\mathbf{x}_k + \mathbf{s}_k) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_k + \mathbf{s}_k) = 0 \\ & \mathbf{g}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ & \bar{m}_k(\mathbf{x}_k + \mathbf{s}_k) \leq \max\{c_{\text{high}}(\mathbf{x}_k), -e\Delta_k\} \\ & \|\mathbf{s}_k\| \leq \Delta_k. \end{aligned} \tag{10.42}$$

The trust-region subproblem is solved to the same termination tolerance as the multifidelity objective function formulation, $\|\nabla_x \mathfrak{L}_k\| \leq \tau_k$, where $\mathfrak{L}_k$ is the Lagrangian. The surrogate model constraint does not have zero as a right hand side to account for the fact the algorithm is looking for interior points. The right hand side, $\max\{c_{\text{high}}(\mathbf{x}_k), -e\Delta_k\}$, where e is a positive constant greater than c used in the definition of τ_k, assures that the constraint is initially feasible and that protection of constraint violation decreases to zero as the number of iterations increase to infinity.

The center of the trust region is updated if a decrease in the objective function is found at a feasible point,

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k & \text{if } f_{\text{high}}(\mathbf{x}_k) > f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) \text{ and } c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ \mathbf{x}_k & \text{otherwise.} \end{cases} \tag{10.43}$$

The trust-region size update must ensure that the predictions of the surrogate models are accurate and that the size of the trust region goes to zero in the limit as the number of iterations goes to infinity. Therefore, we again impose a sufficient decrease condition, that the change in the objective function is at least a constant, a, multiplying Δ_k,

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1\Delta_k, \Delta_{\max}\} & \text{if } f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) - f_{\text{high}}(\mathbf{x}_k) \geq a\Delta_k \text{ and } c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ \gamma_0\Delta_k & \text{otherwise.} \end{cases} \tag{10.44}$$

New surrogate models, $m_{k+1}(\mathbf{x})$ and $\bar{m}_{k+1}(\mathbf{x})$, are then built such that they are fully linear on a region $\mathcal{B}_{k+1}$ having center $\mathbf{x}_{k+1}$ and size Δ_{k+1}. The new fully linear models are constructed using the procedure of Wild *et al.* [41] with the calibration technique of March and Willcox [29].

Convergence of the algorithm is verified by assuring the KKT conditions are satisfied on the surrogate models and that the trust-region radius is sufficiently small to establish that the error between the surrogate model gradients and the high-fidelity function gradients is sufficiently small.

10.3.3 Multifidelity Objective and Constraints Convergence Discussion

To demonstrate convergence of this algorithm we must first show that the trust region size goes to zero and second that the algorithm cannot stop before the KKT conditions are satisfied for the high-fidelity problem. To establish that the trust region size goes to zero, we use the assumption that a quadratic penalty function including all of constraints is bounded from below. This means that the high fidelity objective function must be bounded from below within the feasible region of Eq. 10.36. We therefore know that from the first feasible iterate found, the difference between the high-fidelity function at that point and the value of the high-fidelity function at an optimum of this problem is finite. Since we have required that the decrease in the objective function is at least a constant a multiplying Δ_k this means the sum of the series $\{\Delta_k\}$ is bounded, and accordingly

$$\lim_{k \to +\infty} \Delta_k = 0. \tag{10.45}$$

We now establish that this algorithm must converge to a limit point, $\mathbf{x}^*$, where the KKT conditions,

$$\|\nabla f_{\text{high}}(\mathbf{x}_k) + A(\mathbf{x}_k)^T \lambda(\mathbf{x}_k) + \nabla c_{\text{high}}(\mathbf{x}_k)\lambda_c\| \leq \epsilon, \tag{10.46}$$

$$\| \left[\mathbf{h}(\mathbf{x}_k), \mathbf{g}^+(\mathbf{x}_k)\right] \| \leq \epsilon, \tag{10.47}$$

$$c_{\text{high}}(\mathbf{x}_k) \leq 0, \tag{10.48}$$

with λ_c the Lagrange multiplier for the high-fidelity constraint, are satisfied. The constraint violation criteria, Eq. 10.47 and Eq. 10.48, are established by construction and the use of Algorithm 6. To establish the first-order condition, Eq. 10.46, we must use the property of a fully linear model given in Eqs. 10.10 and 10.38, which shows that as $\Delta_k \to 0$, then $\|\nabla f_{\text{high}}(\mathbf{x}) - \nabla m_k(\mathbf{x})\| \to 0$ and a similar conclusion for the high-fidelity constraint. When the trust-region subproblem given in Eq. 10.42 has been solved and the trust-region constraint is inactive, we know that

$$\|\nabla m(\mathbf{x}_k) + A(\mathbf{x}_k)^T \hat{\lambda}(\mathbf{x}_k) + \nabla \bar{m}(\mathbf{x}_k)\lambda_{\bar{m}}\| \leq \min\left[\beta\epsilon, c\Delta_k\right], \tag{10.49}$$

where $\hat{\lambda}$ are the appropriate Lagrange multipliers computed using the surrogate model as opposed to the high-fidelity function (see Eq. 10.24) and $\lambda_{\bar{m}}$ is a similar Lagrange multiplier for the surrogate model of the high-fidelity constraint. Thus we have established that as $\Delta_k \to 0$, then $\|\nabla f_{\text{high}}(\mathbf{x}) - \nabla m_k(\mathbf{x})\| \to 0$, $\|\nabla c_{\text{high}}(\mathbf{x}) - \nabla \bar{m}_k(\mathbf{x})\| \to 0$, $\|\hat{\lambda} - \lambda\| \to 0$, $\|\lambda_c - \lambda_{\bar{m}}\| \to 0$, and $\|\nabla f_{\text{high}}(\mathbf{x}_k) + A(\mathbf{x}_k)^T \hat{\lambda}(\mathbf{x}_k) + \nabla c_{\text{high}}(\mathbf{x}_k)\lambda_{\bar{m}}\| \leq \epsilon$. In addition, if no constraints are active we have only established convergence to a point such that, $\|\nabla f_{\text{high}}(\mathbf{x})\| \leq a$, but this also satisfies the first-order KKT conditions. The other way in which this algorithm might fail, is that the sequence of iterates could always have $c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) > 0$, in which case the trust region will shrink to zero without making any progress. However, a necessary contradiction is that the error in the value of a fully linear model

decreases with the square of the trust region size. The fact that a margin is kept with respect to the high-fidelity constraint violation and that the error in the value of the constraint value prediction decays at a faster rate than the error in the first-order KKT condition shows the prediction of the constraint surrogate model value will not force the size of the trust-region to zero before the KKT conditions are satisfied.

10.3.4 Multifidelity Objective and Constraint Implementation

The numerical implementation of this multifidelity optimization algorithm is presented as Algorithm 7. A set of possible parameters that may be used in this algorithm are listed in Table 10.1 in Section 10.4. An important implementation issue with this algorithm is finding the initial feasible point. Algorithm 6 is used to minimize the high-fidelity constraint value subject to the constraints with available derivatives in order to find a point that is feasible. However, Algorithm 6 uses a quadratic penalty function to handle the constraints with available derivatives if the constraints are violated. This requires that the objective function is bounded from below to assure the penalty method is effective. Accordingly, a more general problem than Eq. 10.36 to find an initial feasible point is to use,

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \max\{c_{\text{high}}(\mathbf{x}) + d, 0\}^2 \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0. \end{aligned} \tag{10.50}$$

The maximization in the objective prevents the need for the high-fidelity constraint to be bounded from below. The constant d is used to account for the fact that the surrogate model will have some error in its prediction of $c_{\text{high}}(\mathbf{x})$, so looking for a slightly negative value of the constraint may save iterations as compared to seeking a value that is exactly zero. For example, if d is larger than the κ_{cf} in Eq. 10.39 and the value of the constraint surrogate model is more negative than the size of the trust region squared, then we are assured the actual value of the high-fidelity constraint is negative.

A similar issue is in the solution of Eq. 10.42, where a slightly negative value of the surrogate constraint is desired. If this subproblem is solved with an interior point algorithm this should be satisfied automatically; however, if a sequential quadratic programming method is used the constraint violation will have a numerical tolerance that is either slightly negative or slightly positive. It may be necessary to bias the subproblem to look for a value of the constraint that is more negative than the optimizer constraint violation tolerance to assure the solution is an interior point.

A final implementation note is that if a high-fidelity constraint has numerical noise or steep gradients it may be wise to shrink the trust region at a slower rate, increasing γ_0. This will help to ensure that the trust-region does not decrease to zero at a suboptimal point.

10.4 Supersonic Airfoil Design Test Problem

This section presents results of the two multifidelity optimization algorithms on the supersonic airfoil design problem presented in Section 2.1. The airfoil design problem has 11 parameters, the angle of attack, 5 spline points on the upper surface and 5 spline points on the lower surface. The airfoils are constrained such that the minimum thickness-to-chord ratio is 0.05 and that the thickness everywhere on the airfoil must be positive. In addition, there are lower and upper bounds for all spline points. Note, that Cart3D has a finite convergence tolerance so there is some numerical noise in the lift and drag predictions. In addition, because random airfoils are used as initial conditions, Cart3D may fail to converge, in which case the results of the panel method are used.

Algorithm 7: Multifidelity Objective and Constraint Trust-Region Algorithm

0: Find a feasible design vector using Algorithm 6 to iterate on:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \max\{c_{\text{high}}(\mathbf{x}) + d, 0\}^2 \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}) = 0 \\ & \mathbf{g}(\mathbf{x}) \leq 0. \end{aligned}$$

1: Update tolerance, $\tau_k = \min[\beta\epsilon, c\Delta_k]$.

2: Solve the trust-region subproblem:

$$\begin{aligned} \min_{\mathbf{s}_k\in\mathbb{R}^n} \quad & m_k(\mathbf{x}_k + \mathbf{s}_k) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}_k + \mathbf{s}_k) = 0 \\ & \mathbf{g}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ & \bar{m}_k(\mathbf{x}_k + \mathbf{s}_k) \leq \min\{c_{\text{high}}(\mathbf{x}_k), -e\Delta_k\} \\ & \|\mathbf{s}_k\| \leq \Delta_k. \end{aligned}$$

3: If $f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ or $c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ have not been evaluated previously, evaluate the high-fidelity functions at that point.

3a: Store $f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ and $c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k)$ in a database.

4: Accept or reject the trial point according to:

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_k + \mathbf{s}_k & \text{if } f_{\text{high}}(\mathbf{x}_k) > f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) \text{ and } c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ \mathbf{x}_k & \text{otherwise.} \end{cases}$$

5: Update the trust region size according to:

$$\Delta_{k+1} = \begin{cases} \min\{\gamma_1\Delta_k, \Delta_{\max}\} & \text{if } f_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) - f_{\text{high}}(\mathbf{x}_k) \geq a\Delta_k \text{ and } c_{\text{high}}(\mathbf{x}_k + \mathbf{s}_k) \leq 0 \\ \gamma_0\Delta_k & \text{otherwise.} \end{cases}$$

6: Create new models $m_{k+1}(\mathbf{x})$ and $\bar{m}_{k+1}(\mathbf{x})$ fully linear on $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_{k+1}\| \leq \Delta_{k+1}\}$.

7: Check for convergence, if the trust region constraint is inactive, $\|\nabla m(\mathbf{x}_k) + A(\mathbf{x}_k)^T\hat{\lambda}(\mathbf{x}_k) + \nabla\bar{m}(\mathbf{x}_k)\lambda_{\bar{m}}\| \leq \beta\epsilon$, and $\Delta_k \leq \epsilon_2$, the algorithm is converged, otherwise go to step 1.

Constant	Description	Value
a	Sufficient decrease constant	1×10^{-4}
β	Overall Convergence tolerance multiplier	1×10^{-2}
c	Convergence tolerance based on trust region size	1×10^{-2}
d	Artificial lower bound for constraint value	1
e	Additional conservatism for constraint violation	0.1
ϵ, ϵ_2	Termination Tolerance	5×10^{-4}
γ_0	Trust region contraction ratio	0.5
γ_1	Trust region expansion ratio	2
η_0	Trust region contraction criterion	0.25
η_1, η_2	Trust region expansion criterion	$0.75, 2.0$
Δ_0	Initial trust region radius	1
Δ_{max}	Maximum trust region size	20
σ_k	Penalty parameter	$\max\left[e^{k/10}, 1/\Delta_k^{1.1}\right]$
δ_x	Finite difference step	1×10^{-5}

Table 10.1. List of constants used in the algorithm. All parameters used in constructing the radial basis function error model are the same is in Chapter 9.

The following sections present results for three optimization examples each using this airfoil problem to demonstrate the capabilities of the optimization algorithms presented. In the first example, Section 10.4.1, the airfoil drag will be minimized using the constrained multifidelity objective function formulation presented in Section 10.2 with only the simple geometric constraints. In the second example, Section 10.4.2, the airfoil lift to drag ratio will be maximized subject to a constraint that the drag coefficient is less than 0.01, where the constraint will be handled with the multifidelity framework presented in Section 10.3. In the final example, Section 10.4.3, the airfoil lift to drag ratio will be maximized subject to the constrained drag coefficient and both the objective function and the constraints are handled with the multifidelity framework presented in Section 10.3. The initial airfoils for all problems will be randomly generated and likely will not satisfy the constraints.

The three airfoil problems will be solved with four optimization algorithms: Sequential Quadratic Programming[58] (SQP), a first-order consistent multifidelity trust-region algorithm that uses a sequential quadratic programming formulation and an additive correction[32], the high-fidelity-gradient-free approach presented in this chapter using a Gaussian radial basis function and a fixed spatial correlation parameter, $\exp -r^2/\xi^2$ with $\xi = 2$, and the approach presented in this chapter using a maximum likelihood estimate to find an improved correlation length, $\xi = \xi^*$, for the Gaussian radial basis function[29]. The Gaussian correlation functions used in these results are all isotropic, meaning they are the same in all directions. An anisotropic correlation function will likely speed convergence of this algorithm and reduce sensitivity to the problem scaling. The parameters used for the optimization algorithm are presented in Table 10.1.

10.4.1 Multifidelity Objective Function Results

This section presents optimization results in terms of the number of function evaluations required to find the minimum drag for a supersonic airfoil at Mach 1.5 with only geometric constraints on the design. Two cases are tested, the first uses the shock-expansion method as the high-fidelity function and the panel method as the low-fidelity function, the second uses Cart3D as the high-fidelity function and the panel method as the low-fidelity function. These problems are solved using the multifidelity optimization algorithm for a computationally expensive objective function and constraints with available derivatives presented in

High-Fidelity	Low-Fidelity	SQP	First-Order TR	RBF, $\xi = 2$	RBF, $\xi = \xi^*$
Shock-Expansion	Panel Method	314 (-)	110 (-65%)	73 (-77%)	68 (-78%)
Cart3D	Panel Method	359*(-)	109 (-70%)	80 (-78%)	79 (-78%)

Table 10.2. The average number of high-fidelity function evaluations to minimize the drag of a supersonic airfoil with only geometric constraints. The asterisk for the Cart3D results means a significant fraction of the optimizations failed and the average is taken over fewer samples. The numbers in parentheses indicate the percentage reduction in high-fidelity function evaluations relative to SQP.

High-Fidelity	Low-Fidelity	SQP	First-Order TR	RBF, $\xi = 2$	RBF, $\xi = \xi^*$
Shock-Expansion	Panel Method	827 (-)	104 (-87%)	104 (-87%)	115 (-86%)
Cart3D	Panel Method	909*(-)	100 (-89%)	103 (-89%)	105 (-88%)

Table 10.3. The average number of high-fidelity constraint evaluations required to maximize the lift to drag ratio of a supersonic airfoil estimated with a panel method subject to a multifidelity constraint. The asterisk for the Cart3D results means a significant fraction of the optimizations failed and the average is taken over fewer samples. The numbers in parentheses indicate the percentage reduction in high-fidelity function evaluations relative to SQP.

Section 10.2. The average number of high-fidelity function evaluations required to find a locally optimal design are presented in Table 10.2, and show that this approach uses approximately 78% fewer high-fidelity function evaluations than SQP and approximately 30% fewer function evaluations than the first order consistent trust-region method using finite difference gradient estimates.

10.4.2 Multifidelity Constraint Results

This section presents optimization results in terms of the number of function evaluations required to find the maximum lift to drag ratio for a supersonic airfoil at Mach 1.5 subject to both geometric constraints and the requirement that the drag coefficient is less than 0.01. The lift to drag ratio is computed with the panel method; however, the drag coefficient constraint is handled using the multifidelity technique presented in Section 10.3. Two cases are examined, in the first the shock-expansion method is the high-fidelity constraint and the panel method is the low-fidelity constraint, in the second case Cart3D is the high-fidelity constraint and the panel method is the low-fidelity constraint. Table 10.3 presents the average number of high-fidelity constraint evaluations required to find the optimal design using SQP, a first-order consistent trust-region algorithm and the techniques developed in this chapter. A significant decrease in the number of high-fidelity function evaluations is observed when compared with SQP; however, performance is almost the same as the first-order consistent trust-region algorithm.

10.4.3 Multifidelity Objective Function and Constraint Results

This section presents optimization results in terms of the number of function evaluations required to find the maximum lift to drag ratio for a supersonic airfoil at Mach 1.5 subject to geometric constraints and the requirement that the drag coefficient is less than 0.01. In this case, both the lift to drag ratio and the drag coefficient constraint are handled using the multifidelity technique presented in Section 10.3. In the first case, the shock-expansion method is the high-fidelity analysis used to estimate both metrics of interest and the panel method is the low-fidelity analysis, in the second case Cart3D is the high-

	High-Fidelity	Low-Fidelity	SQP	First-Order TR	RBF, $\xi = 2$	RBF, $\xi = \xi^*$
Objective:	Shock-Exp.	Panel Method	773 (-)	132 (-83%)	93 (-88%)	90 (-88%)
Constraint:	Shock-Exp.	Panel Method	773 (-)	132 (-83%)	97 (-87%)	96 (-88%)
	High-Fidelity	Low-Fidelity	SQP	First-Order TR	RBF, $\xi = 2$	RBF, $\xi = \xi^*$
Objective:	Cart3D	Panel Method	1168*(-)	97 (-92%)	104 (-91%)	112 (-90%)
Constraint:	Cart3D	Panel Method	2335*(-)	97 (-96%)	115 (-95%)	128 (-94%)

Table 10.4. The average number of high-fidelity objective function and high-fidelity constraint evaluations to optimize a supersonic airfoil for a maximum lift to drag ratio subject to a maximum drag constraint. The asterisk for the Cart3D results means a significant fraction of the optimizations failed and the average is taken over fewer samples. The numbers in parentheses indicate the percentage reduction in high-fidelity function evaluations relative to SQP.

fidelity analysis and the panel method is the low-fidelity analysis. Table 10.4 presents the number of function evaluations required to find the optimal design using SQP, a first-order consistent trust-region algorithm and the techniques developed in this chapter. Again a significant reduction in the number of high-fidelity function evaluations, both in terms of the constraint and the objective, are observed compared with SQP, and a similar number of high-fidelity function evaluations are observed when compared with the first-order consistent trust region approach using finite differences.

10.5 Summary

This chapter has presented two algorithms for multifidelity constrained optimization of computationally expensive functions when their derivatives are not available. The first method minimizes a high-fidelity objective function without using its derivative while satisfying constraints with available derivatives. The second method minimizes a high-fidelity objective without using its derivative while satisfying both constraints with available derivatives and an additional high-fidelity constraint without an available derivative. Both of these methods support multiple lower-fidelity models through the use of the multifidelity filtering technique presented in Chapter 9 without any modifications to the methods. The performance of the methods show a significant reduction in the number of high-fidelity function evaluations required to optimize a computationally expensive function when an appropriate low-fidelity function is used compared to a single-fidelity sequential quadratic programming formulation. In addition, this chapter demonstrated that these methods performed similarly to a first-order consistent trust-region algorithm with gradients estimated using finite-differences. This shows that these methods provide significant opportunity for optimization of computationally expensive functions without available gradients.

The behavior of the algorithms presented are slightly atypical of nonlinear programming methods. Although the algorithms guarantee convergence to a local minimum of the high-fidelity problem, the local minimum they converge to may not be the one in the immediate vicinity of the initial iterate. For example, the initial model may completely ignore the fact that the initial iterate is on a local minimum and move the iterate to a different point with a lower function value in another region of the design space. In addition, provided the trust-region subproblems can be solved, this algorithm should still be able to converge to high-fidelity optima at which the KKT conditions are not satisfied. This situation occurs when constraint qualification conditions are not satisfied and can cause robustness issues for nonlinear programming methods. In the algorithms presented, the design vector will approach a local minimum and the sufficient decrease test for the change in the objective function value will fail. This causes the size of the trust region to decay to zero around the

local minimum even though the KKT conditions are not satisfied.

In the case of hard constraints, or when the objective function fails to exist if the constraints are violated, it is still be possible to use the algorithm that approximates both a high-fidelity objective and constraint. After the initial feasible point is found, no design iterate will be accepted if the high-fidelity constraint is violated. Therefore the overall flow of the algorithm is unchanged. What must be changed is the technique to build fully linear models. In order to build a fully linear model, the objective function must be evaluated at a set of $n+1$ points that span $\mathbb{R}^n$. This requirement prohibits equality constraints and means that strict linear independent constraint qualification must be satisfied everywhere in the design space (preventing two inequality constraints from mimicking an equality constraint). If these two additional conditions hold then it will be possible to construct fully linear models everywhere in the feasible design space and use this algorithm to optimize computationally expensive functions with hard constraints. Therefore this chapter has presented a provably convergent multifidelity optimization algorithm that does not require estimating high-fidelity gradients, enables the use of multiple low-fidelity models, enables optimization of functions with hard constraints, exhibits robustness to optima at which the KKT conditions are not satisfied, and performs similarly in terms of the number of function evaluations to finite difference based multifidelity optimization methods.

Appendix A ModelCenter Multifidelity Optimization Installation Instructions

The Optimizer package consists of three major packages: the Matlab optimization code (methods presented in Chapters 9 and 10, the Matlab/Java API function (located in the Matlab Optimization directory as "ModelCenterWrapper.m"), and the example ModelCenter project (Rosenbrock.pxc). The example problem can be run from the "Example_Optimization.m" Matlab file (Contained in the Matlab Opimization Directory). Before anything will run, you must make sure:

1. The ModelCenter model must be created and saved. The ModelCenter model can be created using any component supported by ModelCenter (the Rosenbrock example creates the multi-fidelity function in VBScript).

2. The file "ModelCenterWrapper.m"contains machine specific paths that may need to be modified.

   - The line:

     ```
     javaaddpath('c:\Program Files\Phoenix Integration\ModelCenter 9.0');
     ```

     will need to be modified such that it points to the correct ModelCenter installation directory.

   - The line:

     ```
     h.invoke('loadModel','c:\Users\Cory\Desktop\ACDL UROP\Rosenbrock.pxc');
     ```

     will need to be modified such that it points to the correct ModelCenter model file.

   - The name of the ModelCenter component will also need to be set in three lines,

     ```
     h.invoke('setValue',['Model.Rosenbrock.x',numString], x(i));
     h.invoke('setValue','Model.Rosenbrock.fidelity',fidelity);
     f = h.invoke('getValue','Model.Rosenbrock.fnEval');
     ```

     in this example, the component name that will need to be altered to run a different model is "Rosenbrock".

3. The ModelCenter variables must have specific names (unless the "ModelCenterWrapper.m" is modified accordingly). The design variables should be named "x0","x1",...,"x(*n-1*)", The fidelity level must be named "fidelity". The function value to be passed back to Matlab must be named "fnEval". The example model should provide a nice template.

Once these updates have been made, the optimizer should now correctly run from the "Example_Optimization.m" Matlab file. This file takes the "ModelCenterWrapper.m" function as a function handle (this is similar to the method technique of the Matlab optimization toolbox routines); the "ModelCenterWrapper.m" function takes in as arguments, (1) an array of the design variables, (2) a fidelity level (in the example, 1 is used as low-fidelity

and 2 as high-fidelity), and (3) an array of parameters (this can be null). This function then passes these arguments to the ModelCenter model, runs the ModelCenter model, and returns the ModelCenter function evaluation.

- A readme file is in the Matlab Opimization Directory that describes each of the functions used in the unconstrained optimization.
- The two main files are (two-fidelity) "Example_Optimization.m", and (three-fidelity) "Example_Multi.m".

Bibliography

1. Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

2. Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

3. C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

4. S. Lophaven, H. Nielsen, and J. Sondergaard. Aspects of the Matlab toolbox DACE. Technical Report IMM-REP-2002-13, Technical University of Denmark, August 2002.

5. M. J. Aftosmis. Solution adaptive cartesian grid methods for aerodynamic flows with complex geometries. In *28th Computational Fluid Dynamics Lecture Series, von Karman Institute for Fluid Dynamics*, Rhode-Saint-Genése, Belgium, March 3-7 1997. Lecture Series 1997-02.

6. M. Nemec, M. Aftosmis, and M. Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. In *46th AIAA Aerospace Sciences Meeting*, Reno, NV, January 7-10 2008. AIAA 2008-0725.

7. A. Magnus and M. Epton. Panair: A computer program for predicting subsonic or supersonic linear potential ows about arbitrary congurations using a higher order panel method. Technical report, NASA, August 1980. Tech. Rep. CR-3251.

8. H. Akima. A method of univariate interpolation that has the accuracy of a third-degree polynomial. *Association for Computing Machinery Transactions on Mathmatical Software*, 17(3):341–366, 1991.

9. H. Ashley and M. Landahl. *Aerodynamics of Wings and Bodies*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1965.

10. R. J. Mack and N. S. Kuhn. Determination of extrapolation distance with pressure signatures measured at two to twenty span lengths from two low-boom models. Technical Report TM-2006-214524, NASA, 2006.

11. *PASS SE, Program for Aircraft Synthesis Studies-Supersonic Edition*. Desktop Aeronautics, Inc., Palo Alto CA, 2002.

12. I. Kroo. An interactive system for aircraft design and optimization. In *Proceedings of the AIAA Aerospace Design Conference*, February 1992. AIAA Paper 1992-1192.

13. Li W., Shields E., and Geiselhart K. A mixed-fidelity approach for design of low-boom supersonic aircraft. In *48th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, FL, January 2010. AIAA 2010-0845.

14. J. Rodriguez, J. Renaud, and L. Watson. Convergence of trust region augmented lagrangian methods using variable fidelity approximation data. *Structural Optimization*, 15:121–156, 1998.

15. R. P. Brent. *Algorithms for Minimization Without Derivatives*, chapter 3-4. Prentic-Hall, 1973.

16. J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

17. R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *20th International Conference on Very Large Data Base Conference*, Santiago, Chile, 1994.

18. D. Rajnarayan, A. Haas, and I. Kroo. A multifidelity gradient-free optimization method and application to aerodynamic design. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia Canada, September 2008. AIAA Paper 2008-6020.

19. M. Eldred and D. M. Dunlavy. Formulations for surrogate-based optimization with data-fit, multifidelity and reduced-order models. In *Proceedings of the* 11^{th} *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2006. AIAA Paper 2006-7117.

20. Seongim Choi, Juan J. Alonso, Ilan Kroo, and Mathias Wintzer. Multi-fidelity design optimization of low-boom supersonic business multi-fidelity design optimization of low-boom supersonic business jets. In *Proceedings of the* 10^{th} *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004. AIAA Paper 2004-4371.

21. D. Rajnarayan, D. H. Wolpert, and I. Kroo. Optimization under uncertainty using probability collectives. In *Proceedings of the* 10^{th} *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,*, Portsmouth, VA, 2006. AIAA Paper 2006-7033.

22. D. Rajnarayan, I. Kroo, and David H. Wolpert. Probability collectives for optimization of computer simulations. In *Proceedings of the* 48^{th} *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Honolulu, HI, 2007. AIAA Paper 2007-1975.

23. L. C. W. Dixon and G. P. Szegö. The global optimisation problem: An introduction. *Towards Global Optimization*, 2:1–15, 1978.

24. Matthias Schonlau, William J. Welch, and Donald R. Jones. Global versus local search in constrained optimization of global versus local search in constrained optimization of computer models. Technical Report 83, National Institute of Statistical Sciences, March 1998.

25. J. Mockus, V. Tiesis, and A. Žilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

26. D. D Cox and S. John. SDO: A statistical method for global optimization. In N. Alexandrov and M.Y. Hussaini, editors, *Multidisciplinary Optimization: State of the Art*, pages 315–329. SIAM Publications, 1997.

27. J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10(2-3):191–203, 1984.

28. Mathias Wintzer and Peter Sturdza. Conceptual design of conventional and oblique wing configurations for small supersonic aircraft. In *Proceedings of the* 44^{th} *AIAA Aerospace Sciences Meeting and Exhibit*, January 2006. AIAA Paper 2006-930.

29. A. March and K. Willcox. A provably convergent multifidelity optimization algorithm not requiring high-fidelity gradients. 6th AIAA Multidisciplinary Design Optimization Specialist Conference, Orlando, FL, 2010. AIAA 2010-2912.

30. N.M. Alexandrov, J. Dennis, R.M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. Technical Report CR-201745, NASA, October 1997.

31. N.M. Alexandrov, R.M. Lewis, C. Gumbert, L. Green, and P. Newman. Optimization with variable-fidelity models applied to wing design. Technical Report CR-209826, NASA, December 1999.

32. N.M. Alexandrov, R.M. Lewis, C. Gumbert, L. Green, and P. Newman. Approximation and model management in aerodynamic optimization with variable-fidelity models. *AIAA Journal*, 38(6):1093–1101, November-December 2001.

33. J. Castro, G. Gray, A. Giunta, and P. Hough. Developing a computationally efficient dynamic multilevel hybrid optimization scheme using multifidelity model interactions. Technical Report SAND2005-7498, Sandia, November 2005.

34. M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society*, 63(2):425–464, 2001.

35. M. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

36. S. Leary, A. Bhaskar, and A. Keane. A knowledge-based approach to response surface modelling in multifidelity optimization. *Journal of Global Optimization*, 26:297–319, 2003.

37. R.G. Carter. On the global convergence of trust region algorithms using inexact gradient information. *SIAM Journal of Numerical Analysis*, 28(1):251–265, 1991.

38. R. Oeuvray. *Trust-Region Methods Based on Radial Basis Functions with Application to Biomedical Imaging*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 2005.

39. A.R. Conn, K. Scheinberg, and L. Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming*, 111(1-2):141–172, 2008.

40. A.R. Conn, K. Scheinberg, and L. Vicente. Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points. *SIAM Journal of Optimization*, 20(1):387–415, 2009.

41. S.M. Wild, R.G. Regis, and C.A. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal of Scientific Computing*, 30(6): 3197–3219, 2008.

42. S.M. Wild. *Derivative-Free Optimization Algorithms for Computationally Expensive Functions*. PhD thesis, Cornell University, January 2009.

43. S.M. Wild and C. A. Shoemaker. Global convergence of radial basis function trust-region algorithms. Technical Report Preprint ANL/MCS-P1580-0209, Mathematics and Computer Science Division, February 2009.

44. A. March and K. Willcox. Convergent multifidelity optimization using bayesian model calibration. 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Fort Worth, TX, 2010.

45. 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):1–13, February 1999.

46. M. J. Sasena, P. Papalambros, and P. Goovaerts. Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34(3):263–278, 2002.

47. T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

48. T. G. Kolda, R. M. Lewis, and V. Torczon. A generating set direct search augmented lagrangian algorithm for optimization with a combination of general and linear constraints. Technical Report SAND2006-5315, Sandia, August 2006.

49. R. M. Lewis and V. Torczon. A direct search approach to nonlinear programming problems using an augmented lagrangian method with explicit treatment of linear constraints. Technical Report WM-CS-2010-01, College of William and Mary Department of Computer Science, January 2010.

50. G. Liuzzi and S. Lucidi. A derivative-free algorithm for inequality constrained nonlinear programming via smoothing of an l_∞ penalty function. *SIAM Journal of Optimization*, 20(1):1–29, 2009.

51. 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.

52. A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. MPS/SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009.

53. J. Nocedal and S.J. Wright. *Numerical Optimization, 2nd ed.* Springer, 2006.

54. D. P. Bertsekas. *Nonlinear Programming, 2nd ed.* Athena Scientific, 1999.

55. A.R. Conn, N.I. Gould, and P.L. Toint. *Trust-Region Methods*. MPS/SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

56. V.L. Rvachev. On the analytical description of some geometric objects. Technical Report 4, Reports of Ukrainian Academy of Sciences, 1963. (in Russian).

57. P. Y. Papalambros and D. J. Wilde. *Principles of Optimal Design 2nd ed.* Cambridge University Press, 2000.

58. MathWorks, Inc. Constrained nonlinear optimization. Optimization Toolbox User's Guide, V. 5, 2010.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 12 - 2010	Contractor Report	

4. TITLE AND SUBTITLE
Multifidelity Analysis and Optimization for Supersonic Design

5a. CONTRACT NUMBER
NNL07AA33C

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kroo, Ilan; Willcox, Karen; March, Andrew; Haas, Alex; Rajnarayan, Dev; Kays, Cory

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
984754.02.07.07.12.04

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/CR-2010-216874

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 64
Availability: NASA CASI (443) 757-5802

13. SUPPLEMENTARY NOTES
Langley Technical Monitor: Natalia Alexandrov

14. ABSTRACT

Supersonic aircraft design is a computationally expensive optimization problem and multifidelity approaches over a significant opportunity to reduce design time and computational cost. This report presents tools developed to improve supersonic aircraft design capabilities including: aerodynamic tools for supersonic aircraft configurations; a systematic way to manage model uncertainty; and multifidelity model management concepts that incorporate uncertainty. The aerodynamic analysis tools developed are appropriate for use in a mutltifidelity optimization framework, and include four analysis routines to estimate the lift and drag of a supersonic airfoil, a multifidelity supersonic drag code that estimates the drag of aircraft configurations with three different methods: an area rule method, a panel method, and an Euler solver. In addition, five multifidelity optimization methods are developed, which include local and global methods as well as gradient-based and gradient-free techniques.

15. SUBJECT TERMS
Optimization; Multifidelity; Derivative-free; Multidisciplinary; Supersonic

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	119	(443) 757-5802