

IMPROVING ALGORITHMIC EFFICIENCY
OF AIRCRAFT ENGINE DESIGN FOR
OPTIMAL MISSION PERFORMANCE

THESIS

Paul T. Millhouse, Captain, USAF

AFIT/GOR/ENY/98M-02

19980423 067

Approved for public release, distribution unlimited

Disclaimer Statement

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

IMPROVING ALGORITHMIC EFFICIENCY OF AIRCRAFT ENGINE DESIGN
FOR OPTIMAL MISSION PERFORMANCE

THESIS

Presented to the faculty of the Graduate School of Engineering
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Operations Research

Paul T. Millhouse, B.S.

Captain, USAF

March 1998

Approved for public release, distribution unlimited

IMPROVING ALGORITHMIC EFFICIENCY OF AIRCRAFT ENGINE DESIGN
FOR OPTIMAL MISSION PERFORMANCE

Paul T. Millhouse, B.S.
Captain, USAF

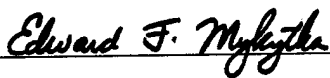
Approved:



Chairman

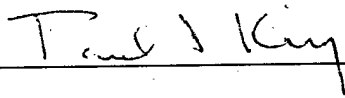
9 MAR 98

Date



9 Mar 98

Date



9 Mar 98

Date

Acknowledgments

Foremost, I would like to thank my wife, Kimberly, for her selfless giving throughout this entire thesis process. Through the hundreds of hours that I spent working on this research, she kept things going at home for our two young children. Additionally, she was always available for encouragement when things were not going well or when the work-load seemed insurmountable. Without her faithful support, this project would never have come together.

I would also like to thank my thesis committee, Lt Col Stuart Kramer, Dr. Edward Mykytka and Dr. Paul King, for their direction and assistance. I appreciate their trust in my judgement as the focus of this research was determined, and the timely advice they provided as I worked through the various problems encountered. These gentlemen were consistently available when I needed their help and provided invaluable expertise in exploring the techniques used in this thesis. Thank you for all of your hard work on this research over the last 12 months.

Table of Contents

Acknowledgments.....	iii
Table of Contents.....	iv
List of Figures.....	vi
List of Tables.....	ix
Nomenclature.....	x
Abstract.....	xii
1. Introduction.....	1
1.1 Problem Background.....	1
1.2 Research Focus.....	2
1.3 Research and Thesis Report Limitations.....	4
1.4 Thesis Organization.....	5
2. Literature Review and Problem Background Information.....	6
2.1 Previous Work on Determining Optimal Engine Parameters for Minimum Fuel Consumption.....	6
2.2 Current Research Overview.....	7
3. Jet Engine Optimization Dimension Reduction.....	21
3.1 Engine Cycle Modeling Improvements.....	22
3.2 Optimal Fan Pressure Ratio (π_{c^*}) Dependency.....	23
3.3 Understanding How (S) and (F) Are Coincidentally Optimized.....	25
3.4 Determining Optimal Reference Engine Mass Flow (m_0).....	43
3.5 Conclusions on Jet Engine Optimization Dimension Reduction.....	51
4. Kriging Techniques.....	53
4.1 Overview.....	53

4.2 Geostatistics	54
4.3 Kriging	71
4.4 Evaluating the Quality of the Kriged Estimate	82
4.5 Determining Feasibility	87
4.6 Kriging and the Use of Penalty Functions	97
4.7 Application of Kriging and the Feasible Region Checking Algorithm to Jet Engine Optimization	101
5. Final Conclusions and Recommendations for Future Research	112
5.1 Summary of Conclusions	112
5.2 Recommendations for Future Research	113
Appendix A: Kriging and Feasible Region Checking Algorithm Computer Codes	115
Appendix B: Standard Genetic Algorithm Operation	128
Bibliography	136
Vita	138

List of Figures

Figure 2-1. Micro-Genetic Algorithm Flow Diagram.....	14
Figure 2-2. Funneling Effect Created by Penalty Functions for a Minimizing Optimization	17
Figure 2-3. Comparison of Regression and Interpolation.....	19
Figure 3-1. On-Design Uninstalled S and Specific Thrust for the Ideal Mixed-Stream Turbofan Engine	26
Figure 3-2. On-Design Condition #1: Uninstalled S and Specific Thrust for the TERMAP Generated Mixed-Stream Turbofan Engine at Varying α	27
Figure 3-3. On-Design Condition #2: Uninstalled S and Specific Thrust for the TERMAP Generated Mixed-Stream Turbofan Engine at Varying α	28
Figure 3-4. On-Design Uninstalled S and Specific Thrust for the TERMAP Generated Mixed-Stream Turbofan Engine at Varying π_c	29
Figure 3-5. Optimal On-Design Uninstalled S and Specific Thrust at $\frac{P_{t5'}}{P_{t5}} \approx 1.00$ for Varying α	30
Figure 3-6. Optimal On-Design Uninstalled S and Specific Thrust at $\frac{P_{t5'}}{P_{t5}} \approx 1.00$ for Varying π_c	31
Figure 3-7. $\frac{P_{t5'}}{P_{t5}}$ Variation with Changing π_c at Various α	32
Figure 3-8. Off-Design Flight Condition #1: Optimal Off-Design Uninstalled S and Specific Thrust at Reference Engine $\frac{P_{t5'}}{P_{t5}} \approx 1.00$	33
Figure 3-9. Off-Design Flight Condition #2: Optimal Off-Design Uninstalled S and Specific Thrust at Reference Engine $\frac{P_{t5'}}{P_{t5}} \approx 1.00$	34
Figure 3-10. Off-Design Flight Condition #3: Optimal Off-Design Uninstalled S and Specific Thrust at Reference Engine $\frac{P_{t5'}}{P_{t5}} \approx 1.00$	35

Figure 3-11. Variation of Off-Design $\frac{P_{15}}{P_{15}}$ Ratio with Changes in Chosen Values of On-	
Design $\frac{P_{15}}{P_{15}}$ Ratio for Off-Design Condition #1	37
Figure 3-12. Variation of Off-Design $\frac{P_{15}}{P_{15}}$ Ratio with Changes in Chosen Values of On-	
Design $\frac{P_{15}}{P_{15}}$ Ratio for Off-Design Condition #2	38
Figure 3-13. Variation of Off-Design $\frac{P_{15}}{P_{15}}$ Ratio with Changes in Chosen Values of On-	
Design $\frac{P_{15}}{P_{15}}$ Ratio for Off-Design Condition #3	39
Figure 3-14. Process Flow for Determining Optimal Engine Mass Flow with Constant Installation Losses.....	47
Figure 4-1. Response Surface for 2-D Kriging Example	58
Figure 4-2. Sample Measurement Locations for 2-D Kriging Example	59
Figure 4-3. Plot of Residuals for 2-D Kriging Example	61
Figure 4-4. Graphical Representation of Semi-Variogram Data for Kriging Example	64
Figure 4-5. Kriging Example Semi-Variogram with Fitted Linear Model	70
Figure 4-6. Selection of Closest Points for Estimating $v_0 = f(x_0)$	77
Figure 4-7. Determining \bar{h}_0 Threshold to for a Desired Kriging Prediction Error Magnitude	87
Figure 4-8. Possible Grid Structure for $q = 2$	89
Figure 4-9. Initialized Feasible/Infeasible Array for Feasible Region Checking Algorithm	92
Figure 4-10. Updated Feasible/Infeasible Array for Feasible Region Checking Algorithm	93
Figure 4-11. Typical Feasible/Infeasible Array Structure After Processing Sampled Points	93
Figure 4-12. Funneling Effect Created by Penalty Functions for a Minimizing Optimization	98
Figure 4-13. Example Determination of Closest Feasible Point.....	100

Figure 4-14. Interaction of Optimizer with Response Function and Estimation

Algorithms 103

Figure 4-15. Variable Coding 107

List of Tables

Table 4-1. Response Evaluations for Kriging Example Sample Points.....	60
Table 4-2. Regression Residuals for Kriging Example	62
Table 4-3. Exhaustive Semi-Variogram Calculations for Kriging Example	63
Table 4-4. Tabular Listing of Semi-Variogram Data Used for Creating Linear Model	69
Table 4-5. Distance and Semi-Variogram Data Between Sample Points and x_0 for Kriging Example.....	79
Table 4-6. Matrix of Distances Between Sample Points for Kriging Example	79
Table 4-7. Matrix of Semi-Variogram Values Between Sample Points for Kriging Example	79
Table 4-8. Predictor Variable Correlation with Prediction Error Magnitude.....	85
Table 4-9. Logic Matrix for Feasible Region Checking Algorithm	91
Table 4-10. Jet Engine Optimization Results With Kriging Disabled and Enabled.....	109
Table 4-11. t-Test Results Comparing Optimization Performance With and Without Kriging	111

Nomenclature

Chapter 1 and 2 Notation:

m_0	Overall engine air mass flow (lbm/sec)
N	Standard Genetic Algorithm (SGA) population size
W_{TO}	Gross takeoff weight (lbf)
W_E	Aircraft empty weight (lbf)
W_F	Weight of all fuel on-board aircraft (including reserve fuel) (lbf)
W_P	Aircraft payload weight (lbf)
α	Bypass ratio
π_c	Fan compressor pressure ratio
π_c	High pressure compressor (core) pressure ratio

Chapter 3 Notation:

A_{0ref}	On-design free-stream cross-sectional area
A_I	Engine inlet cross-sectional area
f_0	Overall engine fuel-to-air ratio
F	Uninstalled engine thrust (lbf)
m_0	Overall engine air mass flow (lbm/sec)
M_0	Free-stream Mach number
M_5	Core flow Mach number at turbine exit
$M_{5'}$	Bypass flow Mach number at mixer entry
P_{i5}	Core flow total pressure at turbine exit
$P_{i5'}$	Bypass flow total pressure at mixer entry
S	Reference wing area; Thrust specific fuel consumption (lbf/lbm/sec)
T_{i4}	Core burner total temperature ($^{\circ}$ R)
W_{TO}	Gross aircraft takeoff weight (lbf)
α	Bypass ratio
β	Aircraft weight fraction
π_c	Fan compressor pressure ratio

π_c

High pressure compressor (core) pressure ratio

Chapter 4 Notation: g_q Grid sub-division in the q dimension in which a new point is located h

Euclidean distance between two points

 h_{range}

Range of the semi-variogram

 \bar{h}_0 Average of the distances between the n points used for kriging and the point being estimated m

Total number of points sampled to generate a semi-variogram

 n

Number of points used to generate a kriged estimate

 n_q Total number of grid sub-divisions in the q dimension $N(h)$ Number of pairs of points in a sample that are a distance h apart p

Number of independent design variables in an optimization problem

 q

Number of independent design variables that have infeasible values

 r Number of independent design variables that do not have infeasible values ($p = q + r$) s^2

Sample variance

 v

Response value associated with a set of design variables

 \hat{v} Response estimate of the actual response v \bar{v}

Average response value for all points sampled

 w

Weight applied in a linear combination of terms

 \bar{w}

Vector of minimum variance weights

 Δ_{Residual}

Difference between two points' response value residuals

 $\gamma(h)$ Semi-variogram value for a distance h σ_0^2

Variance of a kriged estimate

 ∂

Partial derivative

 Γ Matrix of semi-variogram values for the distances between all possible pairings of the n points being used for kriging $\bar{\gamma}_0$ Vector of semi-variogram values for the distances between the n points used for kriging and the point being estimated

Abstract

Automated techniques for selecting jet engines that minimize overall fuel consumption for a given aircraft mission have recently been developed. However, the current techniques lack the efficiency required by Wright Laboratories. Two noted dependencies between turbine engine fan pressure ratio, bypass ratio, high pressure compressor pressure ratio and overall engine mass flow allows for a reduction in the number of independent design variables searched in the optimization process. Additionally, through the use of spatial statistics (specifically kriging estimation), it is possible to significantly reduce the number of time-consuming response function evaluations required to obtain an optimal combination of engine parameters. A micro-Genetic Algorithm (μ GA) is employed to perform the non-linear optimization process with these two computation-saving techniques. Optimal engine solutions were obtained in 25% of the time required by previous automated search algorithms.

IMPROVING ALGORITHMIC EFFICIENCY OF AIRCRAFT ENGINE DESIGN FOR OPTIMAL MISSION PERFORMANCE

1. Introduction

1.1 Problem Background

When a new aircraft is to be developed by the United States Department of Defense (DoD), a request for proposal (RFP) is established to outline the flight conditions and performance requirements of a typical mission. From this RFP, aircraft manufacturers and engine companies are able to initiate the conceptual aircraft design process. Conceptual aircraft design involves the use of many simplifying assumptions integrated with computer modeling codes to generate a first-cut aircraft and engine solution to meet the RFP requirements. While this initial aircraft design may be significantly different from the final aircraft design, its importance should not be underestimated – without a realistic starting design, future modifications can be frustrated by having to re-visit basic design issues.

One of the objectives of the conceptual aircraft design is to make the aircraft as efficient as possible. Improved efficiency is always desirable – it translates to lighter, faster, and more maneuverable aircraft. As a means to this end, efficiency improvement can be attacked on several different fronts, to include using lighter construction materials and reducing individual component and airframe inefficiencies. However, one of the most important aspects of ensuring maximum aircraft efficiency is the proper selection of an engine for the airframe.

Although many different engine configurations can provide sufficient thrust to an airframe capable of meeting RFP requirements, not all engines are equally efficient at the different mission flight conditions. While an engine may be well-suited for efficient operation at one flight condition, this same engine may be terribly inefficient at another flight condition. Based on the mission specified (meaning the flight conditions and the duration of operation required at each flight condition), an optimal engine (with possible alternate optimal engines) exists that will minimize the fuel required for the aircraft to complete the mission. It is this optimal engine design that we seek in this project.

In classical optimization terms, this problem can be viewed as an effort to minimize overall fuel consumption for a non-linear fuel consumption response function. This minimization is subject to the engine satisfying the thrust constraints implied by the mission profile. Additionally, only certain combinations of engine parameters produce engines that can physically sustain operation. Thus, written in math programming notation, the problem at hand is:

Minimize {Overall Fuel Consumption}

Subject To:

 Viable Combinations of Engine Parameters

 Mission Requirements

1.2 Research Focus

Identifying the best engine design for an aircraft mission has been an issue ever since the beginning of aviation. Although solution techniques have improved over the years, the time and

effort required to identify the most efficient engine design has remained high. In recent years, advances in conceptual engine-airframe matching techniques have allowed the entire engine search process to be automated. Although these advances greatly speed the conceptual engine design process, the amount of computer processing still remains at a level that makes aircraft design experimentation impractical.

The purpose of this research, which is sponsored by the U.S. Air Force's Wright Laboratories, is to identify means by which the automated engine search process can be accelerated. By expediting this engine matching process, Wright Laboratories hopes to more efficiently explore new aircraft technologies and focus its efforts on aircraft improvements that will most significantly improve overall mission performance. Two main approaches to streamlining engine optimization will be researched:

1. Exploration of any possible relationships between the design variables that may allow for the total number of independent variables to be reduced. The reduction in design variables will allow the non-linear optimizer to converge to an optimal solution with less objective function evaluations. This reduction in evaluations may be offset by increased processing time (to implement the observed dependencies) required for each evaluation. Thus the overall time required to locate an optimal solution may or may not be improved. One additional benefit is that more efficient engine solutions may be obtained if it is possible to directly select optimal values of the variables that have been eliminated.
2. The use of an unbiased minimum variance estimation technique called *kriging*. Evaluating the fuel consumption implied by each combination of design variables involves the use of relatively time-consuming computer

codes. If estimation techniques can be used to bypass the actual response function evaluation codes, significant time-savings in the overall optimization may be realized.

While the dimension reduction techniques to be explored will only be applicable to mixed-stream turbofan engine design, kriging, if successful, is potentially useful in any number of optimization applications involving time-consuming objective function evaluations. In this application, it may be possible to combine both of these potential time-saving techniques to further reduce processing required to obtain optimal solutions.

1.3 Research and Thesis Report Limitations

For the purposes of this study, only the mixed-stream turbofan engine will be considered. While many other turbine engine cycles exist, the mixed-stream turbofan is most commonly used in modern, high-performance DoD aircraft. Additionally, only four aspects of the engine have been treated as variables – reference engine bypass ratio (α), fan compressor pressure ratio (π_c), high pressure (or core) compressor pressure ratio (π_c) and overall engine air mass flow (m_0). Any other engine parameters that may typically be considered variable by engine designers will be assumed to be constant, regardless of the values of these four design variables.

Since the audience of this research summary is potentially academically mixed – some with optimization expertise, others with turbine engine design expertise – it is worth stating that the thesis will be geared to focus on optimization. Although some jet engine discussion may be required to understand the dimension reduction techniques, this thesis will treat the engine cycle analysis algorithms and governing equations of flight inherent to evaluating a turbine engine's mission performance as *black-box* response functions. This does not imply that the inner workings of these computer codes are not important to this project. However, the focus of this

thesis will be to shed new light on the use of these algorithms, not to re-state the principles that were used to create them. The reader is referred to Hill and Peterson (1970) for a more thorough understanding of turbo-machinery operation. Additionally, a thorough discussion of jet engine optimization methods and the governing equations of flight can be found in Mattingly, et al. (1987).

1.4 Thesis Organization

It will first be necessary to present foundational information to understand the context in which kriging and dimension reduction techniques will be explored. In doing this, we will review previous work done on the engine optimization problem and familiarize the reader with the optimization tools and algorithms under-girding the entire optimization effort. Once the reader is comfortable with the overall optimization problem and how it is being solved, two potential dimension reduction techniques will be investigated in hopes of streamlining the optimization process. Then, using the reduced-dimension problem as a starting point, application of the kriging estimation technique to further streamline the optimization process will be discussed and results presented. Finally, overall conclusions will be drawn about the successes and failures of this research, and recommendations will be made for future areas of research.

2. Literature Review and Problem Background Information

The primary goal of this research is to identify methods that expedite the automated optimization of a jet engine for a specified aircraft mission. In order to achieve this goal, two primary methods will be investigated – problem dimension reduction and kriging techniques. In order to understand the context in which these techniques will be applied, it will be useful to first review past research accomplishments on the jet engine optimization problem and then introduce the underlying optimization framework of this project. Presentation of the current optimization framework will include discussion on improvements made to the engine cycle codes used during this analysis, the introduction of a fifth independent variable (gross takeoff weight (W_{TO})), and the use of a micro-Genetic Algorithm (μ GA) optimizer algorithm to search the design space. The dimension reduction techniques and kriging, which are the focus of this research, are also briefly introduced, although detailed discussion on these topics is reserved for Chapters 3 and 4, respectively.

2.1 Previous Work on Determining Optimal Engine Parameters for Minimum Fuel Consumption

Given a specified aircraft mission and payload, it is possible to design an engine which minimizes the total fuel consumed to execute an aircraft mission. Computer based engine cycle analysis and mission performance algorithms can be combined manually to perform mission optimization. This method of optimization is extremely time consuming, requiring numerous engine design iterations and much trial-and-error searching throughout a large, multi-dimensional design space for a relatively small feasible solution region. (Mattingly, et al., 1987)

In a significant step towards making this design process faster, Nadon (1996) combined a genetic algorithm optimization routine with the various Mattingly (1990) algorithms into a single, automated package capable of locating the optimal engine parameters for a given aircraft mission. Nadon showed that by introducing extra design variables and penalty functions for infeasible solutions, optimal engine designs could be located without human assistance. However, Nadon's ability to locate an optimal engine solution was significantly hindered by inherent limitations in the engine cycle analysis codes used. These engine cycle analysis limitations forced Nadon to treat many truly feasible solutions as infeasible, thus distorting the solution space and, quite possibly, causing a sub-optimal engine solution to be selected as the optimal solution. Additionally, Nadon's optimization algorithms were dogged by the number of independent design variables, which had to be increased (via the introduction of extra design variables) to facilitate the automated optimization. Although significantly more efficient than performing engine optimization manually, Nadon's automated optimizations required longer computer run-times than the research sponsor (Wright Laboratories) desired. Regardless of the practical short-comings, he demonstrated that automated mission optimization is possible and laid the ground work for future projects (such as this one) to improve both the quality of the optimal engine solutions and the time required to locate these solutions. (Nadon, 1996).

2.2 Current Research Overview

Before discussion about computation-saving techniques is possible, it is necessary to establish some of the underlying tools and methods that will be utilized while exploring the use of kriging and problem dimension reduction.

Improvements to the Engine Cycle Computer Codes. A significant improvement over Nadon's work made prior to this research was the incorporation of a more robust engine cycle

evaluation algorithm. Turbine Engine Reverse Modeling Aid Program (TERMAP), an on-design and off-design engine evaluation code created for the U.S. Air Force's Wright Laboratories, was integrated with Mattingly's (1987) mission fuel consumption equations. The primary advantage of using TERMAP as the engine cycle code evaluator is that it uses compressor and turbine mapping information to produce engine performance results at unchoked engine conditions. Additionally, since this research was sponsored by Wright Laboratories, the end users of this optimization algorithm were more comfortable with the TERMAP engine cycle results than the Mattingly engine cycle results (Wright Laboratories has used TERMAP for its engine research for many years). It is worth noting that, for choked engine conditions, engine cycle results were similar regardless of whether TERMAP or Mattingly's (1990) engine equations were used.

The Rubber Aircraft and the Introduction of Gross Takeoff Weight as an Independent Variable. The amount of fuel required to perform a mission profile is directly affected by the gross takeoff weight (W_{TO}) of the aircraft. Given two identical airframes, one loaded with more weight than the other, the heavier aircraft will require more thrust and will consume more fuel during flight than the lighter aircraft. This is because the aerodynamic forces used to generate lift on an aircraft also create drag as a byproduct.

Just as mission fuel consumption is affected by gross takeoff weight, W_{TO} is affected by the fuel stores required to perform the mission (which is directly related to mission fuel consumption). W_{TO} is not only affected by the weight of the fuel itself, but also, in the context of conceptual aircraft design, the weight of the airframe structure required to support the weight of the fuel. In conceptual aircraft design, the proper size of the airframe and engine is still undetermined. When the amount of fuel required to perform a mission changes, so does the size of the airframe required to carry the fuel stores. Thus, in conceptual aircraft design, the size of the aircraft is a design variable, just as the engine parameters being optimized. Because of this

dynamic re-sizing of the aircraft, it is known as a rubber aircraft during conceptual design. This inherent dependency between the conceptual airframe and the engine being optimized requires special handling to locate the optimal combination of both. (Mattingly, et al., 1987, Chapters 1 and 2)

As previously presented in Chapter 1, there are four independent engine variables that are being considered in this optimization application: reference engine bypass ratio (α), fan compressor pressure ratio (π_c), high pressure (or core) compressor pressure ratio (π_c) and overall engine air mass flow (m_o). For each set of viable engine parameters, there exists a minimum W_{TO} that is needed to perform the specified mission. W_{TO} is calculated by:

$$W_{TO} = W_E + W_F + W_P \quad (2-1)$$

where

W_E is the empty weight of the aircraft. For conceptual aircraft design, W_E can be

modeled as a function of the gross takeoff weight so that $W_E = f(W_{TO})$.

Looking at Eq (2-1) and noting that $W_{TO} = f^{-1}(W_E)$, we see that for a

fixed W_P , it is possible to solve for W_E so that W_E becomes solely a

function of W_F . (Mattingly, et al., 1987, 69-70)

W_F is the weight of the fuel required to perform the mission plus any required

fuel reserves

W_P is the payload weight, where payload includes any item (other than fuel) not

fixed to the aircraft. This includes the pilot, equipment and weapons

ordnance.

Notice that W_{TO} is a function of W_F , which is a function of W_{TO} .

This dependency of W_{TO} and W_F can be overcome by treating W_{TO} as an independent variable. W_{TO} is selected in conjunction with the independent engine parameters in an optimization. This W_{TO} is treated as the gross takeoff weight for the set of engine variables being evaluated for fuel consumption. After evaluating the fuel consumption implied by these five variables, the resulting $W_{F(Actual)}$ is compared with the theoretical amount of fuel on board the aircraft ($W_{F(Theoretical)}$), as implied by the gross takeoff weight. Manipulating Eq (2-1):

$$W_{F(Theoretical)} = W_{TO} - W_E - W_P \quad (2-2)$$

where now

W_E is as defined in Eq (2-1)

W_{TO} is the independent gross takeoff weight variable value for this evaluation

W_P is as defined in Eq (2-1)

Based on the function being used to relate W_E and W_{TO} , $W_{F(Theoretical)}$ can be calculated for comparison with $W_{F(Actual)}$.

If $W_{F(Actual)} > W_{F(Theoretical)}$, the point is infeasible (more fuel was needed to perform the mission than was available on-board the aircraft); if $W_{F(Theoretical)} \geq W_{F(Actual)}$, the point is feasible.

The optimal W_{TO} value for any set of viable engine parameters occurs when $W_{F(Theoretical)} =$

$W_{F(Actual)}$. When this condition is met, enough fuel exists to perform the mission, but no excess fuel (and structure) weight has been unnecessarily carried throughout the mission.

By treating W_{TO} as an independent variable, the optimization algorithm is responsible for locating the best gross takeoff weight for the best set of engine parameters. By penalizing the response function value returned to the optimizer for designs with infeasible W_{TO} 's, gross takeoff weights are driven into feasible values. W_{TO} 's that are too large are inherently penalized by the

excess weight that was carried throughout the mission. This excess weight causes the fuel consumption to be higher than if $W_{F(Theoretical)} = W_{F(Actual)}$.

Probabilistic Non-Linear Optimization Algorithms. Probabilistic, non-linear optimizers use randomization techniques to search an entire design space for the *global optimum*, or the point that produces the best response in the design space. This approach to optimization is contrasted with gradient-based optimizers which use various techniques to estimate the direction of greatest improvement. Gradient-based optimizers only have the ability to locate *local optima*, or the best points from sub-sets of the design region, and do not have the ability to search outside of the local region in which they find this optimal point.

The Standard Genetic Algorithm. The *Standard Genetic Algorithm (SGA)*, also known simply as the *Genetic Algorithm (GA)*, is a popular non-linear, probabilistic optimization technique that simulates survival-of-the fittest processes to locate an optimal combination of input design variables. The SGA's primary advantage is its robust ability to identify globally optimal solutions in highly irregular, even non-continuous response function environments. However, this robustness comes at the price of large numbers of objective function evaluations required to locate the optimal solution. Given the highly non-linear nature of the mission fuel consumption response function, this type of optimizer is well-suited for this application. (Nadon, 1996, 16)

In the SGA, sets of design variable coordinates ($x_1, x_2, \dots x_n$) are encoded into a single string of binary digits (i.e. 011101001010111010... – see Appendix B for details on how this encoding is performed). This string of binary numbers represents a *chromosome* that genetically describes the point, much like genes in biological organisms. This chromosome, which can be decoded back to the original coordinates, is unique to this point and, therefore, will not be repeated by any other point in the design space. The SGA manipulates a collection of points

(whose coordinates are converted into chromosomes) known as a *population*. Each member of this population has a certain level of *fitness*, which is analogous to the point's response function value (favorable response function values equate to high fitness levels). Hence the stage is set for the SGA to simulate survival-of-the-fittest processes until a globally optimal point (or, in SGA terms, a most fit member) is located.

An SGA uses three biological processes to search a design space for an optimal solution – selection, crossover and mutation. *Selection* is the process by which the most-fit members of a population are chosen for mating (crossover). This is where the survival-of-the-fittest premise of the SGA is introduced. The likelihood of a member being selected for mating is directly proportional to the member's fitness. In other words, the most fit members are most likely to be chosen to generate the next population. During the selection process, it is also possible to implement a technique known as *elitism*, which simply guarantees that the most fit member of a population is always passed-on to the subsequent population. This ensures that the chromosome with the most favorable fitness value remains in the population for further manipulation and is not lost during the selection process. *Crossover* is the technique used to mate the members selected for reproduction. It typically involves combining the chromosome bit structures of the parents to produce two new children chromosome structures. An important point to note about crossover is that, regardless of the crossover scheme used, mating genetically identical parents generates children identical to the parents. This is essential to the SGA eventually converging to an optimal solution. Finally, *mutation* is a random process by which bits in members' chromosomes are spontaneously changed to produce new points in the design space (recall that every chromosome bit structure can be mapped to a unique set of design variable coordinates, and vice versa). More detail about how these genetic algorithm processes work is located in Appendix B.

The net effect of repetitively evaluating a population for fitness, selecting parents, combining parents to make new children members, and randomly introducing mutation is that many chromosome structures are generated and evaluated for fitness. Since each of these unique chromosomes equates to a unique point in the design space, many different regions of the design space are searched. The selection process ensures that when promising design-space regions are encountered (e.g. regions of relatively high fitness), the bit structure of these favorable solutions is maintained and passed-on to future populations.

It can be shown that the repetitive process of selection and crossover will eventually result in a population completely comprised of the most fit member (multiple copies of the same chromosome bit structure). This most-fit chromosome pattern to which the SGA converges is presumably the optimal point which we seek. While many criteria exist to determine if a population has converged, the technique used in this application involves evaluating a population's *homogeneity*, or genetic likeness, to discern when the SGA has completed its task. When most of the members of a population have about the same chromosome bit structure, the ability of the SGA to search new regions has been greatly reduced and the SGA terminates operation.

In all, the genetic algorithm searches a design region by encoding the coordinates of the points evaluated into a genetic structure that can then be manipulated to converge to a most-fit solution. This convergence is a result of the converging properties of the survival-of-the-fittest approach. Through the various populations created, many different genetic schemes are encountered. By ensuring survival-of-the-fittest, the most-fit of these genetic structures (which is the globally optimal solution to the optimization problem) survives to eventually be the only genetic structure in the population. (Pirlot, 1996, 502-506)

The Micro-Genetic Algorithm. The *Micro-Genetic Algorithm* (μ GA) is a probabilistic optimization technique that is closely related to the Standard Genetic Algorithm. Unlike the SGA which relies on manipulation of large populations of chromosome patterns ($N \geq 30$, where N is the population size) to adequately search the design space, the μ GA employs the SGA with a micro-population ($N \leq 5$) combined with an outer-loop operation (see Figure 2-1).

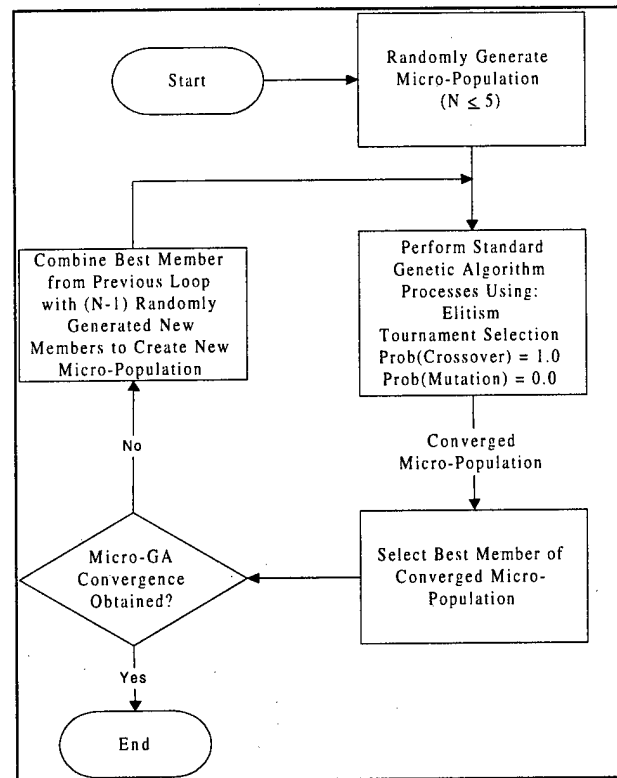


Figure 2-1. Micro-Genetic Algorithm Flow Diagram

In Figure 2-1, we see that a randomly selected initial micro-population is first established and passed to the SGA. The SGA then performs normal genetic algorithm operations using elitism, tournament selection and a 100% crossover rate ($\text{Prob}(\text{Crossover}) = 1.0$). Elitism guarantees that the genetic pattern of the most fit member of the current population is passed on to the subsequent population unaltered by crossover or mutation. This ensures that the most favorable solutions are not lost during the SGA genetic processes. Tournament selection

involves randomly pairing members of a population and selecting the most-fit of the two to be a potential parent for the next population. Each individual tournament pair (multiple tournaments are performed) identifies one population member to become a part of the pool of eligible parents. The parents are then randomly paired to create children for the next population. The 100% crossover rate ensures that all of the identified parents are mated to generate the subsequent population. A crossover rate less than 100% would allow some of the parents genetic schemes to be passed to the next generation without having been modified by the mating process and would reduce the amount of new genetic structures evaluated during the optimization process. Note that the elite member that was maintained from the previous population may also have been selected for crossover. Regardless, the elite member's unaltered genetic pattern is also passed to the new population, so that the new generation only has $(N-1)$ children generated from the previous population.

Mutation is disabled ($\text{Prob}(\text{Mutation}) = 0$) in the μGA . This process, usually responsible for introducing diversity into the population to prevent premature convergence, is replaced with an outer-loop process in the μGA algorithm. For this reason, each micro-population processed by the SGA converges relatively quickly. The SGA is considered converged when the micro-population reaches a user-defined level of homogeneity, at which time the SGA halts and outputs it's final population. The most fit member of the converged micro-population is then combined with $(N-1)$ new, randomly selected members and fed back into the SGA for manipulation. This SGA re-start process is performed until a user-defined outer-loop convergence criteria is met.

The primary advantage of the μGA is efficiency. The μGA has been shown to converge to globally optimal solutions significantly faster than SGA routines, reducing the overall number of objective function evaluations required to obtain an optimal solution. Despite the reduced

number of function calls, the thoroughness of the search process is comparable to the SGA.
(Krishnakumar, 1989, 289-291)

Use of Penalty Functions. The μ GA, like many non-linear optimizers, is not capable of handling infeasible points by itself. It requires a tangible response value be returned every time an objective function call is made. In some applications, the objective function is programmed to return an infeasible response value to the optimizer that is several orders of magnitude worse than any feasible point response could be. This directs the optimizer away from the region by this point in the future. However, for many optimizers, convergence is encouraged when an infeasible point is assigned a penalized response value rather than a single, extremely poor response value. By penalizing an infeasible response based on how far away it is from the feasible region, all available information is utilized in assisting the optimizer in moving back towards the feasible region.

Assume that a point has been determined to be infeasible. If we know the boundaries of the feasible region, we can generate a pseudo-response for this infeasible point based on the response from the closest feasible point. That is, we can, starting with the response from the feasible point, assess a penalty based on how far the infeasible point is from the feasible region. Written as an equation for a minimization problem, the pseudo-response is generated with:

$$\left(\begin{array}{c} \text{Infeasible Point} \\ \text{Pseudo - Response} \end{array} \right) = \left(\begin{array}{c} \text{Closest Feasible} \\ \text{Point's Response} \end{array} \right) + \left(\begin{array}{c} \text{Penalty, based on Distance from} \\ \text{Infeasible Point to Feasible Region} \end{array} \right) \quad (2-1)$$

As depicted for a single independent variable in Figure 2-2, this in effect, funnels the optimizer back towards the feasible region whenever an infeasible point is evaluated.

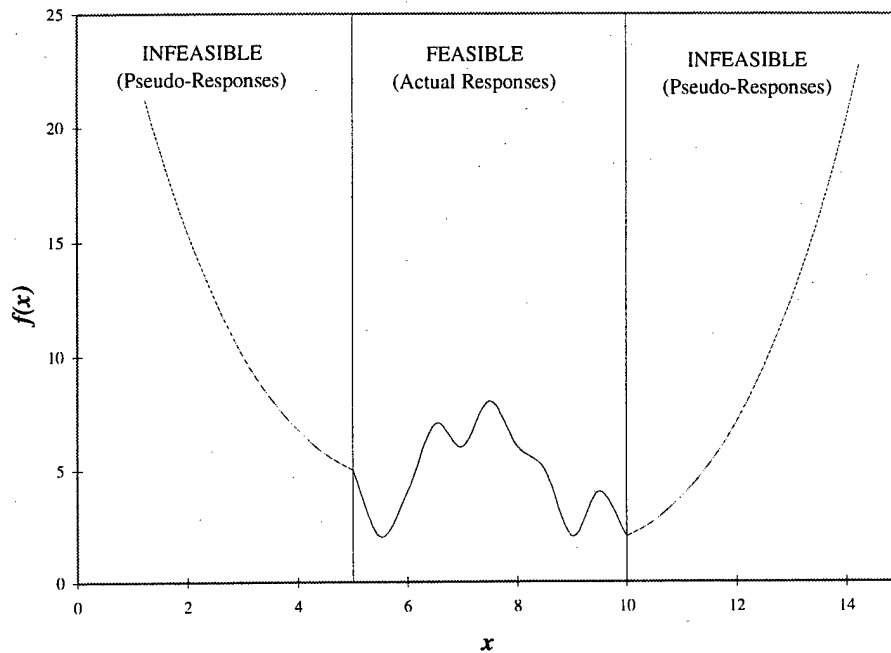


Figure 2-2. Funneling Effect Created by Penalty Functions for a Minimizing Optimization

Unfortunately, as will be discussed in Chapter 4, the actual boundaries of the feasible region are not known a priori in this application. For this reason, the distance to the closest known feasible point (that is, the closest feasible point that has previously been evaluated) is used to estimate the distance to the true feasible region.

Introduction to Research Topics. Dimension reduction refers to the elimination of a portion of the independent design variables in an optimization problem. It is desirable to reduce the number of independent variables because this allows probabilistic non-linear optimizers (like the μ GA) to converge more quickly to an optimal solution. In this application, the baseline optimization process had five independent variables: reference engine bypass ratio (α), fan pressure ratio (π_c), high pressure compressor (core) pressure ratio (π_c), overall engine air mass flow (m_0), and aircraft gross takeoff weight (W_{TO}). Research will be conducted to determine if usable dependencies exist between any of these design variables that may be used to

reduce the dimensions of the problem. While looking for these relationships, it must be remembered that exploiting noted dependencies may or may not be more efficient than simply performing the optimization with the additional independent variables. Achieving this balance of dimension reduction and efficiency will be the focus of the research presented in Chapter 3.

Working with the reduced independent variable set achieved in Chapter 3, kriging is then applied to this problem in hopes of accelerating the optimization process. Kriging is a response estimation technique that uses existing known response values to generate response estimates for new points in the design space. Its origins are traced to the geological sciences where it is used to estimate the size of mineral deposits using limited point samples. Unlike many estimation techniques (like least-squares regression), kriging does not use a pre-defined mathematical expression fit to existing data to estimate responses at new points in the design space. Instead it interpolates between known data points. The primary advantage of this approach is that the estimating function passes through the known data points (see Figure 2-3). Since in the jet engine optimization problem, the best polynomial for the response data is not known a priori and can change based on other user-defined mission parameters, interpolation provides a more robust means of providing accurate estimates. Additionally, since response function evaluations are deterministic (that is, a single response is always obtained with the same input variables), it is appropriate that the estimating function pass through all of the known data points.

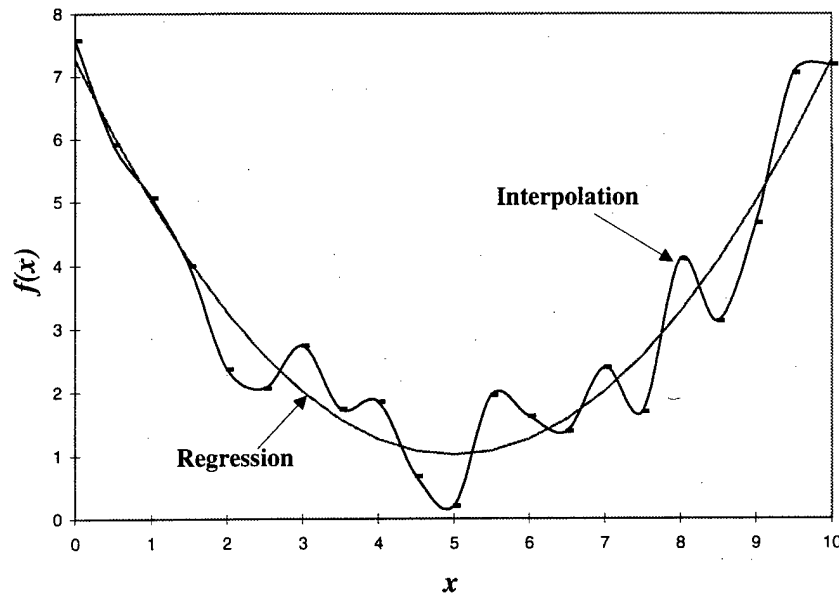


Figure 2-3. Comparison of Regression and Interpolation

Kriging is unique among interpolation techniques because it produces *minimum variance* response estimates. As will be discussed in Chapter 4, kriging makes use of spatial statistics to describe response variation in a design region. Using this spatial information, kriged estimates are generated using linear combinations of known response data to produce response estimates that minimize the effects of the spatial variation on the new point estimate.

Use of kriging in non-geological applications is a new field of study. For this reason, it is uncertain if kriging is even appropriate for jet engine optimization. Additionally, it will be necessary to completely automate the kriging process in order to exploit its use in this application – a task that, based on the literature search performed, appears to have no precedent in aerospace applications. If kriging can be automated for this application, it is uncertain if the computations required to produce kriged estimates will be more efficient than evaluating the actual response function (although, at least conceptually, it does seem like it will be more efficient). So we see that, even though the benefits of kriging look promising, new ground will have to be covered to

make use of the technique. Chapter 4 contains significant detail on the theory and application of kriging to the jet engine optimization problem.

3. Jet Engine Optimization Dimension Reduction

In non-linear optimization, it is often desirable to reduce the number of independent design variables in order to minimize the processing required to locate an optimal solution. If the response function associated with a set of design variables is not guaranteed to be convex (which is often the case in engineering applications), some type of probabilistic optimizer (like the genetic algorithm) will be required to adequately search the design region for the global optimum. It is the necessity of this random search algorithm that drives the need for independent variable reduction – the smaller the set of input variables, the smaller the number of response function calls required for the probabilistic optimizer to converge to an optimal solution.

In many engineering applications, evaluating a response function can consume a significant amount of resources, including computing capability and time. Evaluating one set of design variables can involve executing lengthy, iterative computer codes requiring anywhere from a few seconds to hundreds of hours to complete. In this optimization environment, it is even more important to reduce the number of objective function calls required to locate the optimal solution.

In this application, a preliminary search was performed to locate the set of mixed-stream, turbofan engine design variables – engine bypass ratio (α), fan pressure ratio (π_c), overall core pressure ratio (π_c) and engine air mass flow (m_0) – that minimized the amount of fuel required to perform a defined aircraft mission. At this stage in the design process, I was able to treat the airframe as a rubber aircraft, meaning that the size and weight of the airframe were not fixed. As more efficient engine designs were located (which required less fuel to complete the mission), less fuel was required on-board the aircraft, therefore reducing the size and weight of the

airframe needed to carry the fuel. While the aircraft drag polar and takeoff wing loading $\left(\frac{W_{TO}}{S}\right)$ were assumed to be constant regardless of the size of the aircraft (wing area (S) was varied with gross takeoff weight (W_{TO}) to maintain a constant wing loading), the size (and more importantly the weight) of the airframe was allowed to change during the optimization process. For this reason, W_{TO} was added to the set of design variables to be optimized (see Chapter 2 for details).

While quantifying the computational savings experienced with dimension reduction is important to the overall engine optimization effort, the focus of the research presented in this chapter is to describe the noted dependency of two of the design variables on the other three design variables. In effect, it is proposed that this five dimensional (α , π_c , π_e , m_0 , and W_{TO}) optimization problem can be reduced to a three dimensional (α , π_c , and W_{TO}) optimization problem by exploiting the observed dependencies of π_e and m_0 on the other variables. Exploiting these variable dependencies involves heuristic search algorithms and has a computational price of its own. Therefore, depending on the efficiency of these heuristic algorithms, a total optimization computational savings (with the reduced number of input variables) may or may not be realized. While it was not practical to experimentally prove that the dimension reduction made the optimization process more efficient, discussion of this important aspect of the overall design process is included in this chapter.

3.1 Engine Cycle Modeling Improvements

When Nadon (1996) automated this conceptual engine optimization process, he used engine cycle and aircraft mission evaluation codes developed by Mattingly (1990). Though these equations adequately model engine cycle calculations for choked low pressure turbine (LPT) conditions, engine performance calculations were not possible for unchoked conditions. Since

the mission profiles modeled often involved wide ranges of flight Mach numbers and altitudes, very low throttle settings were sometimes required to perform mission legs. It was at these flight conditions where Mattingly (1990) codes were unable to provide solutions. The result was that near-optimal engines were rejected, simply because the true optimal engine design had unchoked LPT flow on one or more mission legs.

A significant improvement made in this application was the incorporation of a more robust engine cycle evaluation algorithm. Turbine Engine Reverse Modeling Aid Program (TERMAP) (1997), an on-design and off-design engine evaluation code created for the U.S. Air Force's Wright Laboratories, was integrated with Mattingly's (1987) mission fuel consumption equations. The primary advantage of using TERMAP as the engine cycle code evaluator was that it used compressor and turbine mapping information to produce engine performance results at unchoked engine conditions. Additionally, since this research was sponsored by Wright Laboratories, the end users of this optimization algorithm were more comfortable with the TERMAP engine cycle results (Wright Laboratories has used TERMAP for its engine research for many years). It is worth noting that, for choked engine conditions, engine cycle results were similar regardless of whether TERMAP or Mattingly's (1990) engine codes were used.

3.2 Optimal Fan Pressure Ratio (π_c) Dependency

In this section, we discuss a heuristic approach to choosing the optimal fan pressure ratio (π_c) for a given α , π_c , and W_{TO} . Note that the engine mass flow (m_0) will not be included in this dependency relationship. This is because in conceptual engine design, m_0 simply defines the size of the resulting mixed-stream, turbofan engine; it does not affect the inter-relationships of the various engine cycle components. Said another way, for a given set of rational α , π_c , π_c , and W_{TO}

inputs, any value of m_0 will produce a functional conceptual engine design, though it may not necessarily be capable of performing the mission.

Background. At the heart of the heuristic used to identify the optimal π_c value is a relationship first noticed by Branham (1997). While Branham was working with Mattingly's (1990) on-design conceptual engine design codes (the same codes used by Nadon (1996)), he noticed that, regardless of the reference flight condition selected, for fixed values of α and π_c (and other engine cycle efficiency factors), there always existed a fan pressure ratio at which on-design uninstalled Thrust Specific Fuel Consumption (S) was minimized and on-design uninstalled engine thrust (F) was maximized. Upon closer inspection, he also realized that this always happened when the bypass flow Mach number at the mixer (M_5) was slightly less than the core flow Mach number at the turbine exit (M_5). Although he never explored this relationship any further, Branham was able to generate engine designs that always had near-optimal π_c values.

Perhaps some clarification is appropriate as to why the π_c value that minimizes S and maximizes F is considered optimal. S is calculated using

$$S = \frac{f_0}{F / m_0} \quad (3-1)$$

where

$$f_0 \text{ is the overall engine fuel-to-air ratio} = \frac{\left(\text{total engine fuel flow} \left(\frac{\text{lbm}}{\text{sec}} \right) \right)}{\left(\text{engine inlet air flow} \left(\frac{\text{lbm}}{\text{sec}} \right) \right)}$$

F is the uninstalled engine thrust (lbf)

m_0 is the overall engine inlet air flow $\left(\frac{\text{lbm}}{\text{sec}} \right)$

For each mission leg, an aircraft will require a certain amount of installed thrust (T_{req}) to be delivered by the engine(s) in order to perform the specified leg maneuver. T_{req} will be fixed for a given W_{TO} and m_0 (required thrust is affected by m_0 only if non-constant engine installation losses are being modeled). Since engine cycle models work in terms of uninstalled engine performance, it is necessary to translate installed engine thrust into uninstalled engine thrust required (F_{req}). Thus, for every mission leg, there is an F_{req} that must be generated in order for the engine to meet the installed thrust requirement.

In this application, we are trying to identify the engine that performs a specified aircraft mission while consuming a minimum amount of fuel. Therefore, in meeting the identified F_{req} , it is always desirable to make S as small as possible since this value implies the fuel consumption of the engine. It is important to note that the usefulness of minimizing S would be offset if uninstalled engine thrust was diminished in the process. However, as will be shown, choosing π_c to minimize S has the added benefit of maximizing uninstalled thrust. Not only is the engine most efficient in terms of fuel consumption, but it is also most effective in terms of producing thrust. Both objectives are optimized at the same π_c .

3.3 Understanding How (S) and (F) Are Coincidentally Optimized

We now turn our attention to understanding why coincident optimality of S and F occurs. While understanding this principle for reference engine conditions is foundationally important, the real value of this phenomenon is realized only if S and F remain optimal for off-design flight conditions as well. Branham's discovery is explored for both reference and off-design flight conditions in this section.

Application of Reference Flight Conditions. Unfortunately, looking at ideal (that is, no engine losses or inefficiencies) mixed-stream turbofan engines provides only limited insight into why the phenomenon observed by Branham exists. Figure 3-1 shows the performance of the ideal engine, as generated from the ideal engine equations in Mattingly (1996). Note that with no losses modeled, for a given α , π_c , burner total temperature (T_{t4}) and free-stream Mach number (M_0), larger fan pressure ratios always provide more thrust and better fuel consumption properties, thus implying that an infinitely large fan pressure ratio is optimal. While this is nonsensical in terms of real engine design, one important piece of information can be taken from the ideal engine – at this infinitely large π_c , F is maximized and S is minimized.

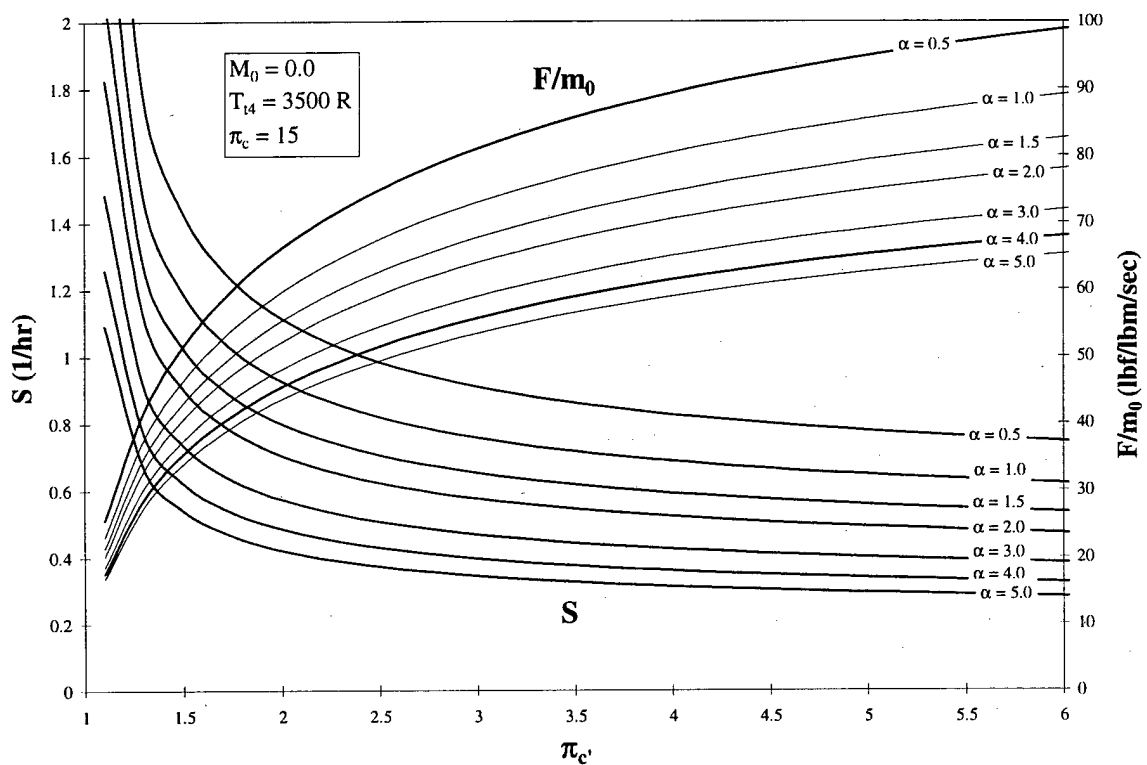


Figure 3-1. On-Design Uninstalled S and Specific Thrust for the Ideal Mixed-Stream Turbofan Engine

We therefore turn our attention to empirical data available from TERMAP on-design engine evaluations that more realistically model engine operation. Imbedded in the TERMAP codes used to perform the on-design engine evaluations are realistic component efficiencies. Figure 3-2 shows the results for the same π_c , burner total temperature (T_{t4}) and free-stream Mach number (M_0) values used for the ideal engine plots in Figure 3-1. Figure 3-3 demonstrates that this phenomenon is not dependent on T_{t4} . Additionally, Figure 3-4 shows the same results with fixed α and varying π_c .

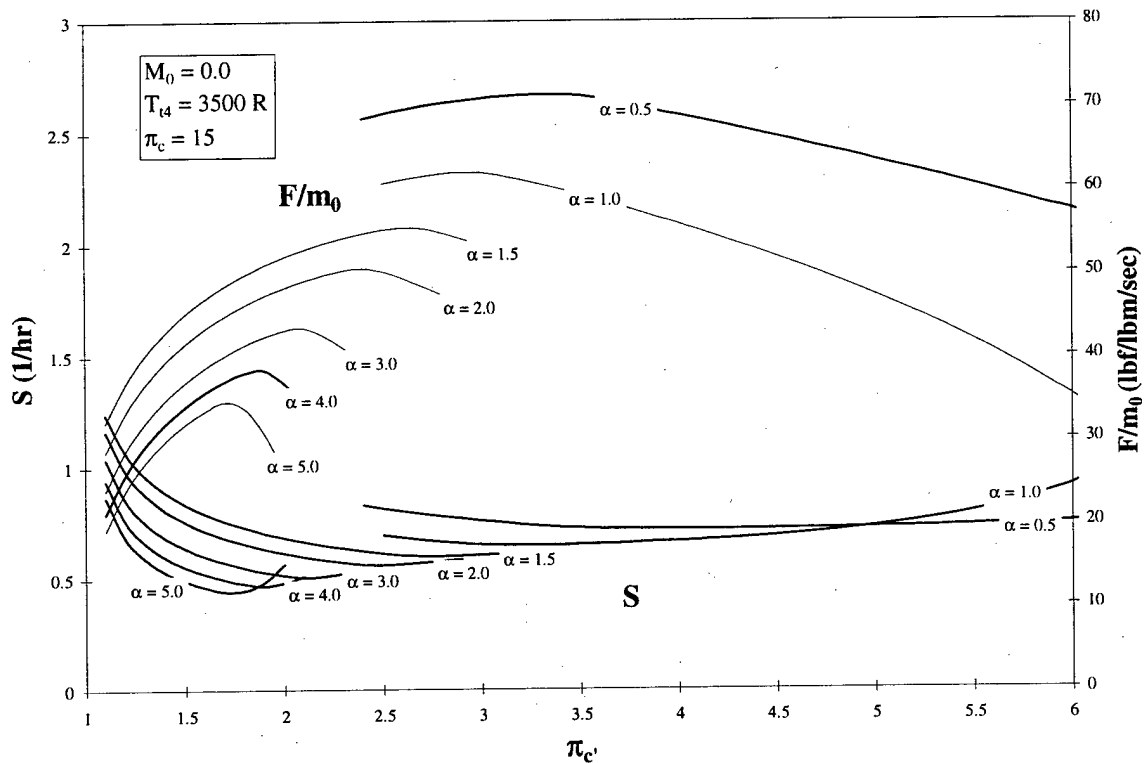


Figure 3-2. On-Design Condition #1: Uninstalled S and Specific Thrust for the TERMAP Generated Mixed-Stream Turbofan Engine at Varying α

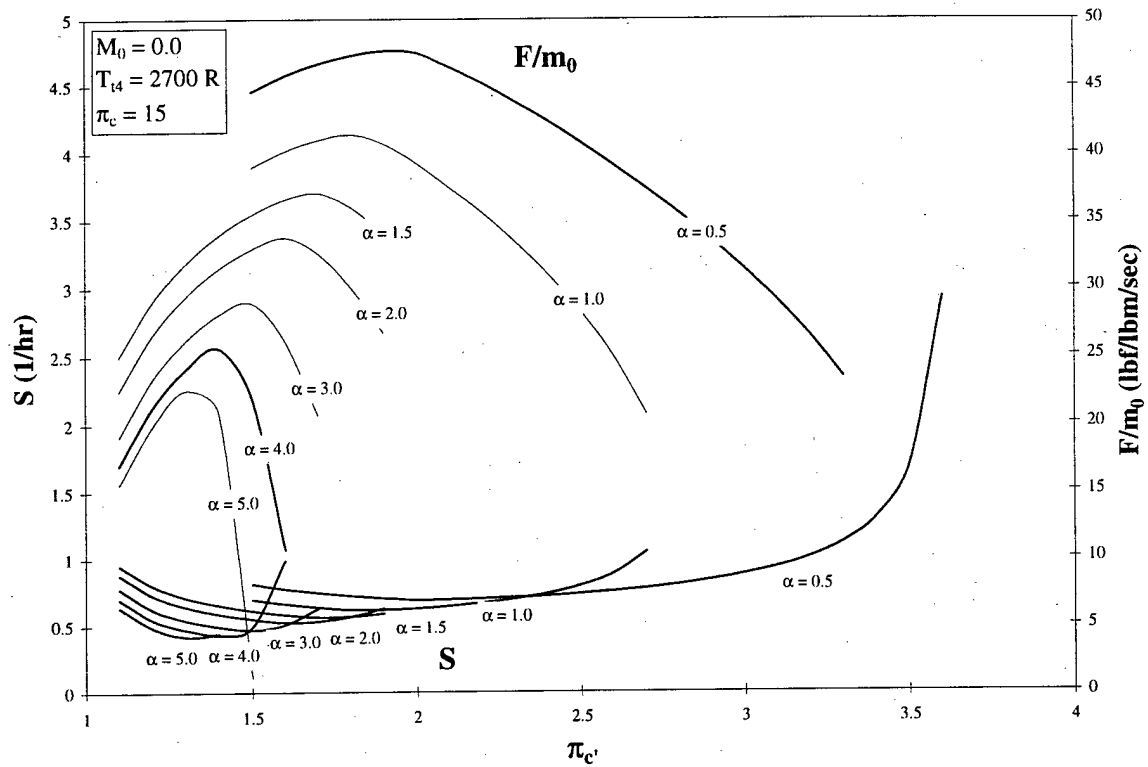


Figure 3-3. On-Design Condition #2: Uninstalled S and Specific Thrust for the TERMAP Generated Mixed-Stream Turbofan Engine at Varying α

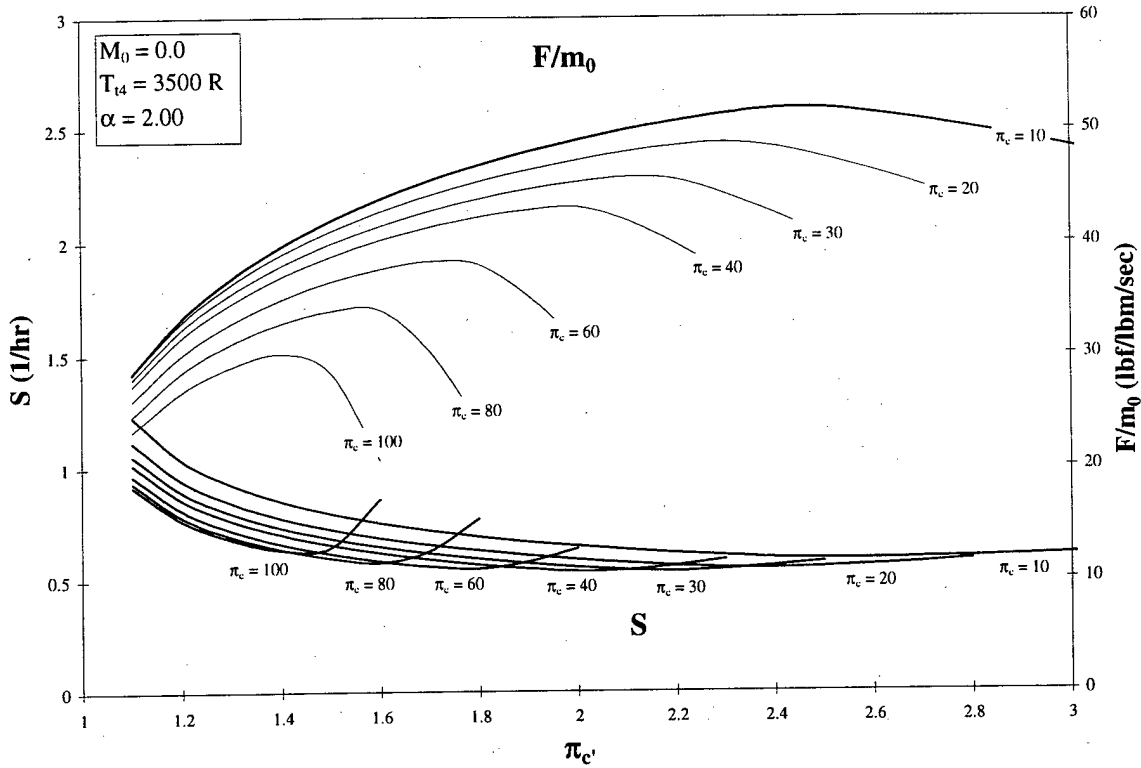


Figure 3-4. On-Design Uninstalled S and Specific Thrust for the TERMAP Generated Mixed-Stream Turbofan Engine at Varying π_c

Notice that with engine inefficiencies modeled, S and F reach optimal values at reasonable fan pressure ratios. Also note that, regardless of the value of T_{t4} , α , or π_c , the fan pressure ratio at which S is minimized is also the fan pressure ratio for which F is approximately maximized. Although these figures were generated using TERMAP, this phenomenon is also observed using Mattingly's (1990) engine codes.

Branham first used the $\frac{M_{5'}}{M_5}$ ratio to identify the π_c value at which this dual optimality is achieved. Analogous to the $\frac{M_{5'}}{M_5}$ ratio is the ratio of the bypass and core flow total pressures at mixer entry $\left(\frac{P_{t5'}}{P_{t5}}\right)$, which is readily available from TERMAP output. When the $\frac{P_{t5'}}{P_{t5}}$ ratio is

plotted against S and F (all of which are engine cycle outputs), approximately optimal S and F are obtained at a fixed $\frac{P_{t5'}}{P_{t5}}$ ratio, regardless of the values of α , π_c , T_{t4} , and M_0 . Figure 3-5 shows that the $\frac{P_{t5'}}{P_{t5}}$ ratio at which this happens is approximately 1.00 for a fixed π_c and varying α .

Figure 3-6 shows this same characteristic to be true with a fixed α and varying π_c .

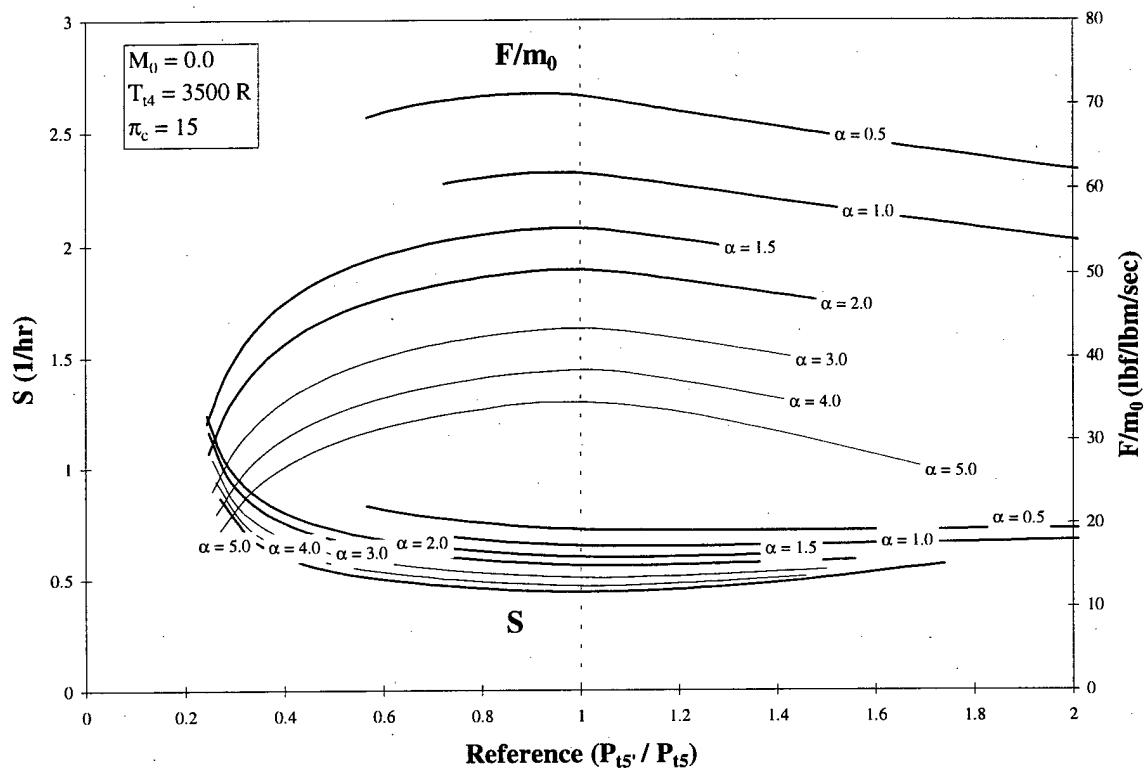


Figure 3-5. Optimal On-Design Uninstalled S and Specific Thrust at $\frac{P_{t5'}}{P_{t5}} \approx 1.00$ for Varying α

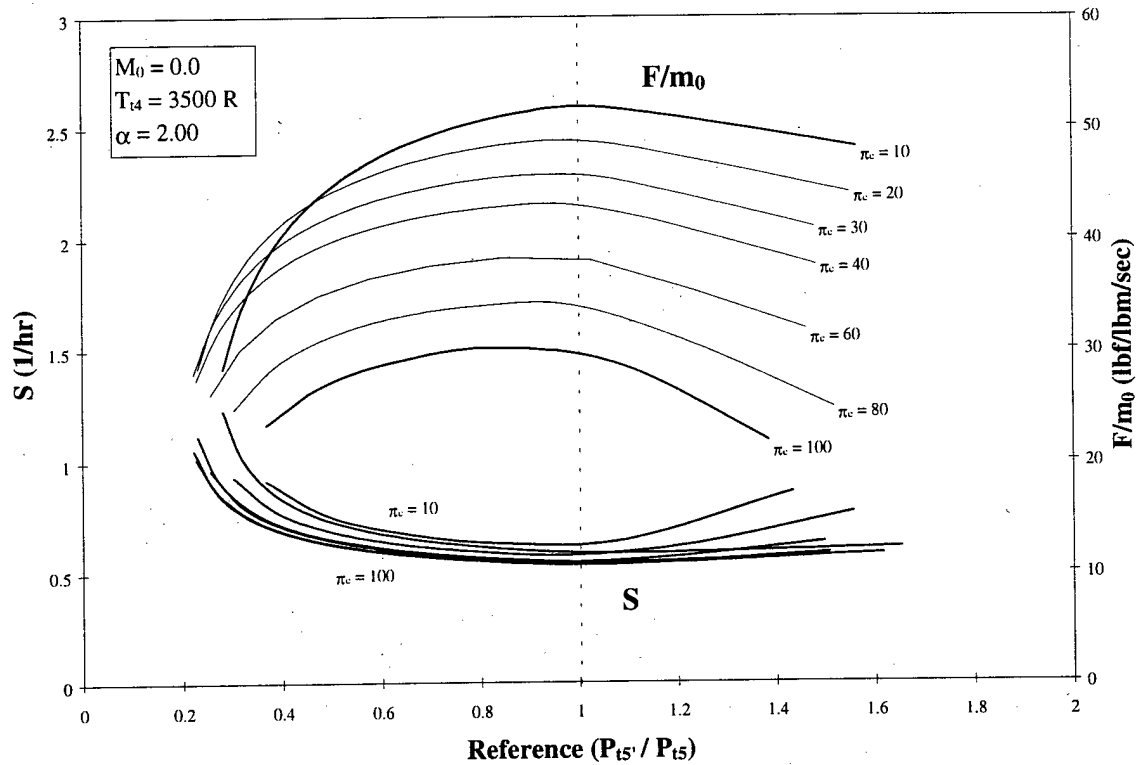


Figure 3-6. Optimal On-Design Uninstalled S and Specific Thrust at $\frac{P_{t5'}}{P_{t5}} \approx 1.00$ for Varying π_c

An important connection to be made here is that the $\frac{P_{t5'}}{P_{t5}}$ ratio is a function of the fan pressure

ratio combined with constant α and π_c values. Figure 3-7 shows the relation between $\frac{P_{t5'}}{P_{t5}}$ ratio

and π_c . Notice that the $\frac{P_{t5'}}{P_{t5}}$ function for each bypass ratio is monotonically increasing (a

property that will be exploited in locating the π_c at which the optimal $\frac{P_{t5'}}{P_{t5}}$ is obtained).

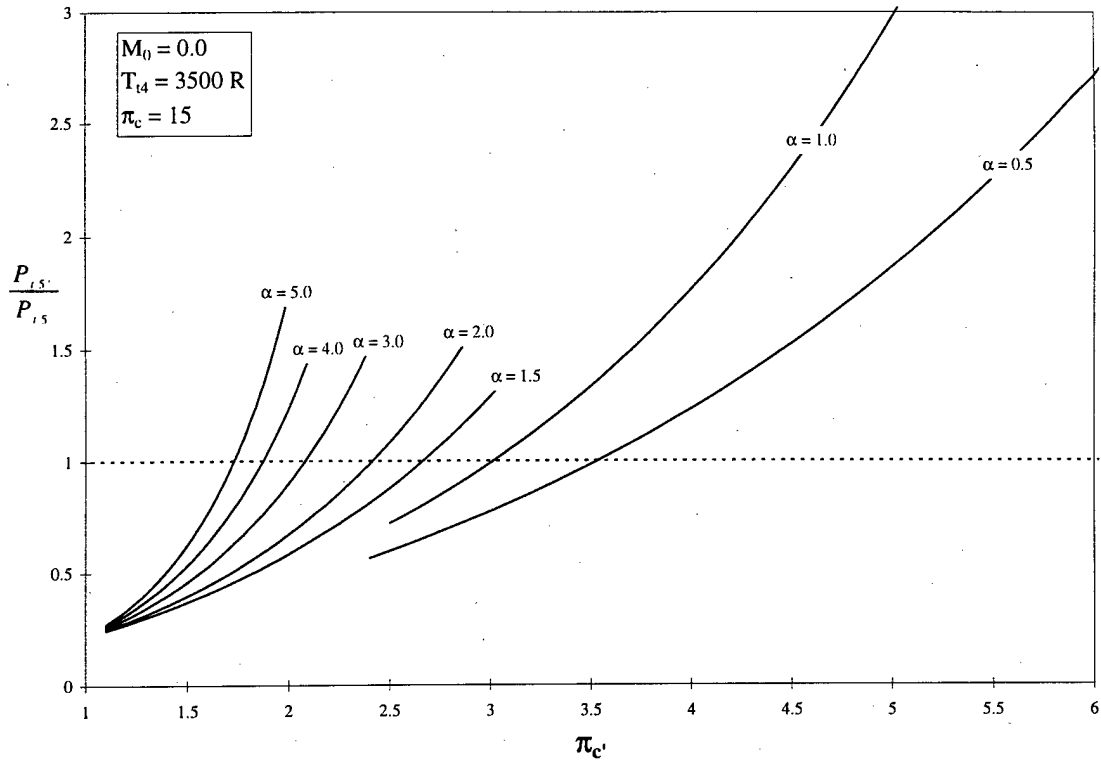


Figure 3-7. $\frac{P_{t5'}}{P_{t5}}$ Variation with Changing $\pi_{c'}$ at Various α

Thus, for the on-design engine, it is possible to vary $\pi_{c'}$ until the $\frac{P_{t5'}}{P_{t5}}$ ratio reaches its optimal value – when this $\frac{P_{t5'}}{P_{t5}}$ ratio is obtained, the best $\pi_{c'}$ has been (approximately) located. This $\pi_{c'}$ is considered best because it maximizes unstalled thrust output while minimizing engine fuel consumption. This minimized fuel consumption leads to a minimized amount of fuel required to meet T_{req} for the specified duration of a mission leg.

Application to Off-Design Flight Conditions. The on-design engine property noted by Branham would not be useful in this application if it applied exclusively to on-design engines. Regardless of the reference conditions chosen for an on-design engine, an aircraft flies some portion (if not all) of its mission at off-design flight conditions. Hence, if the selection of an

optimal π_c value only minimized S and maximized F on-design, this property would be of very limited use. However, it was found that by selecting an optimal π_c value for the reference engine, S and F were also approximately optimized for all off-design flight conditions tested in this research.

Figure 3-8 through Figure 3-10 show off-design S and F values plotted against reference $\frac{P_{t5'}}{P_{t5}}$ ratios for three arbitrary off-design conditions and throttle settings.

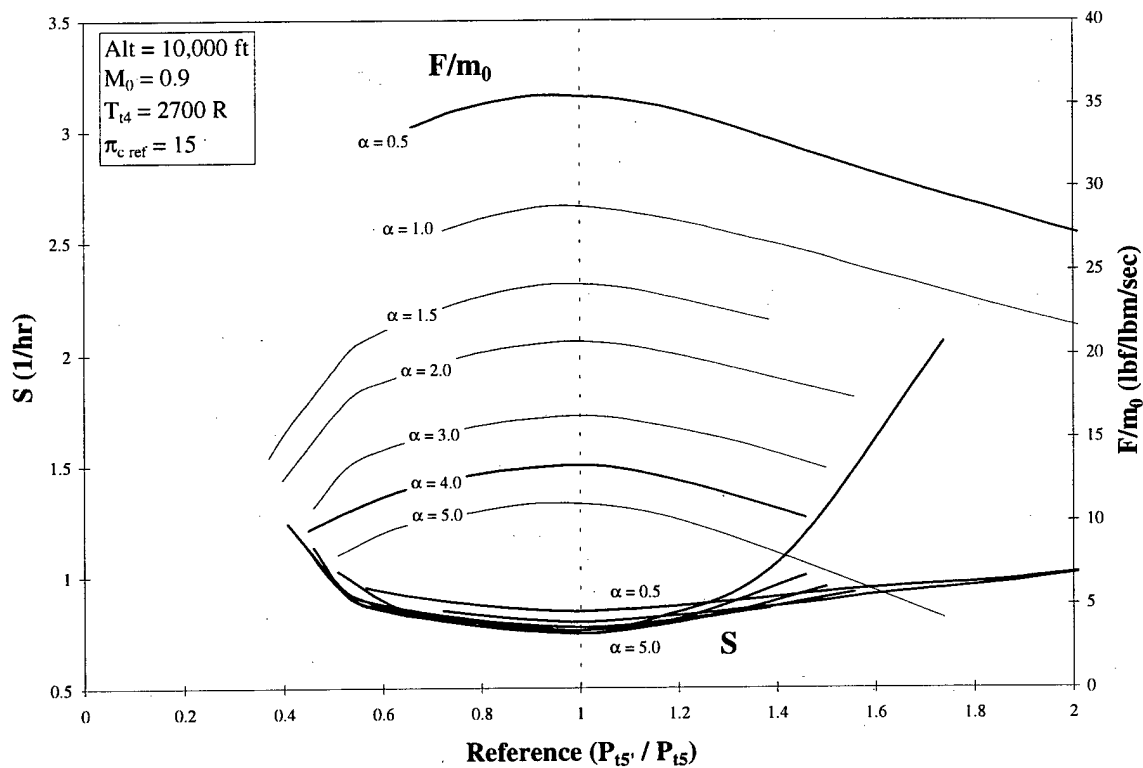


Figure 3-8. Off-Design Flight Condition #1: Optimal Off-Design Uninstalled S and Specific Thrust at Reference Engine $\frac{P_{t5'}}{P_{t5}} \approx 1.00$

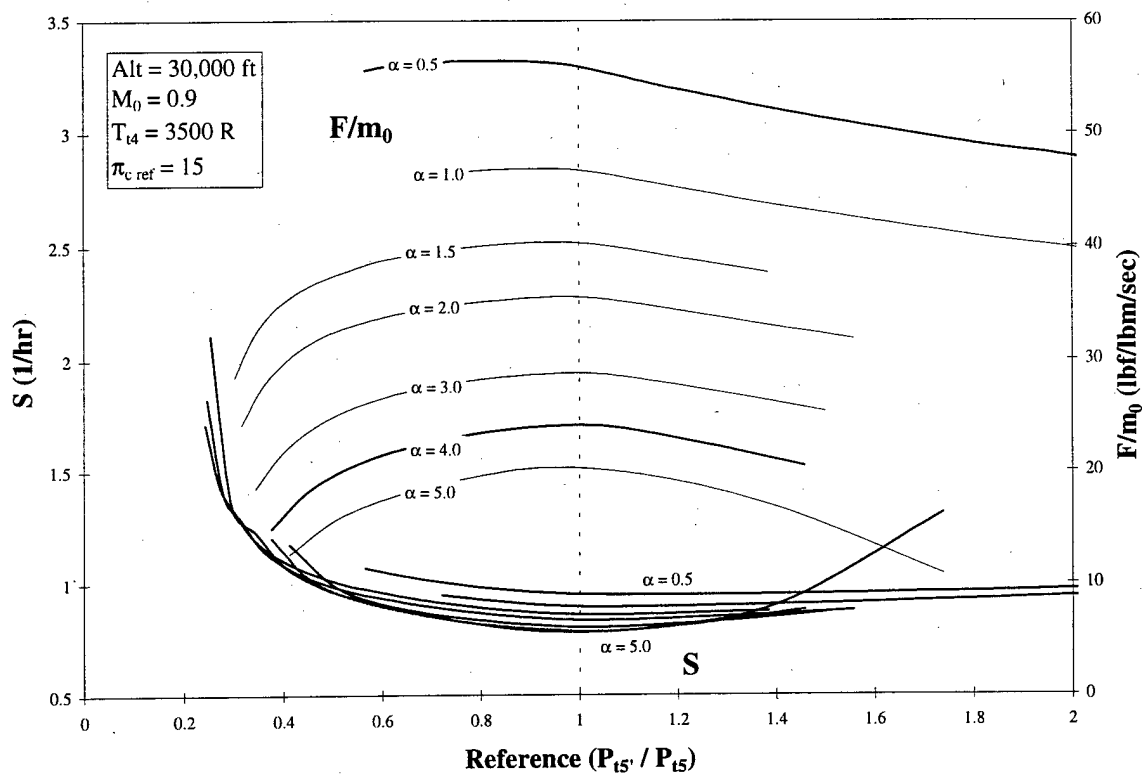


Figure 3-9. Off-Design Flight Condition #2: Optimal Off-Design Uninstalled S and Specific Thrust at Reference Engine $\frac{P_{t5'}}{P_{t5}} \approx 1.00$

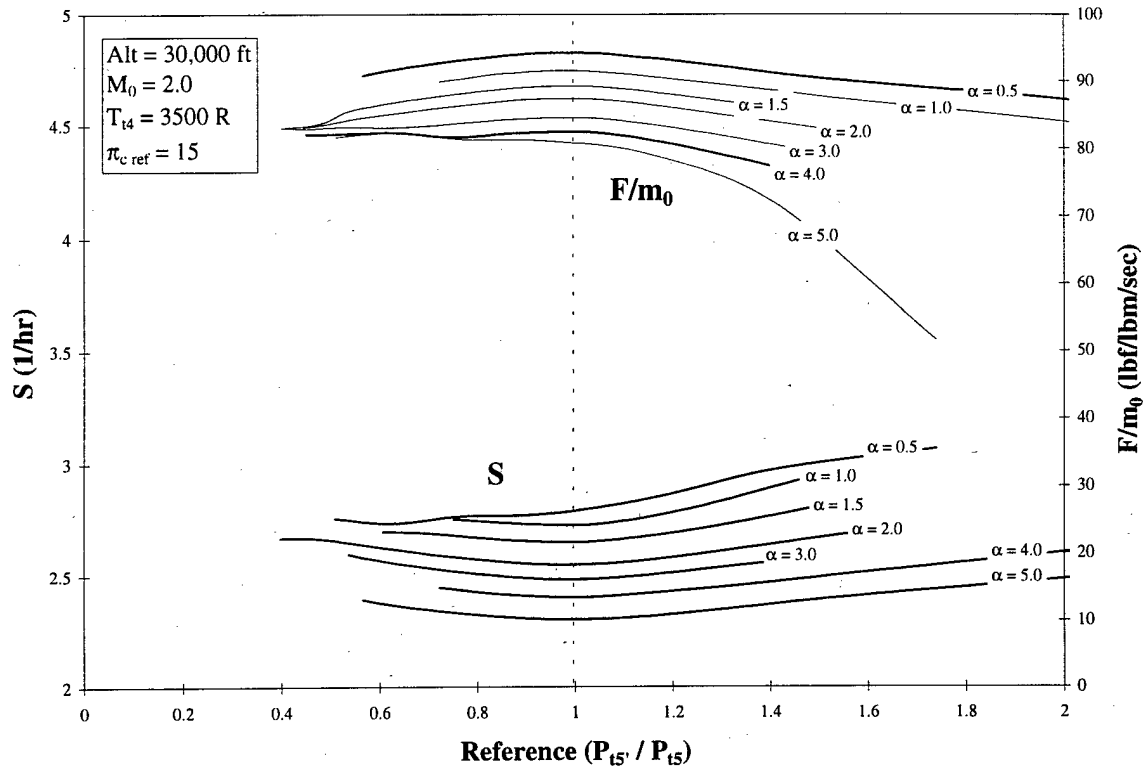


Figure 3-10. Off-Design Flight Condition #3: Optimal Off-Design Uninstalled S and Specific Thrust at Reference Engine $\frac{P_{t5^*}}{P_{t5}} \approx 1.00$

It is seen once again that regardless of the reference bypass ratio, off-design S is always approximately minimized and off-design F is always approximately maximized at the same reference $\frac{P_{t5^*}}{P_{t5}}$ ratio. While an exhaustive list of figures showing many different off-design conditions is not included, this π_c optimality principle held true for all of the various flight conditions used in this application.

Understanding Off-Design Optimality Using Reference Engine Optimization. It is important to understand why S and F optimality is maintained once a departure from reference engine flight conditions has occurred. Similar to operation at reference conditions, the off-design fan pressure ratio that minimizes S and maximizes F is expected to be equal to the value that

causes the off-design $\frac{P_{t5'}}{P_{t5}}$ ratio to be approximately 1.00. Once reference engine conditions are departed, the off-design π_c , π_c and α can be significantly different from the reference π_c , π_c and α (Mattingly, et al., 1987, 120-122). This would seem to imply that the resulting off-design $\frac{P_{t5'}}{P_{t5}}$ ratio is likely to be very different as well.

However, in reality the $\frac{P_{t5'}}{P_{t5}}$ ratio does not change significantly with changing flight conditions when it is first optimized for the reference engine. This is attributable to the fact that, for most flight conditions, the engine spool RPM typically remains at 80 - 100% of the full-throttle reference engine RPM. Over this range of spool RPMs, M_5 and M_5 do not vary significantly from their full-throttle reference engine values (Mattingly, 1996, 558). Since $P_{t5'}$ and P_{t5} are derived directly from these Mach numbers (via the isentropic compressible flow equations), it follows that the $\frac{P_{t5'}}{P_{t5}}$ ratio also remains relatively constant as well. This notion is confirmed in Figure 3-11 through Figure 3-13 where reference and off-design $\frac{P_{t5'}}{P_{t5}}$ ratios from TERMAP are compared.

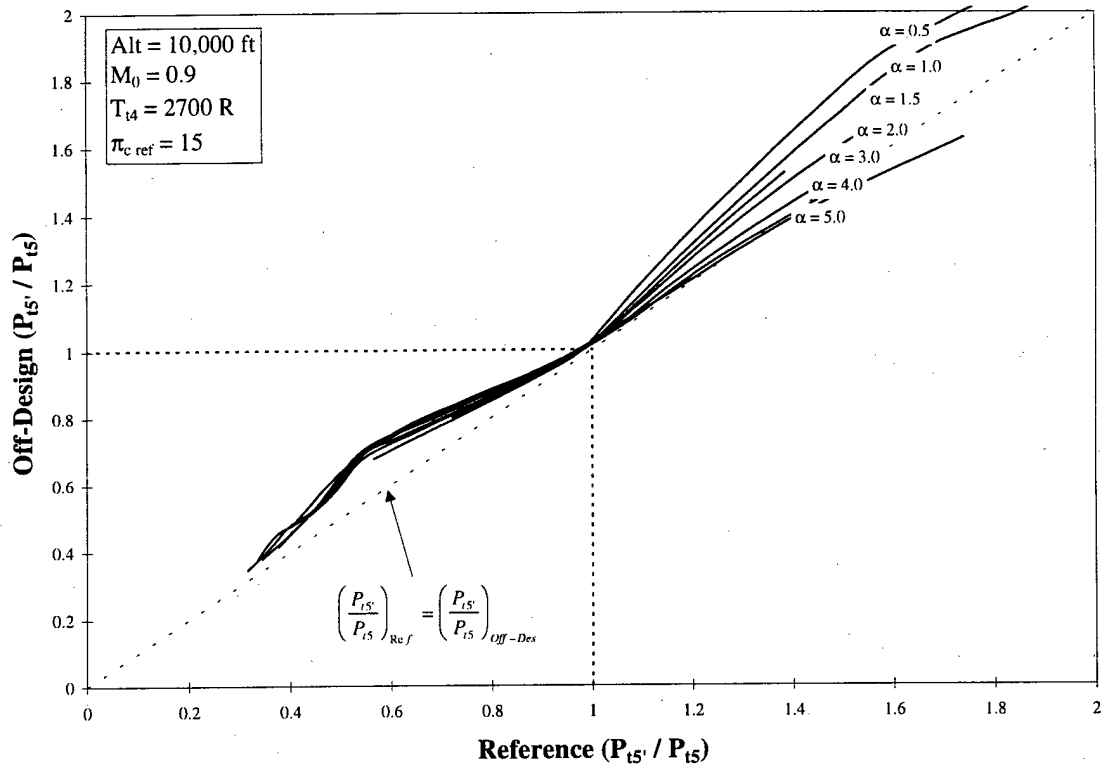


Figure 3-11. Variation of Off-Design $\frac{P_{t5'}}{P_{t5}}$ Ratio with Changes in Chosen Values of On-Design $\frac{P_{t5'}}{P_{t5}}$ Ratio for Off-Design Condition #1

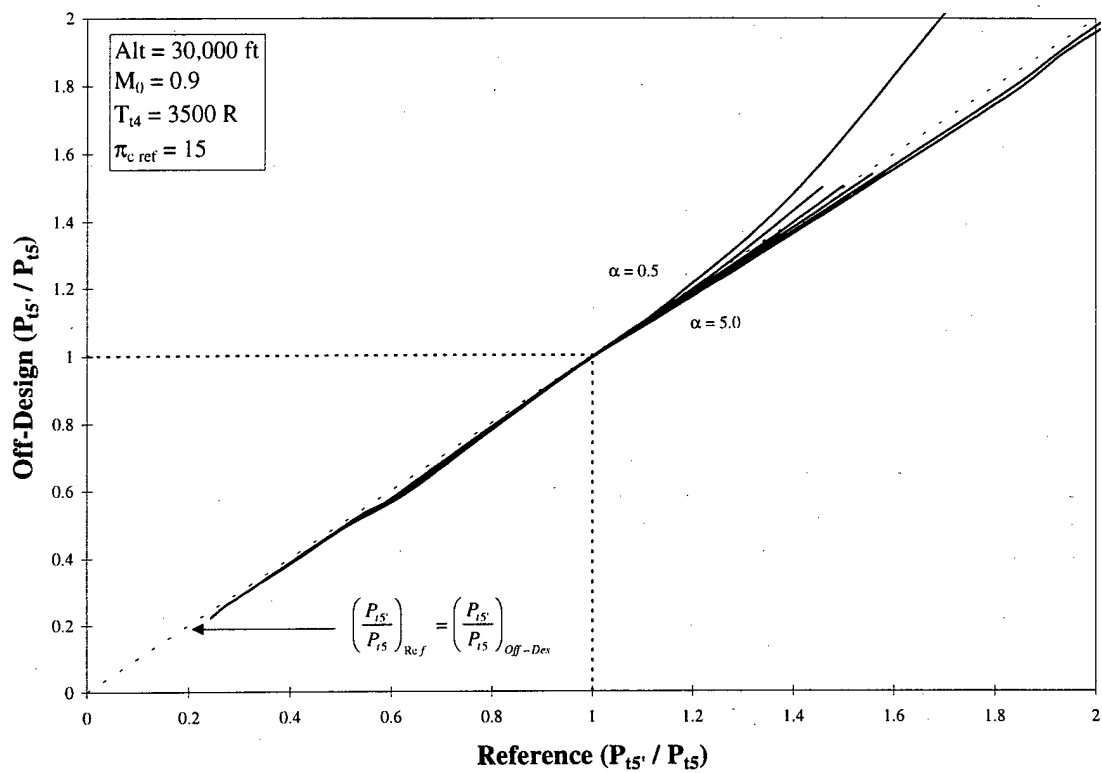


Figure 3-12. Variation of Off-Design $\frac{P_{t5'}}{P_{t5}}$ Ratio with Changes in Chosen

Values of On-Design $\frac{P_{t5'}}{P_{t5}}$ Ratio for Off-Design Condition #2

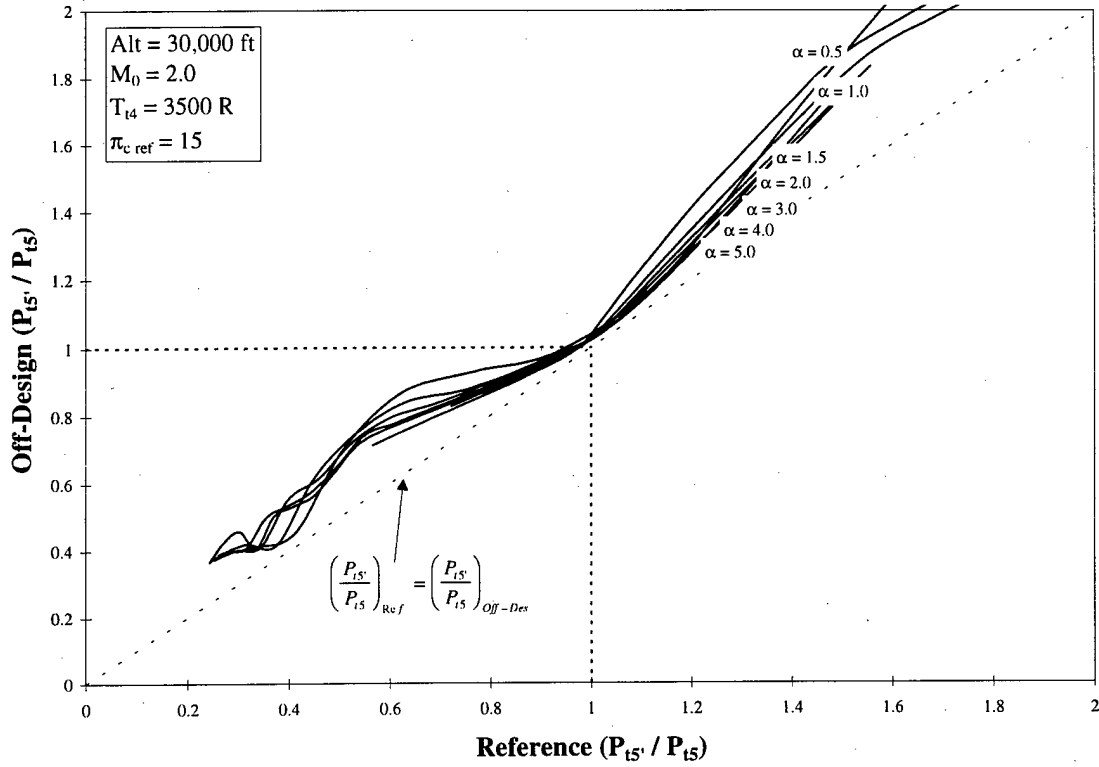


Figure 3-13. Variation of Off-Design $\frac{P_{t5'}}{P_{t5}}$ Ratio with Changes in Chosen Values of On-Design $\frac{P_{t5'}}{P_{t5}}$ Ratio for Off-Design Condition #3

In the previous three figures, we see that, although there may be significant deviations from quality of $\left(\frac{P_{t5'}}{P_{t5}}\right)_{Ref}$ and $\left(\frac{P_{t5'}}{P_{t5}}\right)_{Off-Des}$ across the spectrum of chosen $\left(\frac{P_{t5'}}{P_{t5}}\right)_{Ref}$ values, at

$$\left(\frac{P_{t5'}}{P_{t5}}\right)_{Ref} \approx 1.00, \text{ the curves converge so that } \left(\frac{P_{t5'}}{P_{t5}}\right)_{Ref} \approx \left(\frac{P_{t5'}}{P_{t5}}\right)_{Off-Des}.$$

Application to Jet Engine Optimization for an Aircraft Mission. In Nadon's (1996) work on the jet engine optimization problem for a specified mission, fan pressure ratio was treated as an independent variable. Though this is a viable approach to this optimization, practice has shown that for a given α and π_c , the range of working π_c values for the on-design engine can be

small (reference Figure 3-2 through Figure 3-4, which plot all working values of π_c for the given α and π_c). The range of π_c values that provides a realizable engine across a wide range of off-design flight conditions is even smaller. Additionally, this range of working π_c values shrinks significantly as bypass ratio increases. Since this non-linear optimization problem is typically solved using a probabilistic optimization algorithm (such as the genetic algorithm), it is left to chance that a proper π_c value be selected to complement the selected α and π_c values. Since only a small range of π_c values produce viable engine designs, it is easy for the optimizer to miss these possible values. Hence, many α and π_c combinations would never be evaluated because an appropriate π_c values would not be chosen by the optimizer. This is especially true for the high bypass ratio engines that have extremely narrow ranges of valid π_c values. The net result is that, without very exhaustive sampling, optimal solutions tend to have low bypass ratios where the range of functional π_c values is relatively wide. High bypass ratio engines are never fully explored.

As an alternate approach, the property noted by Branham can be employed to make π_c a dependent variable, determined by the selected α and π_c values. The benefit of this approach is that locating the optimal π_c value is not left to an optimizer that may or may not ever locate its best value. Because of the narrow range of π_c values that produce viable engines for a given α and π_c , using this approach ensures that, not only is a working π_c value found at every engine evaluation, but also the identified π_c is the approximately its best possible value. By implementing a short algorithm to locate the optimal π_c , this variable is effectively removed from the set of independent design variables controlled by the optimizing algorithm. As previously mentioned, this dimension reduction encourages faster optimizer convergence to the global optimum.

In sections 3.2 and 3.3, it was shown that a relationship exists between the optimal $\frac{P_{ts}}{P_{ts}}$ ratio, S and F . It was also shown that, regardless of the α or π_c values selected, the optimal $\frac{P_{ts}}{P_{ts}}$ ratio remained approximately the same. Some iteration was required to discern the best $\frac{P_{ts}}{P_{ts}}$ ratio to achieve π_c optimality (since there was some variation based on the mission profile), but it was finally observed that $0.99 \leq \frac{P_{ts}}{P_{ts}} \leq 1.01$ produced the best results. This target $\frac{P_{ts}}{P_{ts}}$ ratio is presented as a range of values because the precise best $\frac{P_{ts}}{P_{ts}}$ ratio tended to fluctuate slightly depending on the mission off-design conditions and the selected values of α and π_c . This is consistent with the plots shown in section 3.3 that reveal slight discrepancies in the $\frac{P_{ts}}{P_{ts}}$ ratio (depending on flight condition, α and π_c) at which S is minimized and F is maximized.

The goal here is to locate the π_c value that produces the optimal $\frac{P_{ts}}{P_{ts}}$ ratio. To do this, a simple Newton-Raphson convergence algorithm, supplemented by a bisection convergence algorithm, was employed. A description of the Newton-Raphson and bisection root-finding algorithms can be found in Burden et al. (1993). Both of these root-finding algorithms are capable of locating the value of a single input variable that creates a function response of zero. In this case, the single input variable was π_c and the response was $\left(\frac{P_{ts}}{P_{ts}} - 1.00 \right)$. Figure 3-7 shows that $\frac{P_{ts}}{P_{ts}}$ ratio, as a function of π_c , is a monotonically increasing function. By defining

the function used by the root-finding algorithms as $\left(\frac{P_{t5'}}{P_{t5}} - 1.00 \right)$, the algorithms are able to

locate where $\frac{P_{t5'}}{P_{t5}} = 1.00$.

The Newton-Raphson root-finding method uses gradient estimates to quickly converge to the root (or zero) of a function. However, it is possible, depending on the function's slope at the point being evaluated, for this root-finding method to attempt to move to invalid π_c values ($\pi_c \leq 1.00$). Such moves produce invalid $\frac{P_{t5'}}{P_{t5}}$ results (as well as other engine cycle evaluation outputs), thus confounding the success of this root-finding technique. For this reason, the bisection root-finding method is needed to back-up the Newton-Raphson method. Although the bisection method is typically much slower to converge to the root of a function, it is much more robust than Newton-Raphson. Therefore, if invalid values of $\frac{P_{t5'}}{P_{t5}}$ are experienced while attempting to use Newton-Raphson, this root-finding algorithm is aborted and the bisection root-finding method is utilized.

Fan Pressure Ratio Optimization Conclusions. In all, excellent results were obtained using the fan pressure ratio optimization techniques discussed above. The combination of Newton-Raphson and bisection root-finding methods had a 100% success rate locating the desired $\frac{P_{t5'}}{P_{t5}}$ ratio, regardless of the α and π_c values being evaluated. As might be expected, selecting the target $\frac{P_{t5'}}{P_{t5}}$ ratio required special attention. When the target $\frac{P_{t5'}}{P_{t5}}$ ratio was set to 1.00, π_c values for optimized engine solutions were likely to be optimal (e.g. changing π_c in either direction resulted in increased fuel consumption), but were not guaranteed to be optimal.

By slightly varying the target $\frac{P_{t5}}{P_{t5}}$ ratio above or below 1.00, the likelihood of the chosen π_c value being optimal decreased. It was therefore concluded that the precisely best $\frac{P_{t5}}{P_{t5}}$ ratio varies slightly based on other engine and mission parameters and that choosing a target value of 1.00 seemed to provide the highest likelihood of locating the optimal π_c . Despite the shortcomings of using this criteria, use of the fan pressure ratio optimization algorithms proved to be quick and highly effective at obtaining nearly optimal π_c values for any valid pair of α and π_c values. Convergence to the best π_c typically only took 5-10 seconds for each mission evaluation required by the optimization algorithm.

The most significant benefit of using this algorithm was that high bypass ratio engines could be properly considered by the optimizer. When π_c was previously treated as an independent variable, optimal engine solutions tended to have much lower (and less fuel efficient) bypass ratios. With fan pressure ratio converted into a dependent variable (as a function of α and π_c), functioning high bypass ratio engines could be located by the genetic algorithm as easily as low bypass ratio engines. This resulted in all valid combinations of α and π_c being considered in the optimization process.

3.4 Determining Optimal Reference Engine Mass Flow (m_0)

At this point, four of the five variables required to evaluate the mission for fuel consumption have been identified. Recall that α , π_c and W_{TO} were determined by the optimizer algorithm processes, and that π_c was selected as discussed in section 3.2. We are therefore left with sizing the engine to produce the required thrust. Similar to the fan pressure ratio, Nadon (1996) simply allowed m_0 to be an independent variable selected and optimized by the optimizer

algorithm. However, there is a more direct method by which the best m_0 can be selected for a given α , π_c , and π_e .

It is intuitive that it would never be desirable to have an engine that is over-sized for an aircraft's mission. Large engines weigh more than small engines and therefore require more aircraft structure to support the engine and its related components. Additionally, the magnitude of the engine installation losses is directly proportional to the size of the engine (when they are not assumed to be constant). To suffer either of these penalties when an optimal, most efficient engine is desired, is counter-productive. It is, therefore, advisable to make the engine just large enough to meet the aircraft's most thrust-demanding leg. We now turn our discussion to locating the optimal engine mass flow assuming constant engine installation losses.

Iterating to the Best Reference Engine Mass Flow. In order to perform this iterative calculation, it is first necessary to realize that we are iterating to a desired off-design thrust by modifying the mass flow of the reference engine. In order to do this, we must first determine the ratio of off-design m_0 to on-design m_0 . When sizing an engine at some arbitrary off-design flight condition, the ratio of the off-design required engine inlet area (A_1) to the on-design free-stream area (A_{0ref}) is constant ($A_1 / A_{0ref} = \text{constant}$, regardless of the engine size) (Mattingly, et al., 1987, 194). Assuming that the off-design inlet Mach number and the reference free-stream Mach number are kept constant, it follows that the ratio of the off-design and reference mass flows, which are directly related to the areas required by the off-design and reference conditions, is also constant $\left(\frac{m_{0off}}{m_{0ref}} = \text{constant} \right)$. With this relationship, it is now possible to calculate the change in reference engine mass flow to obtain the required off-design engine thrust.

We will be using uninstalled thrust throughout the following calculations. Uninstalled thrust is used because TERMAP (like most engine cycle codes) only generates uninstalled engine

performance. Since in mission optimization we work in terms of installed thrust (the actual thrust delivered to the aircraft) and in engine mass flow optimization we work in terms of uninstalled thrust, it is imperative that the proper uninstalled required thrust has been determined before any of the following calculations are performed.

First, we calculate the deficit in required uninstalled, off-design thrust.

$$F_{off_{def}} = F_{off_{req}} - F_{off_{avail}} \quad (3-2)$$

Using the thrust deficit and the off-design specific thrust $\left(\frac{F_{off}}{m_{0_{off}}} \right)$ of the engine (output from the off-design engine cycle analysis), it is possible to calculate the off-design engine mass flow deficit.

$$m_{0_{off_{def}}} = \frac{F_{off_{def}}}{\left(\frac{F_{off}}{m_{0_{off}}} \right)} = \frac{\text{Off - Design Thrust Deficit}}{\text{Off - Design Specific Thrust}} \quad (3-3)$$

This deficit off-design mass flow can then be translated to a deficit reference engine mass flow using the off-design / reference engine mass flow ratio (again, obtained from the engine cycle analysis).

$$m_{0_{ref_{def}}} = \frac{m_{0_{off_{def}}}}{\left(\frac{m_{0_{off}}}{m_{0_{ref}}} \right)} \quad (3-4)$$

It is now possible to adjust the reference engine mass flow (by $m_{0refdef}$) to obtain the desired off-design thrust.

It is important to note that convergence to the proper m_{0ref} is an iterative process, requiring multiple engine cycle analyses and execution of the above calculations. Another important point is that, during convergence to the proper m_{0ref} , it is possible to overshoot the optimal m_{0ref} . That is, the mass flow becomes larger than is actually required to meet the thrust requirement. For this reason, the engine may need to be down-sized during the iterative process. Regardless of whether the mass flow is too small or too large, the same calculations are used (only the sign of $m_{0refdef}$ changes). Convergence is typically achieved when

$$F_{off_{req}} \leq F_{off_{avail}} \leq (F_{off_{req}} + Tolerance),$$

where *Tolerance* equals some small amount of thrust

determined to be inconsequential to the overall solution. Finally, it is worth re-stating that this algorithm assumes constant engine installation losses. Were installation losses to be more realistically modeled, $F_{off_{req}}$ would be affected not only by the aircraft weight, but also by the variable aerodynamic drag forces on the changing engine and its housing ($F_{off_{req}}$ increases for increasing m_0 values, decreases for decreasing m_0 values). The complexity involved in incorporating non-constant loss models is beyond the scope of this study.

At this point, because the engine has been re-sized (e.g. the reference m_0 has been modified), the mission evaluation process must be re-started from the beginning. This is because all of the fuel consumption properties of the previous legs will have been affected by the change in engine size. To continue mission evaluation without starting over would propagate inaccurate aircraft weight estimates for each of the subsequent legs. Recall that the aircraft is getting lighter as the mission proceeds due to fuel consumption. Since aircraft weight at the beginning of a leg affects the drag force that will be experienced (and consequently the thrust required and fuel

consumption) during the leg, accurate estimates of each leg's beginning aircraft weight are essential.

This entire process is depicted in flow chart format in Figure 3-14.

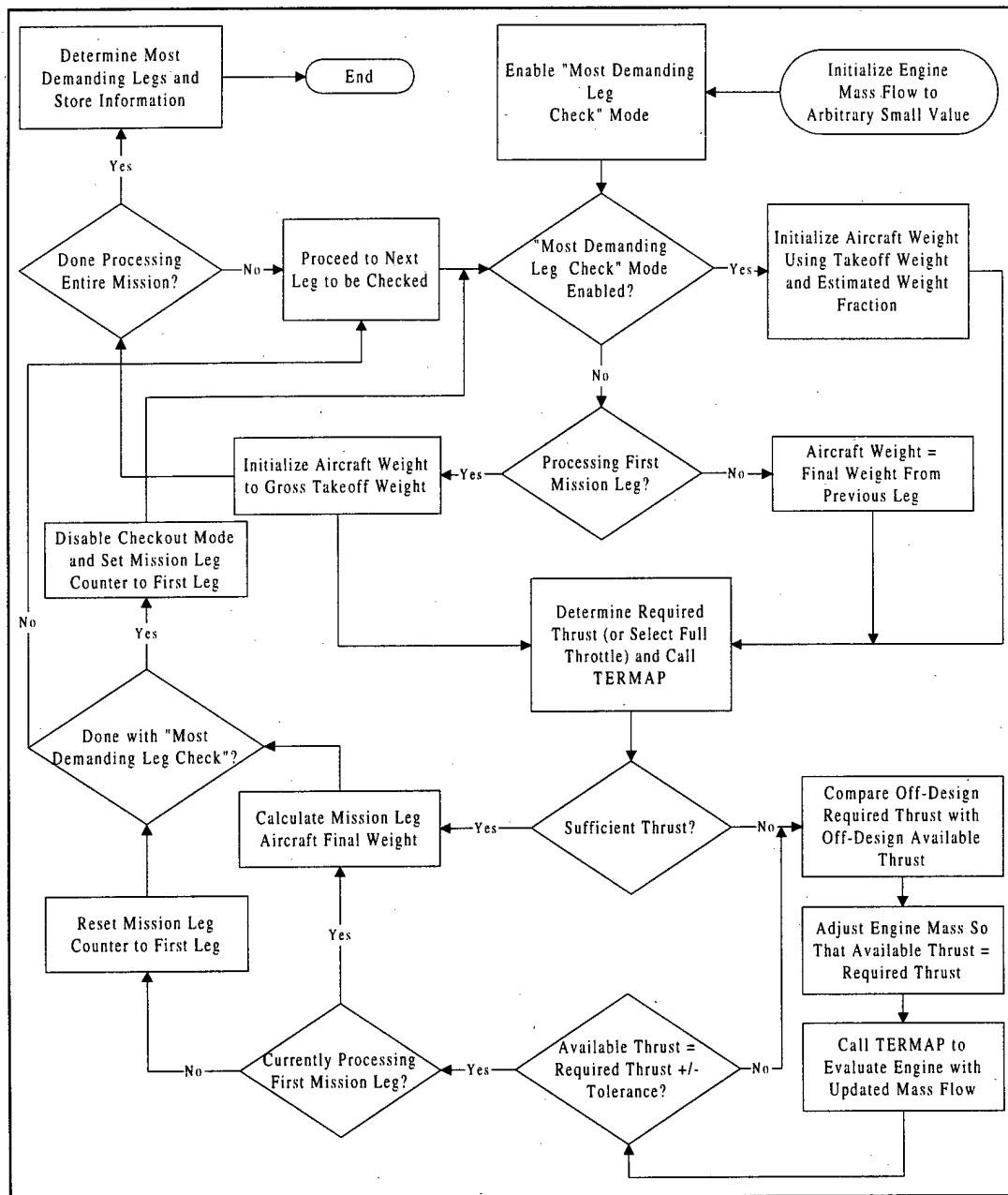


Figure 3-14. Process Flow for Determining Optimal Engine Mass Flow with Constant Installation Losses

Time-Saving Techniques. As a time saving addition, the algorithm depicted in Figure 3-14 attempts to guess which mission leg(s) have forced m_0 to its current optimal value. At the end of the engine sizing process, before the algorithm is exited, the most demanding mission legs are determined and stored. In order to do this, we simply compare the off-design F_{req} and F_{avail} for each mission leg. When F_{avail} is equal to or slightly larger than F_{req} , the engine is having to produce maximum thrust to meet the mission leg's thrust requirements. It follows that this leg is responsible (either solely or in part) for the optimal reference m_0 value being as large as it is. It is likely that, when this leg was encountered during mission processing, the engine had to be re-sized so that it could produce the thrust necessary to perform the leg's required maneuver. Each leg that has equal or nearly equal F_{req} and F_{avail} values are considered to be among the most demanding legs.

Every time the m_0 optimizing algorithm is called, it first evaluates and sizes the engine to the leg(s) that have been most demanding in previous m_0 optimizations. This is done to prevent as many iterations as possible in locating the optimal m_0 . Recall that as the algorithm is stepping through the mission legs, if the engine is re-sized, the evaluation of the mission must be re-started at the first leg. Hence, in the worst case, if the mission legs become more demanding with each successive leg (causing the engine to have to be re-sized), the mission would have to be re-started numerous times to obtain the optimal m_0 . However, if the most demanding leg can be identified and evaluated first, then the engine will be large enough for all other legs, thus preventing mission re-starts.

In implementing this short-cut technique, it was first noticed that, depending on the engine design (α , π_c , π_c and W_{TO}), the most demanding leg of a mission could change. Therefore, it was necessary to determine the frequency of each mission leg being among the k most demanding (where $1 \leq k \leq \text{Total Number of Mission Legs}$). Along with this frequency data,

running averages of the aircraft weight fractions (β , where

$$\beta = \frac{\text{Aircraft Weight at Start of Mission Leg}}{W_{TO}} \text{) experienced in previous } m_0 \text{ optimizations were}$$

also stored. Then, when the m_0 optimization algorithm was initiated for a new engine design, the k legs with the highest occurrences of being among the most demanding legs were checked first. The stored β values were used to estimate the aircraft weight at these most demanding legs (recall that required thrust is affected by the mission leg aircraft weight). Obviously, choosing values of k too large defeats the efficiency of the short-cut technique. However, choosing k too small also has adverse effects because the most demanding leg for the new engine may not be included in the set being tested. In this application, $k = 2$ (for a mission with 15 - 25 mission legs) provided a good balance of efficiency and thoroughness.

Application to Jet Engine Optimization for an Aircraft Mission. Unfortunately, implementation of the engine mass flow optimization algorithm had mixed results. While it did successfully complement the jet engine optimization algorithms, it tended to increase the time required for each objective function evaluation by 200-300%. On average, an objective function call for a mission with 20 legs required 60-80 seconds to evaluate with the mass flow optimizer enabled. Without it, objective function evaluations took 20-35 seconds. Some of this time penalty was offset by the reduced number of objective function calls required for convergence. However, it is unlikely that the reduced number of function calls offset the increased processing time per function call.

Perhaps an increase in processing would be justifiable if the fuel efficiency of the optimal engine solutions were better than solutions obtained without the mass flow optimizer. However, initial indications revealed that the m_0 values for converged engine solutions were not optimal. By reducing the mass flow below the value located by the m_0 optimizer, significantly

more efficient engine solutions were produced. This indicated that the algorithm was not locating the actual minimum m_0 for the α , π_c , π_e and W_{TO} in question. Off-line tests showed that the algorithm was properly locating the minimum mass flow for each individual mission leg for an established leg aircraft weight. It was therefore concluded that the algorithm was not capable of properly dealing with the entire mission.

Recall that thrust required at each leg is dependent on the aircraft weight at the beginning of that leg. Also recall that the engine mass flow optimizer only modifies m_0 if insufficient thrust is experienced on a leg. Therefore, if m_0 is modified on leg t ($2 \leq t \leq \text{Total Number of Mission Legs}$), the algorithm will re-start evaluating the mission at leg 1. This is because the change in engine size will have changed the fuel consumption properties of all legs prior to leg t . Because W_{TO} remains constant throughout the iterative process, it is expected that by the time leg t is evaluated the second time (with the updated m_0), the aircraft will actually be lighter than it was on the original evaluation. This is because the engine now has a larger, less efficient m_0 than it had on the first evaluation, causing more fuel to have been consumed by the time leg t is encountered. The lighter aircraft weight causes the thrust required on leg t to be lower than it was on the first evaluation. However, the algorithm will not lower m_0 this time through because insufficient thrust is not encountered. The net result is that the engine ends up being larger than it needs to be.

An obvious solution to this problem would be to have the algorithm adjust mass flow (if necessary) based on the updated aircraft weight for leg t . Unfortunately, if m_0 is lowered on the second iteration of this algorithm, it is likely that on the third iteration m_0 will again be too small to meet thrust requirements (because aircraft weight will have increased from iteration 2 to 3). While this iterative process is likely to converge to the true best engine mass flow, the time spent performing the necessary iterations could be extremely time intensive.

One additional negative aspect of the mass flow optimizer is its incompatibility with non-constant engine installation loss models. Since with non-constant loss models the modification of engine size directly affects installation losses (which affects the thrust required for each mission leg), the complexity of converging to an optimal m_0 would become even greater than that previously described for the constant installation loss model. It is likely that even more mission evaluation iterations would be required to locate the optimal m_0 , further increasing the evaluation time for a single set of α , π_c , π_c and W_{TO} values.

Engine Mass Flow Optimization Conclusions. It is concluded that using the engine mass flow optimization algorithm is not appropriate for this application. The time required to iterate to an optimal m_0 value could be better spent making more, faster objective function evaluations treating m_0 as an independent variable. It was originally thought that removing this variable from the variable set controlled by the optimizer algorithm would benefit the overall convergence time. However, it is now believed that, given a robust non-linear optimization algorithm (like the genetic algorithm), better mass flow values could be obtained with less processing time without the mass flow optimizer.

3.5 Conclusions on Jet Engine Optimization Dimension Reduction

In summary, this research revealed both advantages and disadvantages of reducing the number of independent variables in a non-linear, non-convex optimization problem. The original supposition that problems with less design variables require less objective function evaluations was found to be true. However, the processing involved with exploiting variable dependencies can cause the overall optimizer convergence time to increase.

Using the fan pressure ratio optimization algorithm, it was possible to quickly identify a π_c value that was very close to optimal for a given bypass ratio and high pressure compressor

ratio. It also enabled the genetic algorithm to locate feasible engine solutions (engines capable of performing the mission) with much higher bypass ratios than were obtained without the fan pressure ratio optimizer. These high bypass ratio engines tended to be more fuel efficient than low bypass engines, thus reducing the amount of fuel required to perform the mission. This lower fuel weight translated to smaller, lighter conceptual aircraft designs.

In contrast, the engine mass flow optimization algorithm was plagued with inefficiency and non-optimal results. It required several iterative aircraft mission evaluations to converge to an optimal m_0 , effectively squelching any computational savings achieved by its use. The mass flow optimizer also routinely selected m_0 values that were larger than that required by the mission. One additional problem was that this iterative approach to identifying the best m_0 was found to be incompatible with a non-constant engine installation loss model, thus confounding this logical next-step improvement to the aircraft / mission optimization problem.

For future work on this problem, it is recommended that π_c be treated as a dependent variable and that m_0 remain part of the independent variable set controlled by the objective function optimizer. Although this leaves four independent variables to be optimized, this appears to be the best combination of independent variable reduction and objective function processing efficiency.

4. Kriging Techniques

4.1 Overview

One of the inherent weaknesses of all probabilistic non-linear optimization techniques is the large number of objective function evaluations required to locate the optimal solution. For many engineering applications, each objective function call involves the use of lengthy computer algorithms requiring from a few seconds to hundreds of hours to provide a solution for a single set of input variables. Compounding the problem, expensive computing resources may be needed to complete the complex calculations for each objective function evaluation. Even for applications with moderately computation-intensive objective functions, non-linear optimization can easily become too slow and costly for practical use.

While choosing the proper non-linear optimization technique can greatly improve the speed with which the global optimum is located, it is also wise to utilize any methods which make it possible to circumvent direct objective function evaluations. In particular, estimation techniques may be used to provide a more quickly obtained, approximate objective function value that does not hinder the optimization algorithm's efforts to locate the optimum.

One estimation technique that is well-known in the mining industry is called *kriging*. Its origin comes from mining engineers' need to estimate ore reserves contained in a volume of earth using only a limited number of point rock samples. As a means to this end, kriging is a linear estimation technique that provides a minimum variance estimate of the mineral concentration for a new geographic point by using information obtained from other samples in the vicinity of the new point. This approach to point estimation is easily applied in general to any problem involving evaluation of points throughout a design region.

In this application, recall that the original optimization problem contained five independent variables. Kriging was investigated after the dimension reduction techniques of Chapter 3 were studied. Therefore, the kriging method was applied when this optimization problem had three independent variables (instead of five): reference engine bypass ratio (α), high pressure compressor (core) pressure ratio (π_c), and aircraft gross takeoff weight (W_{TO}). The other two variables (fan compressor pressure ratio (π_f) and engine air mass flow (m_0)) were determined using the dependencies discussed in Chapter 3. In the previous chapter, it was concluded that m_0 should continue to be treated as an independent variable. Unfortunately, kriging research was performed before it was realized that m_0 optimization was impractical. This discrepancy does not affect the validity of the conclusions of this chapter because the kriging method can be applied to any number of independent variables. The general conclusions made in this chapter will hold true regardless of whether m_0 is treated as an independent variable or not.

4.2 Geostatistics

Before discussing how kriging can be applied to a problem, it is first necessary to understand the theory of how the technique works. Kriging makes use of a field of study known as geostatistics. Geostatistics uses a sampling of known responses from throughout a design space to describe the space's continuity. The geological science roots from which geostatistics arose relies on the smoothness of things in nature. That is, things in nature tend to form in a continuous, smooth fashion. For example, it is expected that earth samples that are rich in mineral content will be located in regions of rich mineral content, and earth samples that are low in mineral content will be located in regions of low mineral content. Additionally, transition from a region of rich mineral content to a region of low mineral content is expected to be gradual.

The Semi-Variogram. Much of statistics involves describing large sets of data with a few descriptive summary indices. Geostatistics is no different. An important tool in geostatistics is the *semi-variogram* (also known more casually as the *variogram*). At the heart of the semi-variogram is the assumption that all available data samples come from a population with a single mean and variance. Traditional statistics would assume that, regardless of where in a region a data point was sampled, it would have an equal probability of being above or below the mean value. Additionally, the covariance between any two sampled points would be zero (all responses, regardless of the proximity of the points to each other, are independent). However, in spatial statistics, it is assumed that points with responses above the mean will tend to be clumped together, and points with responses below the mean will also tend to be clumped together (due to the spatial continuity assumptions discussed earlier). The semi-variogram provides a description of how response data varies from point to point in the region of interest. In doing so, it provides a description of how points co-vary as a function of the distance (and possibly direction) between them.

As previously mentioned, an underlying assumption in spatial statistics is that all sampled data points come from a population with a single mean and variance. Since it is possible in geological sciences (and other applications) for data to reflect some *trend*, a common practice is to remove any trend from the data before creating the semi-variogram. In essence, this establishes that the mean response of the population (with trend removed) is zero and any deviation from the trend is due to the implicit variance of the population. A simple way to filter out any trend in response data is to create a second order linear regression, and then use the residual values (difference between the predicted linear regression response and the actual response) of each point to create the semi-variogram. This technique will be used when creating the semi-variogram for the jet engine optimization problem.

Method for Creating the Semi-Variogram. Applying the previously discussed principles of spatial statistics, it is now possible to create the semi-variogram using the following method:

1. Collect response data at many different locations in the region of interest.

Preferably, arrange the sample locations so that many points will be the same distance apart from each other.

2. Fit a second-order, least-squares linear regression to the response data.

Store the residual terms for each data point with the point's coordinates. These residual values will be used to create the semi-variogram.

3. Exhaustively calculate the Euclidean distance h between every pair of sample points. Store this distance along with the difference in residual values between the two points (Δ_{Residual}).

4. For any given value of h , collect all Δ_{Residual} 's that are associated with this distance, average the squared Δ_{Residual} values, and divide by two. This is the semi-variogram value $\gamma(h)$ for any two points in the region of interest that are a distance h apart. Written as equations:

$$\gamma(h) = \frac{1}{2N(h)} \sum_{(i,j) | h_{ij}=h} (v_i - v_j)^2 \quad (4-2)$$

where

$N(h)$ represents the number of points that are distance h apart

v_i and v_j represent the residual values at any points i and j that are distance h apart.

5. Repeat Step 4 for all h 's identified.
6. Plot each h and $\gamma(h)$ pair. This plot is the semi-variogram.
7. Attempt to fit a mathematical model to the data (to be discussed later). The mathematical model will be needed to make estimates of $\gamma(h)$ at all values of h (necessary to perform kriging).

It is important to note that the above methodology assumes the region is *isotropic*. That is, the expected variation in response is only a function of the distance between two points (h) and is not dependent on the direction of travel. If the expected variation in measured response was different for movement along different axes, the region would be considered *anisotropic*. For the purposes of this research, the design space is assumed to be isotropic (for reasons that will later be discussed).

Creation of Example Semi-Variogram. Working through an example is an excellent way to demonstrate how to create a semi-variogram. Suppose that a two-dimensional region exists, 10 units in length in both the x_1 and x_2 directions. At any point located in this region, we are able to evaluate a response $f(x_1, x_2)$. For this example, we will assume that the response surface is known. Obviously this is not usually the case, or we would not be trying to estimate responses via kriging.

The surface in Figure 4-1 is obtained by plotting the response as a function of the location in the (x_1, x_2) plane. The equation of the surface is shown in Eq (4-3).

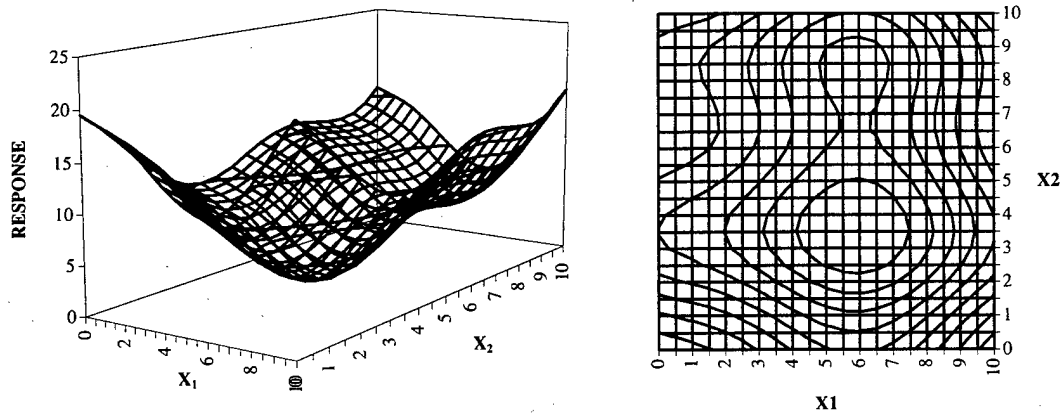


Figure 4-1. Response Surface for 2-D Kriging Example

$$RESPONSE = \left(\frac{x_1 - 5}{2} \right)^2 + \left(\frac{x_2 - 5}{2} \right)^2 + 2[\sin(0.75x_1) + \cos(x_2)] + 5 \quad (4-3)$$

In order to create a semi-variogram that will describe this surface, samples will need to be taken, preferably at standard intervals to create multiple pairs of points that are the same distance h from each other. A standard technique in mining is to create a grid of sample points that are evenly spaced throughout the region. Figure 4-2 shows the locations of the sample points for such a grid.

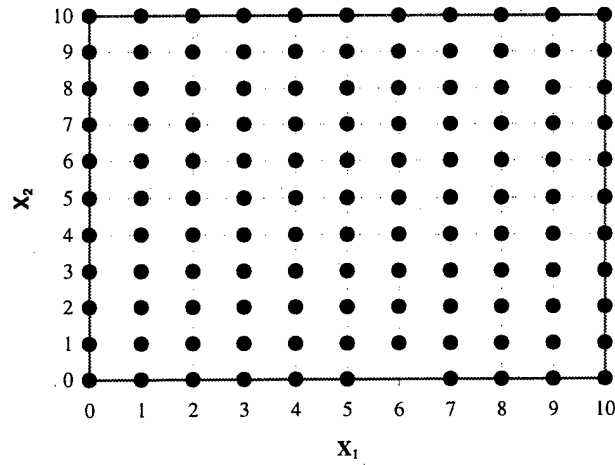


Figure 4-2. Sample Measurement Locations for 2-D Kriging Example

Evaluate the response at all of the sample points. Table 4-1 shows the tabular results of these evaluations.

Table 4-1. Response Evaluations for Kriging Example Sample Points

X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)
0	0	19.50	3	0	15.81	6	0	11.54	9	0	18.15
0	1	16.33	3	1	12.64	6	1	8.38	9	1	14.98
0	2	12.67	3	2	8.97	6	2	4.71	9	2	11.32
0	3	10.27	3	3	6.58	6	3	2.31	9	3	8.92
0	4	10.19	3	4	6.50	6	4	2.24	9	4	8.84
0	5	11.82	3	5	8.12	6	5	3.86	9	5	10.47
0	6	13.42	3	6	9.73	6	6	5.47	9	6	12.07
0	7	13.76	3	7	10.06	6	7	5.80	9	7	12.41
0	8	13.21	3	8	9.52	6	8	5.25	9	8	11.86
0	9	13.43	3	9	9.73	6	9	5.47	9	9	12.08
0	10	15.82	3	10	12.13	6	10	7.87	9	10	14.47
X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)
1	0	18.61	4	0	13.78	7	0	12.53	10	0	21.38
1	1	15.44	4	1	10.61	7	1	9.36	10	1	18.21
1	2	11.78	4	2	6.95	7	2	5.70	10	2	14.54
1	3	9.38	4	3	4.55	7	3	3.30	10	3	12.15
1	4	9.31	4	4	4.47	7	4	3.22	10	4	12.07
1	5	10.93	4	5	6.10	7	5	4.85	10	5	13.69
1	6	12.53	4	6	7.70	7	6	6.45	10	6	15.30
1	7	12.87	4	7	8.04	7	7	6.79	10	7	15.63
1	8	12.32	4	8	7.49	7	8	6.24	10	8	15.08
1	9	12.54	4	9	7.71	7	9	6.46	10	9	15.30
1	10	14.94	4	10	10.10	7	10	8.85	10	10	17.70
X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)	X ₁	X ₂	f(X ₁ , X ₂)			
2	0	17.49	5	0	12.11	8	0	14.94			
2	1	14.33	5	1	8.94	8	1	11.77			
2	2	10.66	5	2	5.27	8	2	8.11			
2	3	8.27	5	3	2.88	8	3	5.71			
2	4	8.19	5	4	2.80	8	4	5.63			
2	5	9.81	5	5	4.42	8	5	7.26			
2	6	11.42	5	6	6.03	8	6	8.86			
2	7	11.75	5	7	6.36	8	7	9.20			
2	8	11.20	5	8	5.82	8	8	8.65			
2	9	11.42	5	9	6.03	8	9	8.87			
2	10	13.82	5	10	8.43	8	10	11.26			

Now that sample data has been obtained, the first order of business is to remove any trend resident in the data. This is obtained by fitting a second-order, least-squares linear regression to the response data, which turns out to be:

$$RESPONSE = 19.9563 - 3.3706x_1 - 2.5853x_2 + 0x_1x_2 + 0.32895x_1^2 + 0.24201x_2^2 \quad (4-4)$$

After obtaining this second order curve-fit, the residual values must be calculated for the creation of the semi-variogram. Figure 4-3 shows a plot of the regression residuals for the region of interest. Table 4-2 lists the residual values in tabular form.

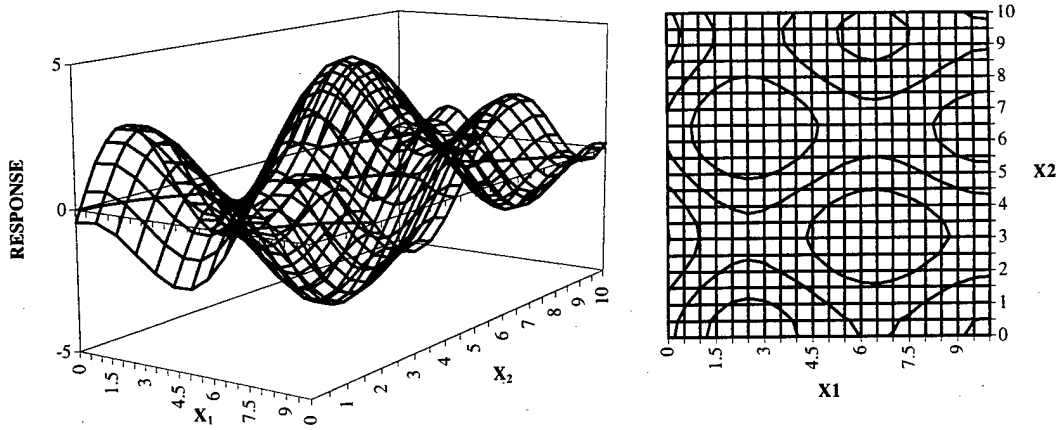


Figure 4-3. Plot of Residuals for 2-D Kriging Example

Table 4-2. Regression Residuals for Kriging Example

X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$
0	0	-0.46	3	0	3.00	6	0	-0.03	9	0	1.88
0	1	-1.28	3	1	2.18	6	1	-0.86	9	1	1.06
0	2	-3.09	3	2	0.37	6	2	-2.66	9	2	-0.75
0	3	-4.11	3	3	-0.65	6	3	-3.68	9	3	-1.77
0	4	-3.29	3	4	0.16	6	4	-2.87	9	4	-0.95
0	5	-1.26	3	5	2.19	6	5	-0.84	9	5	1.08
0	6	0.26	3	6	3.72	6	6	0.69	9	6	2.60
0	7	0.04	3	7	3.50	6	7	0.47	9	7	2.38
0	8	-1.55	3	8	1.90	6	8	-1.13	9	8	0.79
0	9	-2.86	3	9	0.59	6	9	-2.44	9	9	-0.52
0	10	-2.48	3	10	0.98	6	10	-2.06	9	10	-0.14
X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$
1	0	1.70	4	0	2.05	7	0	0.05	10	0	2.23
1	1	0.87	4	1	1.22	7	1	-0.77	10	1	1.40
1	2	-0.93	4	2	-0.58	7	2	-2.58	10	2	-0.40
1	3	-1.95	4	3	-1.61	7	3	-3.60	10	3	-1.42
1	4	-1.14	4	4	-0.79	7	4	-2.79	10	4	-0.61
1	5	0.89	4	5	1.24	7	5	-0.75	10	5	1.42
1	6	2.42	4	6	2.77	7	6	0.77	10	6	2.95
1	7	2.20	4	7	2.54	7	7	0.55	10	7	2.73
1	8	0.60	4	8	0.95	7	8	-1.05	10	8	1.13
1	9	-0.71	4	9	-0.36	7	9	-2.36	10	9	-0.18
1	10	-0.33	4	10	0.02	7	10	-1.97	10	10	0.20
X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$	X_1	X_2	$\text{Res}(X_1, X_2)$			
2	0	2.96	5	0	0.78	8	0	0.90			
2	1	2.14	5	1	-0.05	8	1	0.07			
2	2	0.33	5	2	-1.85	8	2	-1.73			
2	3	-0.69	5	3	-2.87	8	3	-2.76			
2	4	0.13	5	4	-2.06	8	4	-1.94			
2	5	2.16	5	5	-0.03	8	5	0.09			
2	6	3.68	5	6	1.50	8	6	1.62			
2	7	3.46	5	7	1.28	8	7	1.39			
2	8	1.87	5	8	-0.32	8	8	-0.20			
2	9	0.56	5	9	-1.63	8	9	-1.51			
2	10	0.94	5	10	-1.25	8	10	-1.13			

The next step is to determine all of the distances h represented by all of the possible pairings of data points. For example, each point's nearest neighbor in either the x_1 or x_2 direction is 1 unit away ($h = 1$). Each point's nearest diagonal neighbor is $\sqrt{2}$ units away ($h = \sqrt{2}$). By skipping the closest point in the x_1 or x_2 direction, the second closest point in the x_1 or x_2 direction is 2 units away ($h = 2$). This process continues until all possible point combinations are measured. Table 4-3 lists all h values represented by the sample, the number of pairs at this distance h , the sum of squared differences in the residual values for this distance h , and the semi-variogram value $\gamma(h)$. Figure 4-4 plots the data represented in Table 4-3 and represents how semi-variograms are typically presented.

Table 4-3. Exhaustive Semi-Variogram Calculations for Kriging Example

h	$N(h)$	$\sum_{(i,j):h_{ij}=h} (v_i - v_j)^2$	$\gamma(h) = \frac{1}{2N(h)} \sum_{(i,j):h_{ij}=h} (v_i - v_j)^2$
0.00	121	0.00	0.00
1.00	220	300.44	0.68
1.41	200	546.25	1.37
2.00	198	898.88	2.27
2.24	360	2125.95	2.95
2.83	162	1470.89	4.54
3.00	176	1233.67	3.50
3.16	320	2680.05	4.19
3.61	288	3326.20	5.77
4.00	154	1007.70	3.27
4.12	280	2214.55	3.95
4.24	128	1794.44	7.01
4.47	252	2792.98	5.54
5.00	356	3571.75	5.02
5.10	240	1302.07	2.71
5.39	216	1857.48	4.30
5.66	98	1282.52	6.54
5.83	192	2125.28	5.53
6.00	110	218.06	0.99
6.08	200	669.60	1.67
6.32	180	1173.99	3.26
6.40	168	1781.33	5.30
6.71	160	1438.70	4.50
7.00	88	93.11	0.53
7.07	232	972.38	2.10
7.21	140	1193.62	4.26
7.28	144	806.08	2.80
7.62	128	1032.64	4.03
7.81	120	725.04	3.02
8.00	66	108.72	0.82
8.06	232	1212.91	2.61
8.25	108	668.20	3.09
8.49	50	198.24	1.98
8.54	96	831.05	4.33
8.60	96	491.30	2.56
8.94	84	688.02	4.10
9.00	44	177.87	2.02
9.06	80	432.65	2.70
9.22	152	861.16	2.83
9.43	72	410.90	2.85
9.49	64	707.33	5.53
9.85	56	592.82	5.29
9.90	32	67.71	1.06
10.00	82	347.70	2.12
10.05	40	290.85	3.64
10.20	36	376.03	5.22
10.30	48	388.91	4.05
10.44	32	413.28	6.46
10.63	48	129.85	1.35
10.77	28	348.57	6.22
10.82	40	241.00	3.01
11.18	24	239.17	4.98
11.31	18	59.30	1.65
11.40	32	163.22	2.55
11.66	20	157.76	3.94
12.04	24	136.55	2.84
12.21	16	111.42	3.48
12.73	8	64.68	4.04
12.81	12	90.63	3.78
13.45	8	79.59	4.97
14.14	2	23.62	5.91

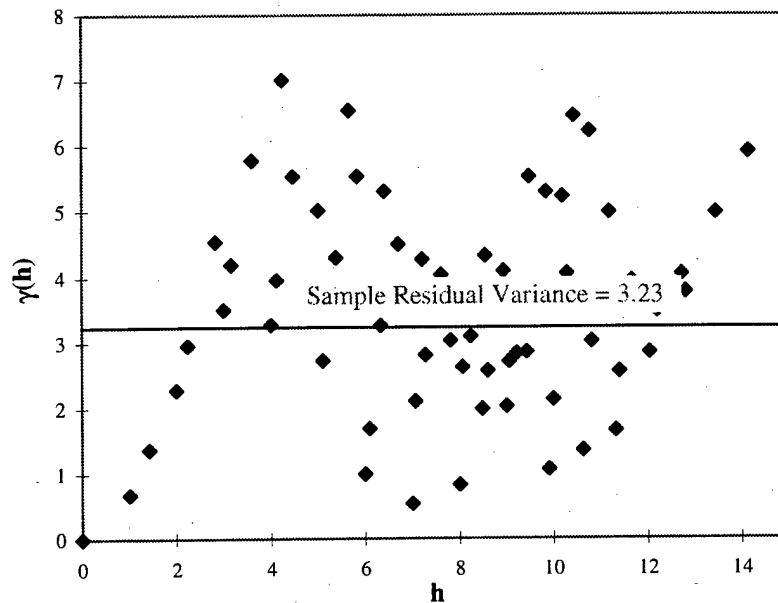


Figure 4-4. Graphical Representation of Semi-Variogram Data for Kriging Example

Comments on the Example Semi-Variogram. As is often the case with semi-variograms, Figure 4-4 is not as clean as we would like it to be. In theory, the semi-variogram should gradually rise from $\gamma(h=0) = 0$ to some relatively steady-state value that closely resembles the sample variance (s^2) (annotated in Figure 4-4). This pattern is expected because, once the immediate neighborhood of any point (in which response values are expected to be correlated) is departed, response values of any new points will be uncorrelated. Thus, if a sample of responses were taken at an arbitrary large h from any point in the design space, it is expected that the responses would be independent and their variance would approximate the entire population's variance (recall that in spatial statistics, the variance of the entire population is assumed constant).

The steady-state $\gamma(h)$ value for large h is known as the semi-variogram's *sill*. The value of h at which the sill is attained is known as the semi-variogram's *range*. The h -value of the

range is denoted by h_{range} . For semi-variogram models with a sill, the sill value is equal to the sample (residuals') variance. That is to say

$$\lim_{h \rightarrow h_{max}} \gamma(h) = s^2 = \frac{1}{m-1} \sum_{i=1}^m (v_i - \bar{v})^2 \quad (4-5)$$

where:

h is the distance between points being evaluated

h_{max} is the maximum h value represented in the response data

$\gamma(h)$ is the semi-variogram value for the distance h

s^2 is the sample (residuals') variance

m is the total number of points in the sample

v_i is the response value at data point i

\bar{v} is the average response for all of the data samples, calculated using:

$$\bar{v} = \frac{1}{m} \sum_{i=1}^n v_i \quad (4-6)$$

where, again, m is the total number of data points in the sample.

(Clark, 1987, 7).

While this example's semi-variogram does show a typical rising $\gamma(h)$ pattern for small values of h , a clear sill value is not evident from the data. Regardless, this semi-variogram (as well as all semi-variograms used in the jet engine optimization problem) is assumed to have a sill equal to s^2 . The only justification for this apparently gross assumption is that accurate kriging estimates are still obtained, both in this example and in application. For mining applications, significant deviations from the sill value for values of $h > h_{range}$ may indicate some important

geological phenomena. However, for this application, any deviations from this theoretical sill value are ignored.

It is worth noting that the previous discussion on creating a semi-variogram is equally valid in the absence of a sample grid. This fact is important in the jet engine optimization problem at hand. The jet engine optimization is controlled by a *micro-genetic algorithm* (μ GA) routine that uses a random search technique to locate the design variable values that achieve the best objective function value. Rather than use computing resources for the sole purpose of creating a semi-variogram sampling grid, response samples are obtained on-the-fly as the μ GA searches the design region. This creates a very irregular sampling pattern and causes every pair of sample points to have its own unique h_{ij} value. Regardless, the semi-variogram can still be generated. The only difference is that a point will exist on the semi-variogram for every pair of sample points instead of one semi-variogram point representing an average of several sample point pairs ($N(h) = 1$ for all h). All of the equations listed in this discussion are still valid. In addition, the following discussion about mathematical modeling is also valid despite the irregular sampling pattern.

Mathematical Modeling of the Semi-Variogram. The final step in creating a semi-variogram is the development of a mathematical model to represent the semi-variogram data. This model is what will be used for kriging since kriging requires semi-variogram values for all possible values of h , not just $\gamma(h)$ values for h 's represented in the sample data. In both the Clark (1987) and Isaaks et al. (1989) texts, numerous semi-variogram models are discussed, and the reader is referred to either of these resources for more in-depth modeling information than is about to be presented.

Mathematical Modeling Technique. For this application, semi-variogram models are to be automatically generated in an effort to use kriging to replace actual objective

function evaluations. When a user begins the optimization process, no a priori information is known about the objective function's response surface, let alone the semi-variogram that would spatially describe this response surface. Thus, any best-model decisions about which semi-variogram model to use would not be possible without human intervention (which was undesirable in this application). Additionally, developing software to smartly choose the best model was impractical. It therefore was decided that a *linear* model (with a sill) would be used to represent the semi-variogram data. This model's primary advantage was its simplicity in creation and use. It also proved to be an effective assumption, producing good kriging results.

Essentially, the linear model used assumes that $\gamma(h)$ increases linearly until the sample variance (s^2) is obtained. The h -value at which the $\gamma(h) = s^2$ is considered to be the range, and $\gamma(h)$ is assumed to equal s^2 for any $h \geq h_{range}$. The slope of the line connecting $\gamma(h=0) = 0$ and $\gamma(h_{range}) = s^2$ is determined by the following technique:

1. For every h -value in the semi-variogram, calculate the slope of the line that would be required to connect the origin and the point. This slope is nothing more than $\frac{\gamma(h)}{h}$ for the point since the line is from the origin (0,0) to the point.
2. Obtain the average of the slopes implied by each semi-variogram point. This becomes the slope of the semi-variogram for $h < h_{range}$.
3. Calculate the value of h at which the line intercepts $\gamma(h) = s^2$. This value of h is the model's range. For any value of $h \geq h_{range}$, $\gamma(h) = s^2$.

Mathematical Modeling Example. To demonstrate this process, a linear model will be fit to the example previously discussed. As previously stated, the sample variance (that

is, the sample variance of the residuals being used to create the semi-variogram) is $s^2 = 3.23$.

First, the slope obtained from each individual point is calculated. Table 4-4 lists this data.

Table 4-4. Tabular Listing of Semi-Variogram Data Used for Creating Linear Model

h	$\gamma(h)$	$\frac{\gamma(h)}{h}$	h	$\gamma(h)$	$\frac{\gamma(h)}{h}$
0.00	0.00	N/A	8.25	3.09	0.38
1.00	0.68	0.68	8.49	1.98	0.23
1.41	1.37	0.97	8.54	4.33	0.51
2.00	2.27	1.13	8.60	2.56	0.30
2.24	2.95	1.32	8.94	4.10	0.46
2.83	4.54	1.61	9.00	2.02	0.22
3.00	3.50	1.17	9.06	2.70	0.30
3.16	4.19	1.32	9.22	2.83	0.31
3.61	5.77	1.60	9.43	2.85	0.30
4.00	3.27	0.82	9.49	5.53	0.58
4.12	3.95	0.96	9.85	5.29	0.54
4.24	7.01	1.65	9.90	1.06	0.11
4.47	5.54	1.24	10.00	2.12	0.21
5.00	5.02	1.00	10.05	3.64	0.36
5.10	2.71	0.53	10.20	5.22	0.51
5.39	4.30	0.80	10.30	4.05	0.39
5.66	6.54	1.16	10.44	6.46	0.62
5.83	5.53	0.95	10.63	1.35	0.13
6.00	0.99	0.17	10.77	6.22	0.58
6.08	1.67	0.28	10.82	3.01	0.28
6.32	3.26	0.52	11.18	4.98	0.45
6.40	5.30	0.83	11.31	1.65	0.15
6.71	4.50	0.67	11.40	2.55	0.22
7.00	0.53	0.08	11.66	3.94	0.34
7.07	2.10	0.30	12.04	2.84	0.24
7.21	4.26	0.59	12.21	3.48	0.29
7.28	2.80	0.38	12.73	4.04	0.32
7.62	4.03	0.53	12.81	3.78	0.29
7.81	3.02	0.39	13.45	4.97	0.37
8.00	0.82	0.10	14.14	5.91	0.42
8.06	2.61	0.32			

Averaging the slope estimates represented by each point (the $\frac{\gamma(h)}{h}$ values) yields a line slope equal to 0.5740. With this value in hand, we calculate the value of h at which the sill (which is equal to s^2) is attained. This is the range (h_{range}).

$$h_{range} = \frac{s^2}{slope} = \frac{3.23}{0.5740} = 5.627 \quad (4-7)$$

Thus, the mathematical model is stated as

$$\gamma(h_{ij}) = \begin{cases} 0.5740 \cdot h_{ij} & 0 \leq h_{ij} < 5.627 \\ 3.23 & h_{ij} \geq 5.627 \end{cases} \quad (4-8)$$

Figure 4-5 shows the resulting semi-variogram model superimposed on the semi-variogram data.

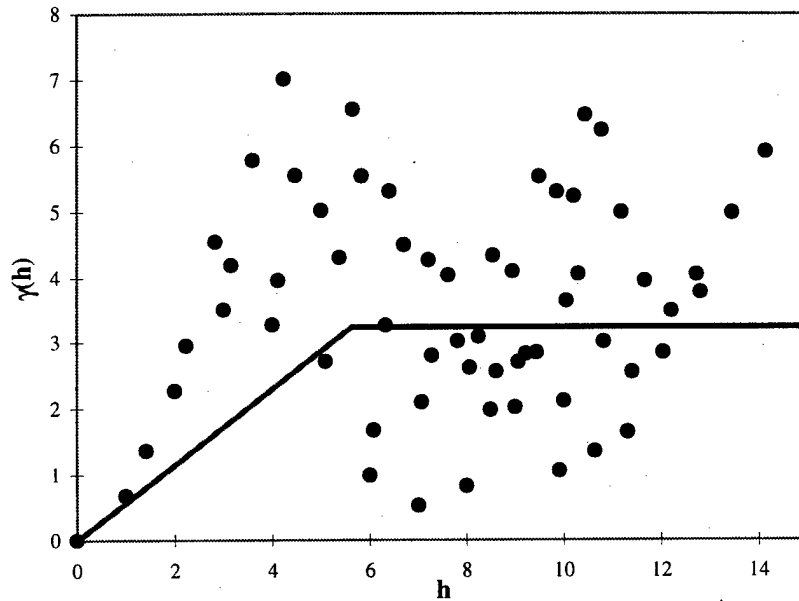


Figure 4-5. Kriging Example Semi-Variogram with Fitted Linear Model

Clearly, the linear model does not produce an impressive fit to the semi-variogram data.

However, in practice for this application, the linear model approximation was adequate for generating accurate kriging estimates.

4.3 Kriging

Once the semi-variogram has been created to describe the spatial continuity of the region of interest and a mathematical model has been generated to approximate the semi-variogram data, it is possible to use this information to create a minimum variance, linear estimate of unsampled points in the region using kriging.

Background on Estimation Techniques. Say that a set of sample points exists with coordinates (represented in vector form) $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_n$, response values $v_1, v_2, v_3, \dots, v_n$ ($v_i = f(\mathbf{x}_i)$, for all i from 1 to n), and *trend-less* response values $r_1, r_2, r_3, \dots, r_n$ (where r_i is the residual response value after trend has been removed). Now say that we are interested in approximating the response value at a new point \mathbf{x}_0 that is in the proximity of the known points. One technique for estimating v_0 ($v_0 = f(\mathbf{x}_0)$) would be to form a weighted average (that is, a convex, linear combination) of the known values. Written as equations:

$$\hat{v}_0 = w_1 v_1 + w_2 v_2 + w_3 v_3 + \dots + w_n v_n = \sum_{i=1}^n w_i v_i \quad (4-9)$$

$$\sum_{i=1}^n w_i = 1. \quad (4-10)$$

where

\hat{v}_0 is an unbiased estimator of v_0 , the actual response of the point being

estimated

v_i is the response value for point i

w_i is the weight applied to response value i

n is the number of data points being used to generate the estimate

Regardless of the actual values of the w_i , as long as they sum to one and the sampled values were without trend, \hat{v}_0 will be an unbiased estimator of v_0 . That is to say that if numerous estimations of different points were to occur using the sample set data, the average estimation error would equal zero. (Clark, 1987, 99)

Now that we know how to keep estimates unbiased, we turn our attention to choosing the averaging weights (the w_i 's) in a way that gives us the best possible estimation results. Numerous criteria exist to choose w_i values, ranging from setting all w_i 's equal to each other to sizing w_i based on how far away x_i is from x_0 . However, kriging introduces a method for selecting w_i values that produce a *minimum variance* estimate of v_0 .

Kriging Estimation Technique. Generating a minimum variance estimate from a set of known sample points makes use of the semi-variogram information previously discussed. Recall that the semi-variogram shows the relative variation in the trend-less residual response (r_i) values as a function of the distance between any two sample points in the region of interest. Using this spatial information, it is possible to estimate the variance of a new point's (x_0) response value (v_0). Estimation variance (σ_0^2) for any unbiased linear estimation is

$$\sigma_0^2 = 2 \sum_{i=1}^n w_i \cdot \gamma(h_{i0}) - \sum_{i=1}^n \sum_{j=1}^n w_i \cdot w_j \cdot \gamma(h_{ij}) \quad (4-11)$$

where

h_{ij} is the Euclidean distance from point i to point j (a subscript of 0 represents the point being estimated)

$\gamma(h_{ij})$ is the Semi-variogram value for the distance h_{ij}

w_i is the weighting applied to point i (recall, $\sum_{i=1}^n w_i = 1$)

n is the number of data points being used to generate the kriged estimate

It is important to note that Eq (4-11) has been modified from its original form presented by Clark (1987). Kriging, when typically used in geological applications, is employed to estimate a response for a region of points, not one specific point. For this reason, Eq (4-11) in its original form also has a term accounting for variation present within the region being estimated. For point estimation, no variation exists between the point and itself. Thus, this term is dropped from the overall variance formula. The elimination of this constant term does not affect the validity of the kriging techniques to be derived.

To obtain the w_i 's that produces a minimum variance estimate, the w_i 's must be selected so as to minimize σ_0^2 . To identify the minimum σ_0^2 , we locate the set of w_i 's at which all partial derivatives of the σ_0^2 function (with respect to all of the w_i 's) equals zero. That is, solve for the set of w_i 's that makes

$$\frac{\partial \sigma_0^2}{\partial w_i} = 0 \quad \text{for } i = 1, 2, 3, \dots, n \quad (4-12)$$

Taking these n partial derivatives produces n equations with n unknowns. However, nothing in this set of equations ensures that $\sum_{i=1}^n w_i = 1$, which is required for the estimation technique to be unbiased. Since simply introducing this last equation would create $(n+1)$ equations with only n unknowns, one further unknown is introduced in the form of a Lagrangian Multiplier (λ). In effect, we return to Eq (4-11) and choose to minimize

$$\sigma_0^2 - \lambda \left(\sum_{i=1}^n (w_i) - 1 \right) \quad (4-13)$$

instead of only σ_0^2 . To minimize this equation, partial derivatives with respect to each of the w_i 's and λ must be set equal to zero. The resulting $(n+1)$ equations with $(n+1)$ unknowns are

$$\begin{aligned} w_1 \cdot \gamma(h_{11}) + w_2 \cdot \gamma(h_{12}) + w_3 \cdot \gamma(h_{13}) + \dots + w_n \cdot \gamma(h_{1n}) + \lambda &= \gamma(h_{10}) \\ w_1 \cdot \gamma(h_{21}) + w_2 \cdot \gamma(h_{22}) + w_3 \cdot \gamma(h_{23}) + \dots + w_n \cdot \gamma(h_{2n}) + \lambda &= \gamma(h_{20}) \\ w_1 \cdot \gamma(h_{31}) + w_2 \cdot \gamma(h_{32}) + w_3 \cdot \gamma(h_{33}) + \dots + w_n \cdot \gamma(h_{3n}) + \lambda &= \gamma(h_{30}) \\ &\vdots \\ w_1 \cdot \gamma(h_{n1}) + w_2 \cdot \gamma(h_{n2}) + w_3 \cdot \gamma(h_{n3}) + \dots + w_n \cdot \gamma(h_{nn}) + \lambda &= \gamma(h_{n0}) \\ w_1 + w_2 + w_3 + \dots + w_n + 0 &= 1 \end{aligned} \quad (4-14)$$

or, stated in the more compact matrix form

$$\begin{bmatrix} \gamma(h_{11}) & \gamma(h_{12}) & \dots & \gamma(h_{1n}) & 1 \\ \gamma(h_{21}) & \gamma(h_{22}) & \dots & \gamma(h_{2n}) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma(h_{n1}) & \gamma(h_{n2}) & \dots & \gamma(h_{nn}) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ \lambda \end{bmatrix} = \begin{bmatrix} \gamma(h_{10}) \\ \gamma(h_{20}) \\ \vdots \\ \gamma(h_{n0}) \\ 1 \end{bmatrix} \quad (4-15)$$

For convenience, names are given to the matrix and two vectors in Eq (4-15).

$$\Gamma \cdot \bar{w} = \bar{\gamma}_0 \quad (4-16)$$

where

$$\Gamma = \begin{bmatrix} \gamma(h_{11}) & \gamma(h_{12}) & \cdots & \gamma(h_{1n}) & 1 \\ \gamma(h_{21}) & \gamma(h_{22}) & \cdots & \gamma(h_{2n}) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma(h_{n1}) & \gamma(h_{n2}) & \cdots & \gamma(h_{nn}) & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix}$$

$$\bar{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ \lambda \end{bmatrix} \quad \bar{\gamma}_0 = \begin{bmatrix} \gamma(h_{10}) \\ \gamma(h_{20}) \\ \vdots \\ \gamma(h_{n0}) \\ 1 \end{bmatrix}$$

Now solving for the weights that produce the minimum variance estimate is a simple linear algebra operation.

$$\bar{w} = \Gamma^{-1} \cdot \bar{\gamma}_0 \quad (4-17)$$

where

Γ^{-1} is the inverse of the Γ matrix defined in Eq (4-16). Each $\gamma(h_{ij})$ element of the Γ matrix comes from evaluating the semi-variogram for the distance h_{ij} (the distance from sample point i to point j).

$\bar{\gamma}_0$ is defined as in Eq (4-16). Each $\gamma(h_{i0})$ element of the $(n+1)$ by 1 $\bar{\gamma}_0$ vector comes from evaluating the semi-variogram for the distance h_{i0} (the distance from sample point i to the point being estimated).

\bar{w} is the $(n+1)$ by 1 vector of weights plus the Lagrangian Multiplier λ (see Eq (4-16) for details).

With the minimum variance estimate weights now in hand, the response value for the point in question, x_0 , is easily obtained using the original response values (the v_i values). Recall that up to this point we have been using residual (r_i) values to create the semi-variogram, build the mathematical model and determine the w_i 's that will produce a minimum variance estimate of v_0 . However, to generate the v_0 estimate (instead of the r_0 estimate), we must apply the w_i 's to the v_i 's, not the r_i 's. Thus, to obtain the estimate, we use

$$\hat{v}_0 = w_1 v_1 + w_2 v_2 + w_3 v_3 + \dots + w_n v_n = \sum_{i=1}^n w_i v_i \quad (4-18)$$

As with any estimation technique, there is some amount of uncertainty about the validity of the estimate value. In regression, parameter standard errors are used to calculate confidence intervals for new response predictions. In kriging, an estimate variance is generated directly from the information used to develop the response estimate using

$$\sigma_0^2 = \sum_{i=1}^n (w_i \cdot \gamma(h_{i0})) + \lambda = \bar{w}^T \cdot \bar{\gamma}_0 \quad (4-19)$$

where \bar{w} and $\bar{\gamma}_0$ are defined as they were in Eq (4-17). (Clark, 1987, Chapter 5)

As was the case in Eq (4-11), Eq (4-19) also has been modified from the form present in Clark (1987). Again, this is because point estimation is being performed, not region estimation. Though the derivation is in a different form, Isaaks, et al. (1989) confirms this response estimate result. Additionally, the validity of the kriging equation (Eq (4-17)) for point estimation (instead of region estimation) is confirmed in this alternate reference.

Notice that this estimate variance differs from a traditional estimate variance (say from regression) in that it is not attempting to capture uncertainty in the estimate as a function of

uncertainty in the data used to create the estimate (presumably introduced through measurement error or some other unmodeled noise source). Instead, the estimate variance is a function of spatial variance as described by the semi-variogram. This subtle distinction is important because estimate uncertainty is no longer a result of uncertainty in the data, but is a result of the response variance present throughout the design space. For this reason, estimate variance may or may not be directly correlated to the accuracy of the prediction. This concept will be further discussed in section 4.4.

Kriging Example. Returning to the example started in section 4.2, let us now attempt to estimate the response at the new point $x_0 = (5.4, 7.2)$. Rather than use all 121 points, we will arbitrarily choose to use the 7 closest known sample points to x_0 . Figure 4-6 shows the point in question in proximity to the 7 closest sample points.

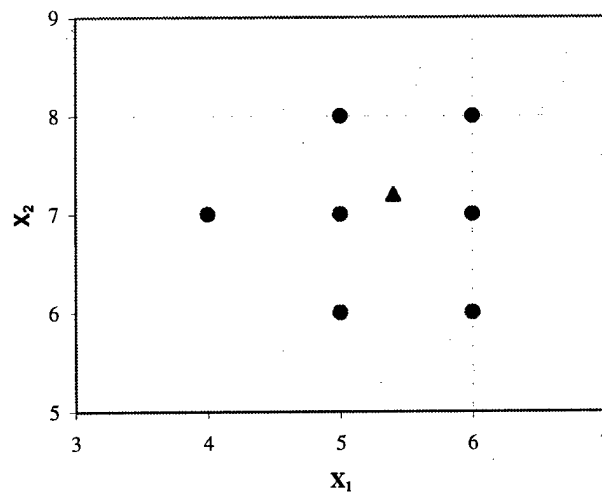


Figure 4-6. Selection of Closest Points for Estimating $v_0 = f(x_0)$

To begin the kriging process, the distances from the seven closest sample points to the point being estimated (the h_{i0} 's) must be calculated. Also, the distances from the sample points to each other (the h_{ij} 's) must be calculated as well. Semi-variogram values must then be assessed

for each of these distances for use in the kriging system of equations. Table 4-5 through Table 4-7 show all of the necessary data to perform kriging.

Table 4-5. Distance and Semi-Variogram Data Between Sample Points and x_0 for Kriging Example

Point # (i)	(x_1, x_2) Coordinates of Point x_i	Response Value (v_i) from Table 4-1 (trend included)	Euclidean Distance to x_0 (h_{i0})	$\gamma(h_{i0})$ (From Eq (4-8))
1	(4,7)	8.04	1.41	0.81
2	(5,6)	6.03	1.26	0.73
3	(5,7)	6.36	0.45	0.26
4	(5,8)	5.82	0.89	0.51
5	(6,6)	5.47	1.34	0.77
6	(6,7)	5.80	0.63	0.36
7	(6,8)	5.25	1	0.57

Table 4-6. Matrix of Distances Between Sample Points for Kriging Example

Point #		1	2	3	4	5	6	7
	(x_1, x_2)	(4,7)	(5,6)	(5,7)	(5,8)	(6,6)	(6,7)	(6,8)
1	(4,7)	0.00	1.41	1.00	1.41	2.24	2.00	2.24
2	(5,6)	1.41	0.00	1.00	2.00	1.00	1.41	2.24
3	(5,7)	1.00	1.00	0.00	1.00	1.41	1.00	1.41
4	(5,8)	1.41	2.00	1.00	0.00	2.24	1.41	1.00
5	(6,6)	2.24	1.00	1.41	2.24	0.00	1.00	2.00
6	(6,7)	2.00	1.41	1.00	1.41	1.00	0.00	1.00
7	(6,8)	2.24	2.24	1.41	1.00	2.00	1.00	0.00

Table 4-7. Matrix of Semi-Variogram Values Between Sample Points for Kriging Example

Point #		1	2	3	4	5	6	7
	(x_1, x_2)	(4,7)	(5,6)	(5,7)	(5,8)	(6,6)	(6,7)	(6,8)
1	(4,7)	0.00	0.81	0.57	0.81	1.28	1.15	1.28
2	(5,6)	0.81	0.00	0.57	1.15	0.57	0.81	1.28
3	(5,7)	0.57	0.57	0.00	0.57	0.81	0.57	0.81
4	(5,8)	0.81	1.15	0.57	0.00	1.28	0.81	0.57
5	(6,6)	1.28	0.57	0.81	1.28	0.00	0.57	1.15
6	(6,7)	1.15	0.81	0.57	0.81	0.57	0.00	0.57
7	(6,8)	1.28	1.28	0.81	0.57	1.15	0.57	0.00

Using Eq (4-16), the component matrix and vectors are generated.

$$\Gamma = \begin{bmatrix} 0 & 0.81 & 0.57 & 0.81 & 1.28 & 1.15 & 1.28 & 1 \\ 0.81 & 0 & 0.57 & 1.15 & 0.57 & 0.81 & 1.28 & 1 \\ 0.57 & 0.57 & 0 & 0.57 & 0.81 & 0.57 & 0.81 & 1 \\ 0.81 & 1.15 & 0.57 & 0 & 1.28 & 0.81 & 0.57 & 1 \\ 1.28 & 0.57 & 0.81 & 1.28 & 0 & 0.57 & 1.15 & 1 \\ 1.15 & 0.81 & 0.57 & 0.81 & 0.57 & 0 & 0.57 & 1 \\ 1.28 & 1.28 & 0.81 & 0.57 & 1.15 & 0.57 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad \bar{\gamma}_0 = \begin{bmatrix} 0.81 \\ 0.73 \\ 0.26 \\ 0.51 \\ 0.77 \\ 0.36 \\ 0.57 \\ 1 \end{bmatrix}$$

Using Eq (4-17), the minimum variance weights are obtained.

$$\bar{w} = \Gamma^{-1} \cdot \bar{\gamma}_0 =$$

$$\begin{bmatrix} -1.1145 & 0.4304 & 0.5561 & 0.4304 & -0.0750 & -0.1523 & -0.0750 & 0.3530 \\ 0.4304 & -1.5521 & 0.5704 & -0.1469 & 0.7837 & -0.0017 & -0.0838 & 0.1572 \\ 0.5561 & 0.5704 & -2.2354 & 0.5704 & -0.0231 & 0.5843 & -0.0231 & -0.2631 \\ 0.4304 & -0.1469 & 0.5704 & -1.5521 & -0.0838 & -0.0017 & 0.7837 & 0.1572 \\ -0.0750 & 0.7837 & -0.0231 & -0.0838 & -1.3909 & 0.7749 & 0.0143 & 0.3172 \\ -0.1523 & -0.0017 & 0.5843 & -0.0017 & 0.7749 & -1.9787 & 0.7749 & -0.0387 \\ -0.0750 & -0.0838 & -0.0231 & 0.7837 & 0.0143 & 0.7749 & -1.3909 & 0.3172 \\ 0.3530 & 0.1572 & -0.2631 & 0.1572 & 0.3172 & -0.0387 & 0.3172 & -0.8722 \end{bmatrix} \begin{bmatrix} 0.81 \\ 0.73 \\ 0.26 \\ 0.51 \\ 0.77 \\ 0.36 \\ 0.57 \\ 1 \end{bmatrix} =$$

$$\bar{w} = \begin{bmatrix} -0.027 \\ 0.001 \\ 0.493 \\ 0.137 \\ -0.004 \\ 0.314 \\ 0.086 \\ -0.049 \end{bmatrix}$$

The kriged estimate is obtained using Eq (4-18) and the trend-less response data in Table 4-5.

$$\hat{v}_0 = (-0.027)(2.54) + (0.001)(1.50) + (0.493)(1.28) + \dots$$

$$(0.137)(-0.32) + (-0.004)(0.69) + (0.314)(0.47) + (0.086)(-1.13)$$

$$\hat{v}_0 = 5.973$$

The variance of this estimate is also readily available using Eq (4-19).

$$\sigma_0^2 = \bar{w}^T \cdot \bar{\gamma}_0 = 0.287$$

To verify the quality of this estimate, we now compare it with the actual response value at point x_0 . This true value is obtained using Eq (4-3), from which the original response surface for this example was generated.

$$v_0 = f(x_0) = 5.890$$

We see that kriging has produced a response estimate that is 0.083 units (1.42% difference) from the actual value.

Notes on the Kriging Process. Having been through the arduous process of creating a semi-variogram, representing the semi-variogram mathematically, and then developing a minimum variance estimate via kriging, the reader may be confused as to the value of this estimation technique given that the true response value could have been obtained simply by using Eq (4-3). Keep in mind, however, that this was simply an example to demonstrate the process of kriging, not an example for which kriging was intended. Kriging is intended to save time and money by avoiding actual response surface evaluations. In geological applications, response surface evaluations require expensive drilling and laboratory analysis. When applied to the jet engine optimization problem discussed later in this paper, response function evaluations require significant computational effort which makes the optimization process very slow. Even though the kriging process (to include modeling the semi-variogram) involves processing a significant amount of response information, this process can be readily automated and executed in a fraction of the time required to evaluate the true response function.

4.4 Evaluating the Quality of the Kriged Estimate

Once a response estimate has been obtained, it is desirable to assess the accuracy of the estimated value. As discussed in section 4.3, it is inappropriate to assume that estimate variance is the best indicator of the estimate's accuracy. In the previous example, we were able to simply evaluate the true response at point x_0 and compare it with the estimated response. Obviously, performing this type of accuracy assessment in a real application would defeat the purpose of developing a response estimate. What we want is an indicator from the kriging process that can be directly correlated to prediction error. Then, we can set a threshold on this indicator parameter at which the kriged estimate is either accepted or rejected. If it is rejected, then an actual response function evaluation will be required.

Possible Estimation Accuracy Indicators. To begin the search for an appropriate indicator parameter, it is helpful to list parameters that could reasonably be expected to be correlated with prediction error.

1. Kriging estimate variance (σ_0^2) (from Eq (4-19))
2. Kriging estimate standard deviation (σ_0) (equal to $+\sqrt{\sigma_0^2}$)
3. Distance of any of the n sample points to the point being estimated (h_{i0})
4. Average of all of the h_{i0} values $\left(\bar{h}_0 = \frac{1}{n} \cdot \sum_{i=1}^n h_{i0} \right)$

A review of kriging literature revealed that estimate variance (σ_0^2) was most commonly used as the kriged estimate's measure of accuracy. However, given that, in kriging, estimate variance is a function of the spatial properties of the region and not a function of the response data, it was not clear that a connection necessarily existed between variance and accuracy. While this assumption is intuitive for other estimation techniques (i.e. regression), it seemed prudent to verify this proposal by performing experimentation on sample response surfaces. For the

purposes of this experimentation, all of the potential accuracy indicators (including variance) will be evaluated.

Method for Determining Good Indicator Parameters. In the jet engine optimization problem, three design variables exist (recall that two of the original five variables have been eliminated through the techniques discussed in Chapter 3). Very little a priori information exists about the objective function response surface, primarily because it changes significantly based on the values of other preset constants in the problem. It has been deduced by experience that the response surface is relatively smooth and is likely to contain trend that will need to be filtered before kriging is performed.

Because of the unpredictability of the jet engine response surface, using the jet engine evaluation codes to develop a general understanding of predictor parameters is no more desirable than using a simulated response surface. For this reason, a reasonable response surface (that is, a surface with smooth peaks and troughs) with three input variables was used to isolate promising predictor parameters. While this may appear to be an arbitrary means of locating the best indicator variable, application of experimentation results to the jet engine optimization problem enabled satisfactory distinction to be made between accurate and inaccurate kriged estimates.

The response surface chosen to perform experimentation is actually a three-design-variable variation of the two-design-variable example surface introduced in section 4.2. The new surface's mathematical representation is

$$RESPONSE = \left(\frac{x_1 - 5}{2}\right)^2 + \left(\frac{x_2 - 5}{2}\right)^2 + \left(\frac{x_3 - 5}{2}\right)^2 + 2[\sin(0.75x_1) + \cos(x_2) + \sin(0.50x_3)] + 25 \quad (4-20)$$

The design region is defined as

$$0 \leq x_1 \leq 10$$

$$0 \leq x_2 \leq 10$$

$$0 \leq x_3 \leq 10$$

The goal is to first establish an isotropic semi-variogram that describes this response surface (with trend removed) and then perform numerous kriging estimations at random locations in the design space that can be compared with actual response values. Once kriging parameters from each estimate and the differences between the estimated responses and the actual responses have been obtained, it will be possible to look for correlations between the parameters and the prediction errors.

The experiment was set up as follows:

1. 50 randomly selected sample points were established in the design region via direct response function evaluation ($m = 50$).
2. The region's semi-variogram was generated using these sample points.
3. A linear mathematical model (with a sill) was applied to represent the semi-variogram data.
4. Response estimates for 1000 randomly selected points were generated using the kriging technique. Five sample points were used to generate each kriged estimate ($n = 5$).
5. For each of the 1000 points estimated via kriging, actual response values were obtained in order to calculate kriging estimation error.
6. The magnitude of the estimation error was stored along with the various kriging indicator parameters for each point estimated via kriging.
7. Off-line analysis on the data was performed to look for correlations between the various indicator parameters and the estimation error.

Best Estimation Indicators. The results of this experiment show that either the kriged estimate variance or standard deviation is the best indicator of kriging estimation error. Table 4-8 shows the correlation of each potential indicator with the magnitude of the prediction error.

Table 4-8. Predictor Variable Correlation with Prediction Error Magnitude

Predictor	Correlation with Prediction Error
σ_0^2	0.672417
σ_0	0.639926
h_{10}	0.534942
h_{20}	0.46985
h_{30}	0.419282
h_{40}	0.377354
h_{50}	0.368472
\bar{h}_0	0.514043

It is seen that estimate variance (σ_0^2) is most highly correlated, thus affirming its utility as an accuracy predictor as presented in various literature references. However, something that is not captured in Table 4-8 is the range of prediction errors that can be assessed with the various predictor variables. Recall that estimate variance is generated using semi-variogram information (see Eq (4-19)). If all of the data points used to generate a kriged estimate lie outside of the semi-variogram's range (e.g. all of the h_{i0} 's $\geq h_{range}$), then each of the $\gamma(h_{i0})$ values will equal the population variance (s^2). Thus, regardless of the weights applied to each data point, the estimate variance, which is a linear combination of the $\gamma(h_{i0})$ values, will always equal s^2 (because

$\sum_{i=1}^n w_i = 1$). Therefore, as a predictor variable, σ_0^2 is not able to discern levels of accuracy once

all of the h_{i0} 's $\geq h_{range}$. Since accurate kriging estimates are attainable despite all h_{i0} 's $\geq h_{range}$, it is necessary to select a predictor that is not limited in this way.

For this reason, in the jet engine optimization problem, the mean distance of the point being estimated to the data points (\bar{h}_0) is used. Although, in Table 4-8, h_{10} actually has a slightly higher correlation to prediction error, it was decided that, in general, \bar{h}_0 would be less sensitive to the data sample locations. While \bar{h}_0 does not appear to have impressive predictive qualities, in practice it is possible to ensure that only response estimates with acceptable estimation errors are treated as valid. This is achieved by selecting conservative values of \bar{h}_0 , which has the unfortunate side-effect of causing some accurate estimates to be rejected as inaccurate.

Determining Appropriate Acceptance Threshold for the Selected Predictor Variable. To determine an appropriate \bar{h}_0 value by which estimates are to be accepted or rejected, \bar{h}_0 is plotted with prediction error magnitude. Using a first-order linear regression, a simple mathematical expression is developed to relate \bar{h}_0 to the expected prediction error magnitude. Based on the application, an acceptable level of prediction error can be selected, and the appropriate \bar{h}_0 threshold value determined using this linear relationship.

Figure 4-7 depicts this regression process and illustrates that setting the \bar{h}_0 acceptance threshold = 0.63 provides rough assurance that kriging prediction errors remain at or below 5 response units. If necessary, this \bar{h}_0 threshold can also be lowered to provide added insurance against prediction error magnitudes greater than 5.

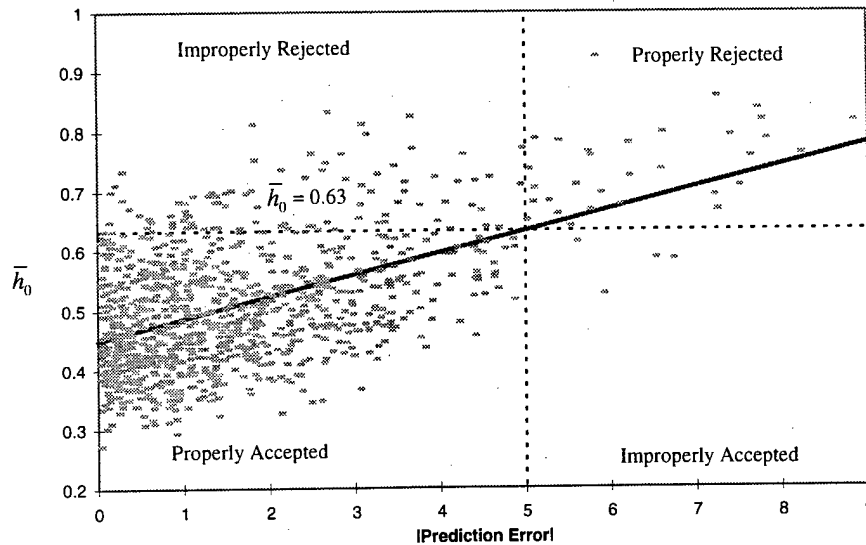


Figure 4-7. Determining \bar{h}_0 Threshold to for a Desired Kriging Prediction Error Magnitude

Notice in Figure 4-7 that all kriged responses with $\bar{h}_0 \leq 0.63$ would have been accepted using this as the acceptance criteria. This includes a small set of responses (about 5 out of the 1000) with |Prediction Error| values greater than the desired 5 units. Also notice that all estimates with $\bar{h}_0 > 0.63$ would have been rejected. This includes a significant number of estimates that actually had prediction errors less than 5 units. Thus, even though a large number of accurate estimates are unnecessarily rejected, more importantly, only a very small fraction are incorrectly accepted. It is important to note that appropriate threshold values for the indicator variable will be specific to each individual application and must be determined (using the technique displayed in Figure 4-7) for each new response surface.

4.5 Determining Feasibility

The kriging algorithm is not capable of predicting infeasibility – that is, portions of the design region that are invalid combinations of design variables. It's geological roots do not require such accommodations. However, in order to use kriging in this and other optimization

applications, a means of predicting infeasibility is imperative. Otherwise, it is possible for the kriging algorithm to return feasible response estimates for points that, if evaluated using the actual response function, would be handled as infeasible responses.

For problems with constraints that can be explicitly modeled, a simple check to see if the design variables lie in the feasible region is all that would be required before invoking the kriging algorithm. However, for many non-linear optimization problems (including this application), constraints that define the feasible/infeasible regions cannot be explicitly represented. The constraints may be implicit to the complex, iterative algorithms that generate the response function values. In these cases, it is necessary to estimate the feasible region so that kriging is only invoked when feasibility is assured.

Feasible Region Checking Algorithm Description. Presumably, the response function being called during an optimization is capable of reporting whether or not the design point being evaluated is feasible or infeasible. If this is the case, it is possible to build a database of feasible and infeasible points as an optimizer moves throughout a design region looking for the optimal solution. It is from this feasible/infeasible database that a feasible region checking algorithm can predict if a new design point is likely to be feasible.

Say that an optimization problem involves p independent design variables, each of which has a defined range of values to be explored by the optimization algorithm. Note that this implies an p -dimensional box-shaped design region, since the range of possible values for any of the design variables is not affected by the values of the other $(p-1)$ variables. Now say that only q of these variables have regions of infeasible values and that r of these variables do not have infeasible values ($q + r = p$). To know that some of the variables do not have regions of infeasibility requires a priori knowledge of the design region. If no information is known, it is always possible to let $q = p$ ($r = 0$) and treat all variables as having regions of infeasibility. For

the purposes of the feasible region checking algorithm, we need only concern ourselves with the q variables with infeasible regions. Keeping track of infeasible values for the r variables that have no infeasible values is simply a waste of resources – these r variables do not affect a point's feasibility.

With the q remaining variables, we will first establish a q -dimensional grid whose grid spacing is defined by the user. For example, if $q = 2$, we would establish a two-dimensional grid that sub-divides the ranges of possible values for both of these variables. If the first of these two variables (q_1) is to be explored for values ranging from -100 to +100 and the second of these variables (q_2) is to be explored for values ranging from +500 to +1000, a possible grid structure could be as shown in Figure 4-8.

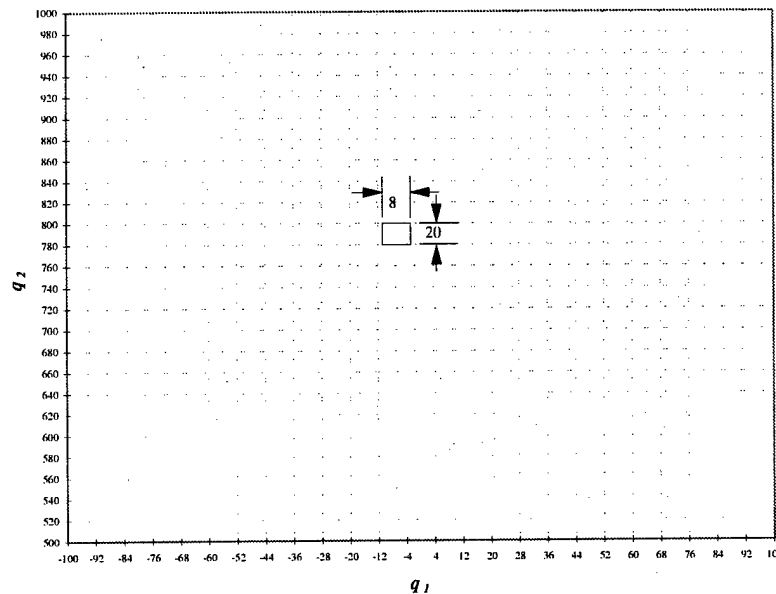


Figure 4-8. Possible Grid Structure for $q = 2$

Note that in this example, the region has been sub-divided 25 times in each dimension, resulting in 25^2 or 625 regional sub-divisions. Also note that the scaling is different on each axis, so these sub-divisions are actually rectangular in shape. The number of sub-divisions for each variable

does not need to be the same; however, it is likely that the range of possible values for each variable was chosen proportionally, and, therefore, the grid resolution needed in each axis would be approximately the same.

Once the grid is defined, the feasible region checking algorithm will take the feasible and infeasible points located by the optimization algorithm and map this information into the grid structure. This mapping is achieved by first determining the grid sub-division in which a point is located (as determined by the point's coordinates), and then placing the feasible/infeasible information associated with the point into the grid. Identifying the grid sub-division in which a point is located is achieved using

$$g_{q_i} = \text{Integer} \left[\left(\frac{(q_i - \min(q_i))}{(\max(q_i) - \min(q_i))} \right) \cdot n_{q_i} \right] \quad (4-21)$$

where

g_{q_i} is the sub-division in the q_i dimension in which the new point is located

q_i is the value of new point's coordinate in the q_i dimension

n_{q_i} is the total number of sub-divisions in the q_i dimension

$\min(q_i)$ is the minimum possible value in the q_i dimension

$\max(q_i)$ is the maximum possible value in the q_i dimension

$\text{Integer} []$ implies taking only the integer portion of the floating point number

that result from the division inside of the brackets. In other words,

round down to the closest integer value.

At this point, the grid structure is more easily represented as a q -dimensional array of numbers. For the jet engine optimization problem, four different numbers were used to indicate four possible information states of a grid sub-division:

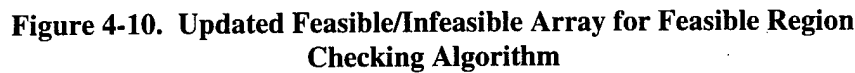
- 0 = No feasible/infeasible information known
- 1 = All observed points in grid sub-division are infeasible
- 2 = Some observed points in grid sub-division are feasible, some are infeasible
- 3 = All observed points in grid sub-division are feasible

When the feasible region checking algorithm is initialized, all members of the q -dimensional array are set to 0 to indicate no information known about the design region. From here, the algorithm processes the database of feasible and infeasible points (located by the optimizer) and updates the array using the following logic:

Table 4-9. Logic Matrix for Feasible Region Checking Algorithm

Current Grid Sub-Division Value	New Database Point is:	
	Feasible	Infeasible
0	3	1
1	2	1
2	2	2
3	3	2
New Grid Sub-Division Value		

Returning to the example introduced in Figure 4-8, let us go through the process of updating this array based on new feasible/infeasible point information. As previously stated, we start with a two-dimensional array of all 0s, indicating no information known about the design space.



	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
	0	1	0	1	1	0	0	1	1	1	1	0	1	1	1	1	0	1	1	0	1	0	0	1
	0	0	0	0	0	1	1	1	2	2	1	1	1	1	1	1	0	0	0	0	1	0	1	
	0	0	0	0	0	1	1	0	3	3	3	2	1	1	2	2	0	1	0	0	1	0	1	
	0	0	0	1	1	1	2	3	3	3	3	3	3	3	2	1	0	1	1	1	0	0	0	
	0	0	1	1	1	2	3	0	3	3	3	3	3	3	2	1	1	1	1	1	1	1	0	
	0	0	0	1	2	3	3	3	3	3	3	3	3	3	3	2	1	0	2	2	1	1	0	
	0	0	1	1	2	3	3	3	3	3	3	3	3	3	2	2	2	3	2	0	1	1	0	
	0	1	0	3	3	3	3	3	0	3	3	3	3	3	3	0	0	3	3	1	0	0	0	
	0	1	1	2	3	3	3	3	3	3	3	3	3	3	3	0	3	3	1	0	1	1	0	
	0	2	3	3	3	3	0	3	3	0	3	3	3	3	3	3	3	3	2	0	0	0	0	
	0	2	3	0	3	3	3	0	3	3	3	3	3	3	3	3	3	3	1	0	0	0	0	
q_2	0	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	0	0	0	
	0	1	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	1	0	0	
	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	3	1	1	0	
	0	1	2	3	3	3	0	3	3	3	3	3	3	3	3	3	3	2	2	0	3	2	1	
	0	1	2	3	0	3	0	3	3	3	3	3	3	3	3	3	3	2	0	0	3	3	0	
	0	1	1	2	2	3	3	3	0	3	3	3	3	3	3	3	3	2	0	0	3	3	2	
	0	1	1	1	0	2	3	3	3	3	3	3	3	3	3	3	3	0	0	3	3	2	1	
	0	0	1	1	1	2	3	3	3	3	3	3	3	3	3	3	3	2	0	0	0	3	2	
	0	0	0	1	1	2	0	0	2	2	2	2	0	1	2	2	0	2	0	3	3	0	1	
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	1	0	
	0	1	0	0	0	1	1	0	1	0	1	1	0	1	1	0	1	1	1	1	1	0	0	
	0	0	0	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	0	0	
	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1	1	1	0	0	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44			

Figure 4-11. Typical Feasible/Infeasible Array Structure After Processing Sampled Points

93

of 2s surrounding the feasible region, indicating a transition zone from feasible to infeasible points.

Using the Feasible Region Checking Algorithm. Once information has been established concerning the shape of the feasible region, it is possible to use this information to predict the feasibility of a new point. Recall that the reason we need a feasible region checking algorithm is to complement the kriging algorithm in making response function estimates without having to evaluate the actual response function (which is presumably time consuming to evaluate).

Conceptually, the feasible region checking algorithm is accessed before any kriging takes place. The new point is passed to the kriging algorithm for evaluation only if the feasible/infeasible array indicates that the point is in a feasible sub-division. That is, a response function estimate for the point is made if and only if the array sub-division in which it lies equals 3, implying that all points that have been evaluated and that lie in this sub-division have been feasible. If it equals any other value, which implies either no information is known about this sub-division or that the sub-division contains at least one known infeasible point, the new point is sent to the actual response function for evaluation. It is important to note that the point is evaluated even if it lies in a grid sub-division that has been labeled infeasible (array value of 1). This ensures that feasible points that share a grid sub-division with infeasible points are not missed. If a feasible point is identified in a grid sub-division previously deemed infeasible, the feasible/infeasible array code will be updated to the not-sure value of 2. Regardless of whether the grid sub-division has a 1 or 2 associated with it, any future points in this sub-division will be sent to the actual objective function for evaluation. Thus, in reality, the feasible region checking algorithm is used only to determine if a point lies in a feasible region – it does not attempt to use any information about the infeasible region in its processing. However, one side-benefit of

keeping track of the infeasible points is the graphical representation of both the feasible and infeasible regions plus the transition area between them (as seen in Figure 4-11).

Note that an array sub-division value of 3 does not guarantee that the point is feasible. It is conceivable that the sub-division contains both feasible and infeasible points, and that, only by chance, no infeasible points been identified. The impact of this phenomenon can be minimized by proper selection of the array grid, but, using this approach, it is not possible to completely eliminate the possibility of mis-classifying an infeasible point as feasible. This issue will be discussed in further detail in the next two sections.

Selecting Grid Resolution. Analyzing Figure 4-11 brings light to the question of choosing the number of divisions to select in each of the q dimensions. If the number of divisions is too large, then the final resolution of the array will be poor. Especially for more irregular feasible regions, much information about the intricacies of the feasible region's shape will be lost if the grid is too coarse, much like the intricacies of a picture are lost when viewed on a television monitor with low pixel resolution. However, to choose the number of divisions too large (very fine resolution) inhibits the usefulness of feasible region checking algorithm.

Because the feasible region checking algorithm will always call the actual response function whenever the feasible array indicates 0, it is counter-productive to make the grid resolution too fine. Imagine a design region that is sub-divided so finely that over the course of an optimization, no two points are in the same grid sub-division. Thus, every time the feasible region checking algorithm is accessed to see if a point is feasible, the feasible array indicates a 0 for no information. This results in the optimizer being directed to the actual response function for evaluation, causing the possible benefits of the estimation algorithm to be short-circuited. For this reason, grid resolution must be chosen carefully, ensuring that it is fine enough to

adequately distinguish between feasible and infeasible regions, but not so fine that the feasible region checking algorithm inhibits the use of the estimation algorithm.

Cautionary Notes on Using the Feasible Region Checking Algorithm. As previously mentioned, one possible short-coming of the feasible region checking algorithm is that it can label a grid square as definitely feasible (3) when in reality it should be labeled not-sure (2). This can occur when the first point sampled from a feasible/infeasible region transition sub-division is feasible.

Say that the optimizer algorithm needs to evaluate a point in a feasible/infeasible transition sub-division. The first time the optimizer attempts to do this, the feasible array indicates a 0 for no information. Thus, the point is evaluated by the actual objective function. If the point being evaluated happens to come from the feasible portion of this transition sub-division, the entire sub-division will be labeled as definitely feasible (3). From this point on, any time the optimizer attempts to evaluate a point in this grid sub-division, it will be assumed to be feasible and the kriging algorithm will be used to estimate the response function value. Assuming that the kriging estimates always meet the acceptance criteria (see section 4.4 for details), no further calls to the response function will ever be made. Thus, it is possible for points that are actually infeasible to be returned to the optimizer as feasible points with response functions.

While the incidence of this situation may seem rather unimportant, it can cause the optimizer to be misled into believing that an infeasible point is the optimum. This is especially true when the optimal solution is located on or near the edge of the feasible region. For this reason, when kriged estimates have been used to locate an optimal solution, it is always advisable to perform a truth-check on the optimal point to ensure that it is feasible. If the optimizer is plagued with convergence to infeasible points, it is recommended that the number of grid sub-

divisions be increased. This will cause more calls to the actual response function to be made, but should help in preventing optimizer convergence to infeasible points.

4.6 Kriging and the Use of Penalty Functions

Aside from providing response estimates for new points, kriging allows for one additional feature to be added to an optimization process. The μ GA, like many non-linear optimizers, is not capable of handling infeasible points by itself. It requires a tangible response value be returned every time an objective function call is made. In some applications, the objective function is programmed to return an infeasible response value to the optimizer that is several orders of magnitude worse than any feasible point response could be. This directs the optimizer away from the region by this point in the future. However, for many optimizers, convergence may be encouraged when an infeasible point is assigned a penalized response value rather than a single, extremely poor response value. By penalizing an infeasible response based on how far away it is from the feasible region, all available information is utilized in assisting the optimizer in moving back towards the feasible region. (Nadon, 1996, 18-20)

If we knew a priori what the feasible region was, when an infeasible point is encountered, it would be best to identify the shortest path to the feasible region, evaluate this feasible point, and use it to generate a pseudo-response for the infeasible point. That is, starting with the response from the feasible point, we can assess a penalty based on how far the infeasible point is from the feasible region. Written as an equation for a minimization problem, the pseudo-response would be generated with:

$$\left(\begin{array}{c} \text{Infeasible Point} \\ \text{Pseudo - Response} \end{array} \right) = \left(\begin{array}{c} \text{Closest Feasible} \\ \text{Point's Response} \end{array} \right) + \left(\begin{array}{c} \text{Penalty, based on Distance from} \\ \text{Infeasible Point to Feasible Region} \end{array} \right) \quad (4-22)$$

As depicted for a single independent variable in Figure 4-12, this in effect, funnels the optimizer back towards the feasible region whenever an infeasible point is evaluated.

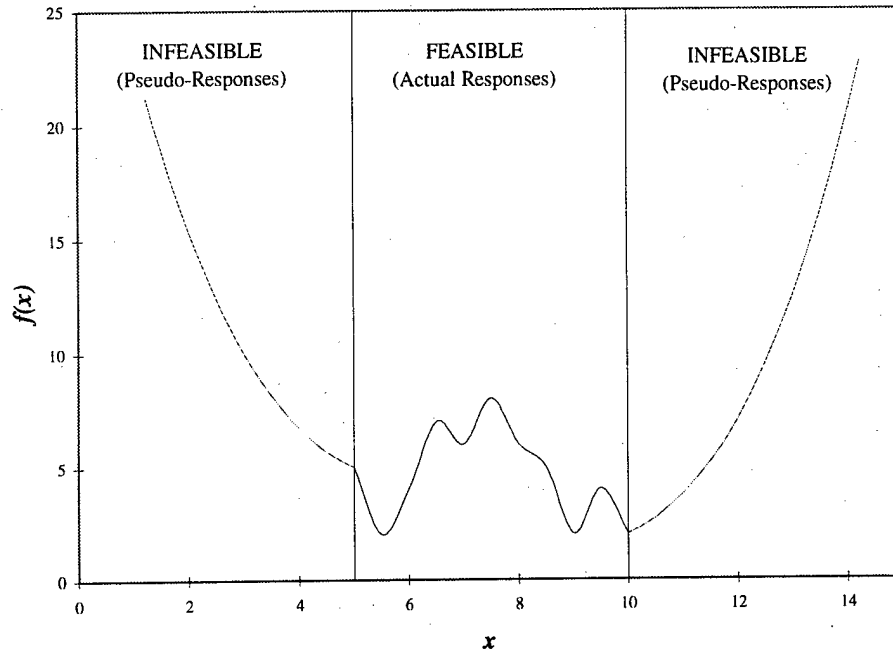


Figure 4-12. Funneling Effect Created by Penalty Functions for a Minimizing Optimization

Assessing Penalty Functions with Incomplete Feasible/Infeasible Region Information. In this problem, the feasible/infeasible boundaries are not explicitly stated – we only have an approximation of it based on the database of feasible points that have previously been located. Thus, we must use this information to best create the same desired funneling effect towards the feasible region.

When working with incomplete feasible region information located in the feasible points database, a method of assessing penalized pseudo-responses for infeasible points is to use the objective function value of the closest known feasible point as the starting value for the penalizing function. One possible way of implementing this approach would be to search the database of feasible points to locate the point that has the shortest Euclidean distance in all p

dimensions from the new, infeasible point. However, if any of the p dimensions has no infeasible values (r is the greater zero), the closest feasible point in the database may not actually be the closest point to the feasible region.

For example, say that an optimization problem with three decision variables exists ($p=3$). In the two dimensions with infeasible regions (along the q_1 and q_2 axes), there exists two points (called Point 1 and Point 2) with slightly different (q_1, q_2) coordinates. Point 1 has q_1 and q_2 values that are slightly within the feasible region (hence the point is feasible); Point 2 has q_1 and q_2 values that are slightly outside of the feasible region (hence the point is infeasible). Note that the value of the third variable (r_1) is irrelevant in determining feasibility for both Point 1 and 2 since all values of r_1 are feasible as long as q_1 and q_2 are feasible.

From an optimization stand-point, we would like to direct optimization away from Point 2 towards Point 1 since it is the direction needed to achieve feasibility. However, even though the Euclidean distance between Point 1 and Point 2 in only the q_1 and q_2 dimensions is practically zero, the values of the third variable (r_1), which has no feasible bounds, could possibly be at opposite extremes of the r_1 range of possible values. Hence, when Euclidean distance is measured in all three dimensions, Point 1 may not be the closest point to Point 2. Instead the closest point could be at some arbitrary location in the feasible region. Pointing the optimizer towards this arbitrary point may get the optimizer back into the feasible region, but may also produce erratic pseudo-response surfaces (response values used by the optimizer in the infeasible regions) that inhibit speedy convergence.

For this reason, it is more desirable to measure Euclidean distance to the feasible region only in terms of the q design variables with infeasible regions. Once the closest feasible point in q dimensions is located, our best guess of the distance to the feasible region (based on the information that we know about the feasible region) is a new, third point with Point 1's q_1 and q_2

coordinates and Point 2's r_1 coordinate. The reason for this is simple – to keep the Euclidean distance to the feasible region in all n dimensions smallest, keep the distance contribution due to the r variables equal to zero. Let the distance used for the penalty function only be defined by the difference in the q variables required to return to feasibility.

To better explain this concept, a simple two dimensional example is useful. In this example, there are a total of two design variables ($p = 2$), one of which has infeasible values ($q = 1$) and one of which does not have infeasible values ($r = 1$). Figure 4-13 shows the actual (unknown) feasible region, the approximate feasible region (as defined by the identified feasible points in the feasible points database), and a new, infeasible point being evaluated.

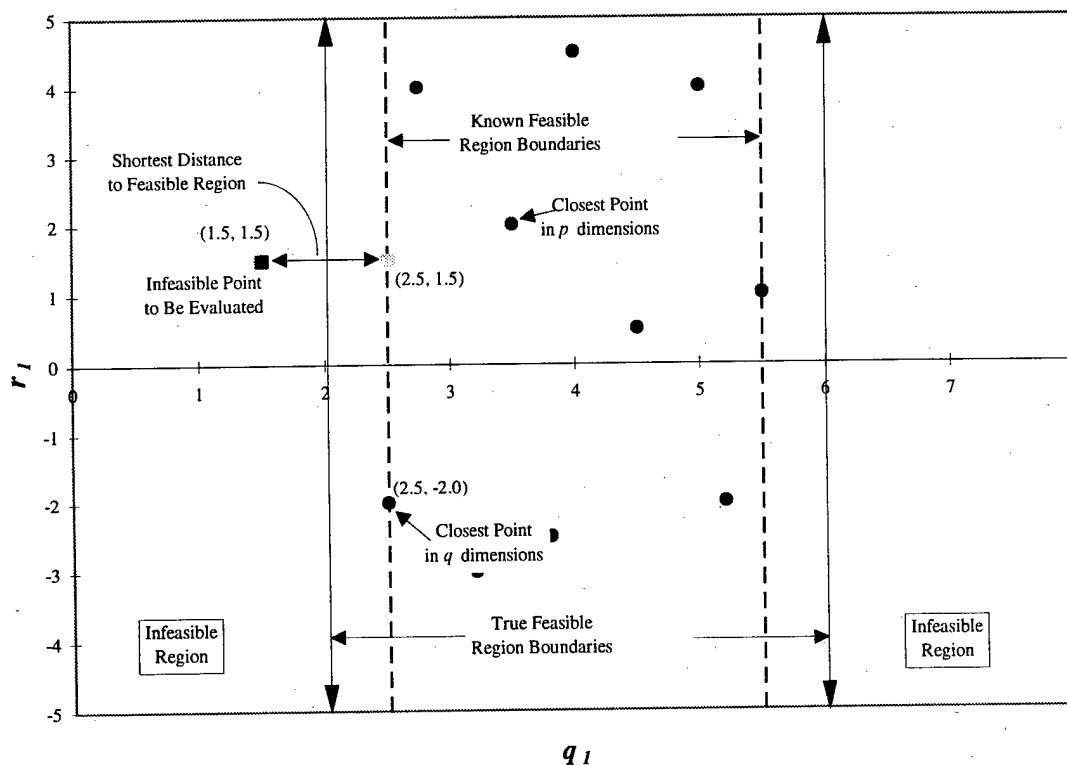


Figure 4-13. Example Determination of Closest Feasible Point

In Figure 4-13, we see depicted how using the closest point in all p dimensions to generate a penalized pseudo-response value for an infeasible point, does not provide the shortest path to the feasible region. Instead, by using the closest point in q dimensions, and then projecting the infeasible point onto the known feasible region boundary, we determine the shortest distance to the feasible region. In practice, this projection simply involves creating a new point (as depicted by the light gray square in Figure 4-13) that has q coordinates from the feasible point, and r coordinates from the original, infeasible point. Note that this point is most likely a point that has not been previously evaluated. We know that it is feasible, but we do not know its response value.

Using Kriging to Generate Infeasible Point Pseudo-Responses. It is here that kriging allows a useful shortcut. Just as this estimation algorithm can avoid costly response function calls during normal feasible point evaluations, it can serve the same purpose for developing a penalized pseudo-response for the infeasible point. From the kriged estimate of the new feasible point, a pseudo-response value can be generated based on the distance from the original infeasible point to the new feasible point. As is the case with normal feasible point kriging, if the kriged estimate does not meet acceptance criteria, the true response function can be called to obtain the base response value. Note that we already know the new point to be feasible, so no checks with the feasible region checking algorithm are required.

4.7 Application of Kriging and the Feasible Region Checking Algorithm to Jet Engine Optimization

For this application, a micro-Genetic Algorithm (μ GA) was selected as the optimization algorithm to locate the combination of three design variables that minimizes overall fuel consumption of an aircraft as it performs a stated mission profile. Of the $p = 3$ design variables, only two of the variables had infeasible ranges of values associated with them ($q = 2, r = 1$). The

response to be optimized was overall fuel consumption. The response function was actually a complex sequence of aircraft mission processing codes linked with a jet engine cycle evaluation algorithm. All constraints that defined the feasible and infeasible regions were implicit to the response function and varied based on the mission profile being optimized.

The impetus for developing the previously discussed estimation techniques was that each call to the response function (utilizing both dimension reduction techniques presented in Chapter 3) took on the order of 60 - 80 seconds. Given the large number of objective function calls required by the μ GA to adequately and consistently converge (on the order of 600 objective function evaluations), computing time to obtain an optimal solution was unacceptably large. Given that the response surface was thought to be reasonably smooth, kriging was selected as a means of estimating objective function evaluations, thus bypassing many calls to the actual response function. Since regions of infeasibility were anticipated in the design region, the feasible region checking algorithm was selected to complement the kriging codes in providing estimated responses. Kriging was also used to generate penalized pseudo-responses for encountered infeasible points (as discussed in section 4.6).

Implementation. Implementation of these response function bypassing techniques was fairly straight-forward. Figure 4-14 provides a graphical representation of how the estimation algorithms were incorporated into the optimization process.

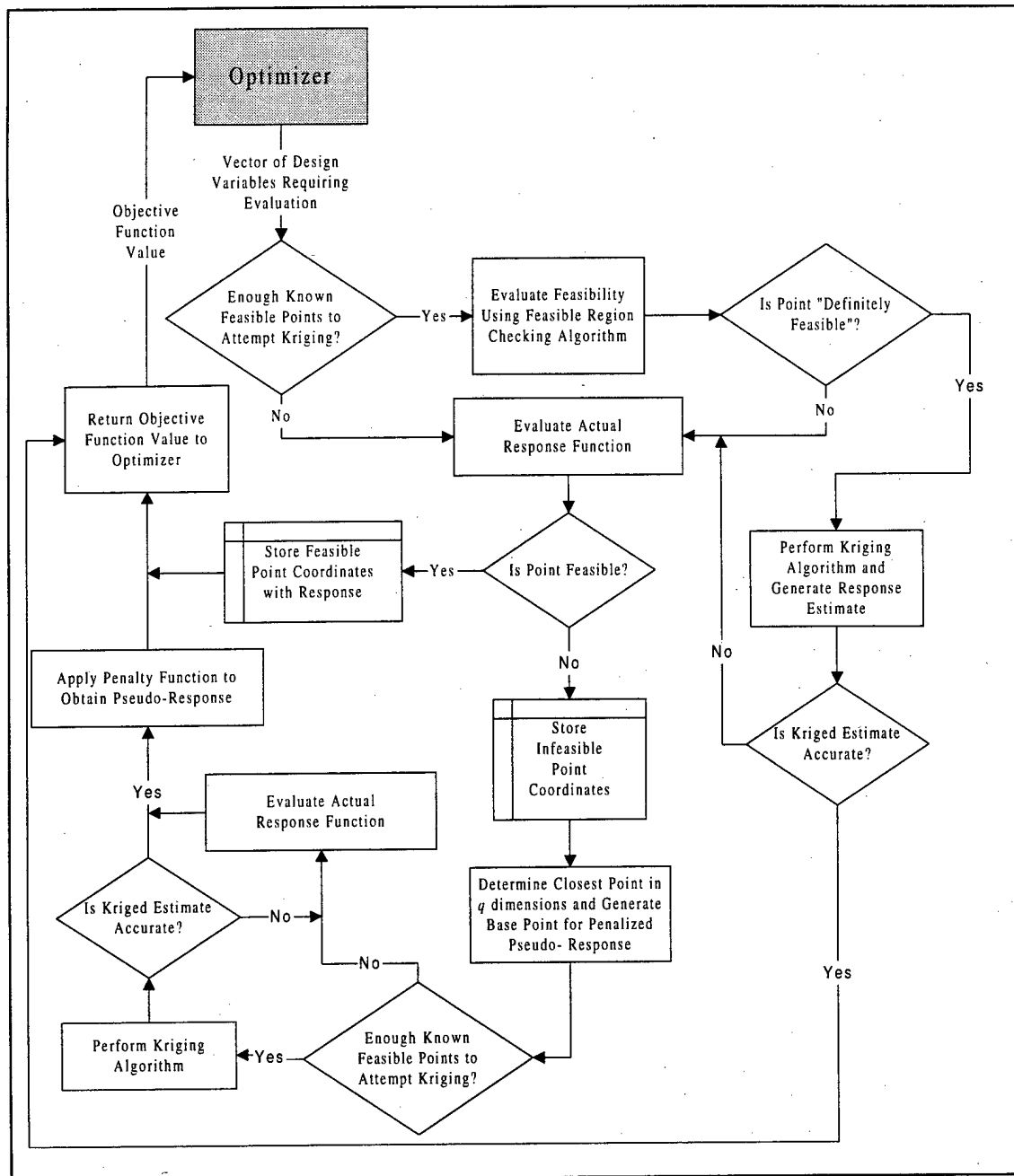


Figure 4-14. Interaction of Optimizer with Response Function and Estimation Algorithms

Once enough feasible points were located for kriging to be performed (recall that the first step of the kriging algorithm is to remove trend using a second-order, least-squares linear

regression, which requires at least $2n + \binom{n}{2} + 1$ data points to solve for this same number of unknowns), the estimation algorithms were attempted whenever an objective function evaluation was needed. If the new point fell in a grid sub-division that was considered definitely feasible (see section 4.5), the kriging algorithm was employed to generate a response estimate. If this response estimate met the kriging acceptance criteria (based on the average distance to the points being used to generate the estimate, \bar{h}_0), this estimated response was passed back to the optimizer for use. However, if either the grid sub-division check or the kriging acceptance criteria check failed, a call to the actual response function was made and the resulting response value passed back to the optimizer. Whenever a call to the actual response function was made, more true information was learned about the response surface and the feasible/infeasible regions. Any such information obtained through objective function calls was stored for later incorporation into the information database used by the kriging and the feasible region checking algorithms.

One of the benefits of using this approach was that the estimates made by the kriging algorithm became more precise and more likely to be accepted as the optimization proceeded. Because the kriging algorithm was able to self-detect the accuracy of its estimates, it was able to assist the optimizer when it could produce good response estimates, and stay out of the way when it did not have enough information to make good estimates. Of course, whenever a kriging attempt was failed, the actual response function was called, thus establishing a known data point in that part of the design region. By establishing another known data point, the kriging algorithm became better equipped to make better estimates in that region if called upon to do so later in the optimization. In practice, the μ GA optimizer quickly established a useful database of feasible points via response function calls, and then was able to reap the processing speed benefits of the kriging estimator.

The same acceptance criteria was applied to all kriged estimates, even if the purpose of the estimate was for creating a penalized pseudo-response for an infeasible point. If the acceptance criteria was failed while attempting to estimate this value, the actual response function was called to obtain the base response from which penalty was applied. While it is conceivable that such an accurate response value was not needed to apply a penalty function (the objective of directing the optimizer back to the feasible region can be met with somewhat inaccurate base responses), it was decided that the most conservative approach was to use only accurate response values.

Configuration. As previously mentioned, this optimization problem had three design variables ($p = 3$), two of which had regions of infeasibility ($q = 2$). Thus a two-dimensional feasible/infeasible grid was needed for the feasible region checking algorithm. Given the range of possible values that these variables could attain and the response sensitivity to variations in these variables, each dimension was sub-divided into 25 sections. As a result, the design region in these two dimensions was divided into 25^2 or 625 sub-divisions (similar to the example discussed in section 4.5).

The kriging algorithm was configured so that mean distance to the data points (\bar{h}_0) was used to determine the quality of the kriged response estimate. The kriged estimate acceptance threshold was set to reject any estimate that was more than 5 counts different than the true response value. Of course, remember from section 4.4 that \bar{h}_0 is not perfectly correlated with the true estimate error. Hence, this acceptance criteria must be thought of as a rough attempt to filter bad estimates from the optimizing algorithm. Regardless of the acceptance criteria threshold value, it is still possible for inaccurate information to be passed to the optimizer (refer to Figure 4-7 in section 4.4 for a graphical representation of why this is true).

Additionally, the kriging algorithm was configured to use the four (4) closest known points (meaning points that had previously been evaluated using the actual response function) to the point being estimated in order to generate the kriged estimate ($n = 4$). The choice to use four points was fairly arbitrarily, although the decision was made with the intuition that choosing too few or too many points could introduce distortion into the kriged estimates. While no exhaustive evaluations were made to find the optimal number of points to use in kriging, four points did, in practice, generate response estimates with acceptable prediction errors.

Performance Enhancements. In the kriging example presented in sections 4.2 through 4.4, all of the design variables were scaled similarly (possible values ranging from 0 to 10). However, in the jet engine optimization problem, scaling for each of the three design variables was widely varied. One of the variables typically ranges from 0 to 8, another from 10 to 100, and another from 5000 to 40000. As is often the case, changes in response were proportional to the percent change in each variable rather than the magnitude of the change in each variable. Creating a semi-variogram (which is heavily dependent on the Euclidean distance between two points) in this environment produced useless results. However, it was discovered that by transforming each of the axes into a coded -1 to +1 scale (a practice common in Response Surface Methodology), useful semi-variograms could be generated. Thus, the kriging algorithm used in this application converts all data into this coded space before performing any estimation. Figure 4-15 depicts how variable coding transforms variables with different scalings into variables with similar scaling.

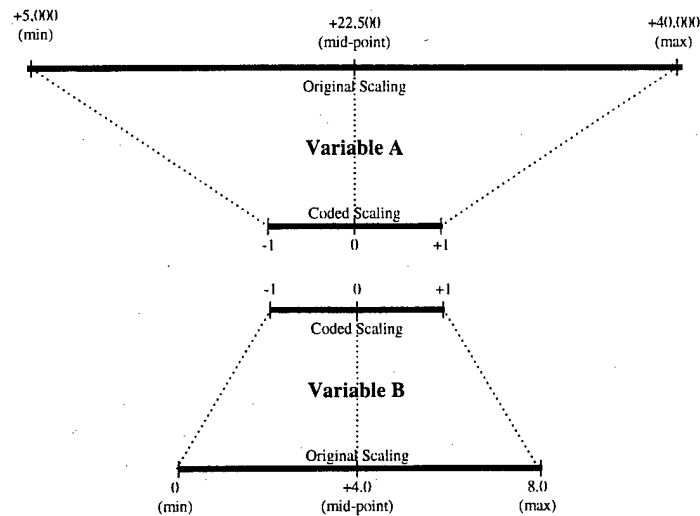


Figure 4-15. Variable Coding

Another enhancement made in this application involved designing the estimation algorithms to operate as efficiently as possible. Through the use of globally defined information, it was possible to code both the kriging and the feasible region checking algorithms to store current semi-variogram and feasible/infeasible array information between estimation routine calls, and only update them when new points had been added to the information database. This significantly enhanced computational efficiency since semi-variogram and feasible/infeasible array generation require a large number of floating point operations. For example, to generate a semi-variogram, the Euclidean distance and the change in response value must be computed for every possible pairing of all feasible points. If there are k feasible points in the database, $\binom{k}{2}$ pairings exist. Since what we are really trying to identify is the slope of the line leading up to semi-variogram's sill, it is possible to update this slope estimate in a running average fashion. Thus, if only one new point had been added to the database since the last time the kriging algorithm was invoked, only the pairings with this new point needed to be generated and the effects incorporated into the current slope estimate. The result was that only k pairings required

computations rather than all $\binom{k+1}{2}$ pairings (of which $\binom{k}{2}$ of these pairings had previously been processed).

Results. Significant reductions in processing time were experienced when the kriging algorithm (in concert with the feasible region checking algorithm) were enabled. The total number of objective function calls required for the μ GA was unaffected by the kriging algorithm's operation. Likewise, the optimal objective function values obtained with the optimizer were also unaffected by the kriging algorithm's operation. In essence, the kriging algorithm did not hinder the optimizer at all; it did, however, allow for much faster objective function evaluations.

Recall that the typical call to the actual response function required (utilizing both dimension reduction techniques presented in Chapter 3) approximately 60 - 80 seconds to return a response value. The kriging algorithm call required only about 1 - 2 seconds to return a response estimate. For a typical optimization run (that is, a start-to-finish convergence of the μ GA optimizer), out of the approximately 600 response evaluations made by the μ GA optimizer and the penalty function calculations, about 80% of the calls were handled by the kriging algorithm. Although it is difficult to get an accurate assessment of processing time reduction (due to variations in computer processor task loading), optimization runs with kriging disabled took on the order of 7 - 11 hours while optimization runs with kriging enabled only took on the order of 1 - 3 hours.

Table 4-10 shows the results of 40 μ GA convergences, each performed on the same mission but started at different random number seeds. 20 of the runs were performed with kriging enabled, 20 were performed with kriging disabled. For the runs performed with kriging, data is provided showing how much the kriging algorithm was used. Run-time data must not be

taken as scientific data, but do indicate a general trend in processing speed that is consistent with the use of the much faster kriging algorithm.

Table 4-10. Jet Engine Optimization Results With Kriging Disabled and Enabled

Kriging Disabled				Kriging Enabled				
Run #	# Response Evaluations to Converge	Time Required to Converge (hours)	Optimal Objective Function Value	Run #	# Response Evaluations to Converge	# Response Evaluations Met By Kriging Algorithm	Time Required to Converge (hours)	Optimal Objective Function Value
1	417	9.28	5471	1	1046	819	2.63	4517
2	556	10.87	4611	2	749	591	1.95	4714
3	810	12.67	4687	3	707	413	4.37	4766
4	570	9.70	4925	4	636	576	0.91	4618
5	599	9.71	4626	5	632	537	1.45	4634
6	790	12.05	5619	6	519	459	0.92	5617
7	440	4.80	5226	7	611	526	1.33	4649
8	456	4.98	4647	8	466	396	1.21	5276
9	621	7.01	4665	9	689	508	3.18	4705
10	762	8.29	4756	10	551	435	1.92	4725
11	993	11.70	4661	11	683	599	1.35	4642
12	495	5.45	4493	12	1065	942	2.08	4559
13	474	4.71	4844	13	818	714	1.71	4626
14	715	7.77	5222	14	460	389	1.21	4947
15	554	5.74	4722	15	631	591	0.68	6078
16	951	10.75	4499	16	449	364	1.26	4621
17	571	6.28	4503	17	526	477	0.84	4861
18	859	11.67	5036	18	721	639	1.44	4773
19	421	6.73	4603	19	556	405	2.64	5023
20	826	14.42	5084	20	524	411	2.05	4842
Mean	644	8.729	4845	Mean	651.95	539.55	1.7565	4859.65
Standard Deviation	180.89	2.96	328.76	Standard Deviation	171.33	151.79	0.90	388.38

Note that when kriging was enabled, the percentage of response evaluations met by the kriging algorithm increased as the run number increased. This was because the runs were done consecutively, thus giving later runs the benefit a more populated database of feasible points. This added benefit that allows for speedy replications of the entire μ GA (a must for stochastic optimization routines) was not available to runs with the kriging algorithm disabled. Essentially, every run with kriging disabled was truly equivalent to starting from scratch. In contrast, with

kriging enabled, fewer and fewer calls to the actual response function were required as the multiple replications were performed.

A relatively broad range of optimal objective function values was experienced in these 40 μ GA optimizations. Typically, if an optimizer is successfully locating the global optimum (instead of local optima), it will always converge to the same point. The dispersion in the optimal solutions in these runs implies that the optimizer was not properly configured to adequately search the highly non-linear response surface in this problem. When a response surface contains numerous local optima (as is the case in this application), the difficulty of locating the true globally optimal point is greatly increased. While an SGA or μ GA optimizer can still robustly locate a global optimum in this situation, further study (that was not performed in this research) is required to fine-tune the optimizer parameters so that the design space is more thoroughly searched. Regardless of this noted short-coming in the optimizer configuration, results for runs performed with and without kriging are still comparable since the μ GA optimizer was configured identically for both sets of runs.

Table 4-10 verifies that, while time required for convergence is much shorter when kriging is enabled, the number of response evaluations and optimal objective function values are essentially identical. By assuming the data to be normally distributed and independent, a t-test (with different population variances) can be performed to confirm our intuition. At the 95% confidence level, the convergence times means were significantly different, and the number of objective function calls and optimal values were not significantly different. t-statistic and p-value (the probability of the null hypothesis being true) results for each of these three categories tested are shown in Table 4-11.

Table 4-11. t-Test Results Comparing Optimization Performance With and Without Kriging

Category	t-Statistic	p-Value
# Response Evaluations to Converge	0.143	0.444
Time Required to Converge	10.071	4.22E-10
Optimal Objective Function Value	0.129	0.449
H_0 : Sample Means Are Equal H_a : Sample Means Are Not Equal		

Conclusions. The use of kriging estimation, complemented by the feasible region checking algorithm, was successful at significantly reducing the computation time required for jet engine optimization to be performed. Using a μ GA optimizer, no negative side-effects from the operation of the kriging algorithm were experienced. Through these favorable results, it is apparent that the response surface implied by the aircraft and flight profile parameters used by the response function is indeed smooth and conducive to the use of kriging for estimation.

While only limited analysis was performed to ensure that poor response estimates were being rejected, initial indications show that no estimates with significant deviation from its true response value were accepted and used by the optimizer. It was found, however, that several kriged estimates were unnecessarily rejected (meaning that the estimate was acceptable but the \bar{h}_0 value indicated a poor estimate). While further refinement of the acceptance threshold would enhance processing, leaving this value at a somewhat conservative value does provide a level of insurance that poor response estimates are not passed to the optimizer.

5. Final Conclusions and Recommendations for Future Research

5.1 Summary of Conclusions

Problem dimension reduction had mixed results. First, fan pressure ratio optimization was successful at not only eliminating one of the independent variables, but also was able to quickly locate nearly the best π_c value for a given bypass and core compression ratio. This ensured that high bypass ratio engines were fully explored and that the optimal solution was a much better estimate of the global optimum of the entire design space. It is recommended that future work on jet engine optimization problems continue to exploit this noted relationship.

In contrast, establishing engine mass flow variable dependency proved to be a poor enhancement to this design process. The m_0 optimization algorithm was plagued with inefficiency and non-optimality. It also was conceptually shown to be incompatible with non-constant installation loss models that may be incorporated in the future. In all, it is recommended that m_0 remain an independent variable to be optimized by the probabilistic optimization algorithm.

It was determined that kriging was highly effective in reducing the computational effort required to obtain an optimal engine solution. Up to 80% of the response evaluations made by the μ GA optimizer and penalty function calculations were handled by the kriging algorithm working in concert with the feasible region checking algorithm. Although computational time is a somewhat unreliable indicator of efficiency, significant reductions in processing time were experienced with kriging enabled. This time reduction was consistent with the significantly reduced number of floating point operations required to generate kriged estimates over evaluating the actual mission response function. No significant reduction in solution quality and

no increase in the number of objective function evaluations were required to obtain the optimal solution were experienced.

Aside from application to jet engine optimization problems, kriging appears to have potential application to any number of non-linear optimization problems. The kriging algorithm can be easily automated and is capable of providing quick estimates, regardless of the function being evaluated. Although care must be given to ensure that the response function is relatively smooth, kriging appears to have potential application to any problem involving computationally expensive response function evaluations. If nothing else, a generic kriging algorithm (like the one located in Appendix A) should be quickly tested for estimation accuracy to see if it could be used as a time-saving technique.

5.2 Recommendations for Future Research

1. Further exploration of the fan pressure ratio optimality phenomenon should be conducted.

Although a rudimentary explanation of how the principle works was presented in this thesis, no engine cycle calculations revealed this principle to be true in general for all mixed-stream turbofan engines.

2. The non-constant installation loss model should be implemented to obtain more accurate mission modeling results. Regardless of the loss model selected, implementation should be reasonably generic given the modular design of the optimization codes. There appears to be no reason why all model codes currently used by Wright Laboratories could not be incorporated in the mission evaluation codes, thus producing higher fidelity optimizations. Care will have to be taken to ensure that kriging remains a viable estimation technique with the implementation of these more complex models.

3. Many more mission profiles should be optimized with kriging enabled. The mission tested was arbitrarily chosen and is not expected to have any special properties that enabled kriging to work successfully. However, more exhaustive testing is required to ensure that successful kriging is possible for jet engine optimization in general.
4. For the jet engine optimization problem, kriging should be compared with other estimation techniques to ensure that it is truly the best choice for bypassing mission evaluations. There is nothing about kriging theory that guarantees it to be the best estimation technique. The fact that it produces a minimum variance response estimate does not imply that some other method of interpolation or extrapolation would not better serve jet engine optimization.
5. As mentioned before, the kriging estimation algorithm should be tested for prediction accuracy in other various applications involving computationally expensive response function evaluations. As was experienced in this application, computational savings could be significant, thus allowing more thorough exploration of the design space.

Appendix A: Kriging and Feasible Region Checking Algorithm

Computer Codes

Computer codes have been written to accomplish the kriging estimation discussed in this thesis. Both the Kriging.m and Check_Feasible.m algorithms are generically coded and can conceivably be used for any application with any number of independent variables. These algorithms are written in the Matlab (version 4.2c) programming language, which was the operating system used for the entire optimization project.

These algorithms may be used freely with the understanding that these codes are not guaranteed to be free from error and that all liability for the accuracy of these codes rests on the user. I would appreciate hearing about any uses of these codes via email at pmillhouse@aol.com.

Ample commenting has been included in the codes, so no additional discussion about the design of the codes is provided in this appendix.

Kriging.m Algorithm:

```
function [KRIGED_RESP, GOOD_EST, SV_DATA] = Kriging(EST_VEC, DATA_MAT, ...
                                                MAXMIN_MAT, CODE_MAT, ...
                                                KRIGE_DATA, SV_DATA)

%-----
%
% FILENAME: Kriging.m
% PURPOSE : To perform automated kriging techniques with the specified purpose
%           of trying to avoid expensive objective function calls. This will
%           take a matrix of collected data and kriging estimates for new points.
%
% CODED BY : Paul Millhouse
% DATE    : 1 March 1998
%
% INPUTS
%
% EST_VEC - Column vector (Px1) of independent variables for which a
%           response estimate is needed (P is the number of independent
%           variables)
%
% DATA_MAT - An (Nx(P+1)) matrix of known data points and their response
%             values. N corresponds to the number of known data points.
%             The first P columns are the independent variable values.
%             The (P+1)st column is the response associated with the
%             P design variables. The P variables must be in the same
%             order as the P variables in EST_VEC.
%
% MAXMIN_MAT - (Px2) matrix of minimum and maximum possible values for the
%              P variables. Structure is as follows:
%
%              [ Var(1)_Min    Var(1)_Max ;
%                Var(2)_Min    Var(2)_Max ;
%                .             .
%                Var(P)_Min    Var(P)_Max ]
%
% CODE_MAT - (Px2) matrix of values to be used for variable coding. The
%            minimum values will correspond to a coded variable value
%            of (-1). The maximum values will correspond for a coded
%            variable value of (+1). Structure is the same as
%            MAXMIN_MAT.
%
% KRIGE_DATA - (3x1) column vector of information needed by the kriging
%              algorithm. Structure is as follows:
%
%              [ NUM_EST_POINTS ;
%                QUALITY_MEASURE;
%                KRIGING_TOL    ]
%
%              where:
%
%              NUM_EST_POINTS - # of known data points to be used in
%                               generating the kriged estimate
%              QUALITY_MEASURE - =0 for mean coded distance to points
%                               used in generating kriged estimate
%                               =1 for kriged estimate variance
%              KRIGING_TOL - Value used to determine if kriged
%                             estimate is valid. If the quality
%                             measure value is less than
%                             KRIGING_TOL, then GOOD_EST = 1.
%                             Otherwise, GOOD_EST = 0.
%
% SV_DATA - A (4x1) column vector of information used to generate the
%            isotropic semi-variogram. Structure is as follows:
%
%            [ SLOPE ;
%              RANGE ;
%              CALC_SV ;
%              DATA_MAT_POINT ]
%
%              where:
```

```

%
%           SLOPE    - Current estimate of the slope of the linear
%                     semi-variogram model. If no estimate
%                     is known, user should be set equal to 0.
%           RANGE    - Current estimate of the coded distance (h) at
%                     which the sill of the linear semi-variogram
%                     is obtained. If no estimate is known, user
%                     should set equal to 0.
%           CALC_SV   - =1 to re-calculate semi-variogram with new data.
%                     =0 to use current values of SLOPE and RANGE and
%                     not update them with new data.
%           DATA_MAT_POINT - Row in DATA_MAT at which new data begins.
%                     Data above this row is assumed to already be
%                     represented in SLOPE and RANGE.
%
% OUTPUTS
%
%   KRIGED_RESP - Estimated response for the given vector of independent
%                 variables
%
%   GOOD_EST    - =1 if the estimate met acceptance criteria
%                 =0 if the estimate did not meet acceptance criteria
%
%   SV_DATA     - Same structure as input vector. SLOPE, RANGE and
%                 DATA_MAT_POINT have been modified (as required) to
%                 reflect data that has been processed in DATA_MAT. These
%                 estimates can be passed to this function next time it is
%                 called to expedite semi-variogram processing (previously
%                 processed data points will not have to be re-processed).
%
% CALLED UNITS
%
%   NONE
%
% GLOBAL DATA
%
%   None
%
% LIMITATIONS
%
%   This algorithm follows the unbiased, minimum variance kriging prediction
%   method outlined in Edward H. Isaaks and R. Mohan Srivastava's text: An
%   Introduction to Applied Geostatistics, and in Isobel Clark's text:
%   Practical Geostatistics. It assumes the semi-variogram to
%   be isotropic (versus anisotropic), which implies that the same sample
%   variance is obtained regardless of the axis from which h is measured,
%   where h is the distance between two points.
%
%   This algorithm assumes a linear (with a sill) mathematical semi-variogram
%   model. That is, semi-variogram data is fit to a sloped line that extends
%   from the semi-variogram origin to the sill value. The sill value is
%   defined as the variance of the residual values (from the sample points
%   after a second-order regression surface has been fit to it). See
%   Clark's text (chapter 5) for details.
%
%   This code can use either kriged estimate variance or the average distance
%   from the point being estimated to the known data points as acceptance
%   criteria (as set by QUALITY_MEASURE). The acceptance threshold is defined
%   by KRIGING_TOL. The number of points used in for an estimate is set with
%   the variable NUM_EST_POINTS. All of these settings are embedded in the
%   KRIGE_DATA input vector.
%
% ADDITIONAL INFO
%
%   None
%
%-----
%
%-----
% GLOBAL CONSTANTS
%-----

```

```

% None

%-----
% MAIN CODE
%-----

% Assign data from input arguments
NUM_EST_POINTS = KRIGE_DATA(1);
QUALITY_MEASURE = KRIGE_DATA(2);
KRIGING_TOL = KRIGE_DATA(3);
SLOPE = SV_DATA(1);
RANGE = SV_DATA(2);
CALC_SV = SV_DATA(3);
DATA_MAT_POINT = SV_DATA(4);

% Define size of DATA_MAT
N = size(DATA_MAT,1);
P = size(DATA_MAT,2) - 1;

% Calculate "P_CHOOSE_2"
P_CHOOSE_2 = P*(P-1)/2;

% Check criteria by which kriging cannot be performed. Exit this function
% if kriging not possible.
if (N < (2*P + P_CHOOSE_2 + 1)) | (N < NUM_EST_POINTS)
    KRIGED_RESP = +inf;
    GOOD_EST = 0;
    return;
end

% Ensure valid input for NUM_EST_POINTS and QUALITY_MEASURE
if NUM_EST_POINTS <= 0
    NUM_EST_POINTS = 5;
end
if (QUALITY_MEASURE ~= 0) & (QUALITY_MEASURE ~= 1)
    QUALITY_MEASURE = 1; % Default to using estimate variance
end

% Ensure valid input for SLOPE, RANGE and DATA_MAT_POINT
if SLOPE < 0
    SLOPE = 0;
end
if RANGE < 0
    RANGE = 0;
end
if (DATA_MAT_POINT < 1) | (DATA_MAT_POINT > (N+1))
    DATA_MAT_POINT = 1;
elseif (DATA_MAT_POINT == (N+1))
    CALC_SV = 0;
end

% Generate CENTER_VEC and HALF_RANGE_VEC
CENTER_VEC = (MAXMIN_MAT(:,2) + MAXMIN_MAT(:,1)) ./ 2;
HALF_RANGE_VEC = (CODE_MAT(:,2) - CODE_MAT(:,1)) ./ 2;

%*****
% Create coded matrix for regression (used to remove trend from the data
% before kriging) and determining closest points in coded space.
%*****
for I = 1:N
    REG_DATA(I,1) = 1; % Column of 1's
    % First order terms
    for J = 1:P
        REG_DATA(I, (J+1)) = (DATA_MAT(I,J) - CENTER_VEC(J,1)) / HALF_RANGE_VEC(J,1);
    end
    COUNT = 0;
    % First order cross-terms
    for J = 1:(P-1)
        for K = (J+1):P

```

```

        COUNT = COUNT + 1;
        REG_DATA(I, (P+1+COUNT)) = REG_DATA(I, (J+1)) * REG_DATA(I, (K+1));
    end
end
% Second order terms
for J = 1:P
    REG_DATA(I, (P+1+COUNT+J)) = REG_DATA(I, (J+1))^2;
end
end

% Create vector of responses to use in regression
RESPONSE_VEC(:,1) = DATA_MAT(:, (P+1));

if CALC_SV % Only if the semi-variogram is to be updated with new information

    % Perform least-squares regression
    REG_COEFFS = inv(REG_DATA.' * REG_DATA) * REG_DATA.' * RESPONSE_VEC;

    % Generate vector of regression prediction residuals (used in creating the
    % semi-variogram)
    for I = 1:N
        RESID_VEC(I,1) = (REG_DATA(I,:)*REG_COEFFS) - RESPONSE_VEC(I,1);
    end

    %*****
    % Now that the vector of response errors has been generated, we are ready to
    % create the semi-variogram. First we must generate the matrix of changes in
    % sample coordinate and the associated changes in response. To facilitate this,
    % a new matrix is first created with the original coordinate data with the
    % newly calculated regression response error appended in the right-most column.
    % We continue to work in coded space so that the different scales of the
    % variables is not a problem.
    %*****
    SV_MAT = REG_DATA(1:N, 2:(P+1));
    for I = 1:N
        SV_MAT(I, (P+1)) = RESID_VEC(I);
    end

    TOTAL_DM = 0;
    DM_COUNTER = 0;
    for I = DATA_MAT_POINT:N
        for J = 1:N
            DIFF_VEC = (SV_MAT(I,1:P) - SV_MAT(J,1:P));
            DISTANCE = sqrt (DIFF_VEC * DIFF_VEC');
            TOTAL_DM = TOTAL_DM + 1;

            %*****
            % Since the Euclidean distances stored in column 1 of DELTA_MAT will
            % later be used to estimate the slope of the semi-variogram line, it is
            % necessary to filter out small values (this value is in the denominator
            % of the slope estimate). Therefore, DELTA_MAT is only updated if
            % the Euclidean distance was bigger than 0.01.
            %*****
            if DISTANCE > 0.01
                DM_COUNTER = DM_COUNTER + 1;
                DELTA_MAT(DM_COUNTER,1) = DISTANCE; % The semi-variogram distance 'h'
                DELTA_MAT(DM_COUNTER, (2)) = (1/2)*(SV_MAT(I,P+1) - SV_MAT(J,P+1))^2;
                % The semi-variogram function value 'gam(h)'
            end
        end
    end

    %*****
    % Now update the semi-variogram math model. For simplicity, the linear
    % model (with a sill) has been used. The least squares estimate of the slope
    % of the line is the average slope implied by each data point. The line
    % is assumed to pass through the origin.
    %*****
    SUM = 0;
    for I = 1:DM_COUNTER
        SLOPE_EST = DELTA_MAT(I,2) ./ DELTA_MAT(I,1);
        SUM = SUM + SLOPE_EST;
    end
end

```

```

NEW_DATA_SLOPE = SUM / DM_COUNTER;

% Obtain variance all sample residuals
VARIANCE = var(RESID_VEC);

%*****
% Identify the (coded) value of h at which the sill is attained. This
% value of h is known as the range of the semi-variogram. The sill is
% the point at which the predicted variance is equal to the sample variance.
%*****
NEW_DATA_RANGE = VARIANCE / NEW_DATA_SLOPE;

%*****
% Calculate previous number of slope and sill estimates and to produce a
% weighted average with the newly calculated estimates
%*****
if DATA_MAT_POINT == 1 % No previous slope estimate to average
    SLOPE = NEW_DATA_SLOPE;
    RANGE = NEW_DATA_RANGE;
else
    DMP_CHOOSE_2 = (DATA_MAT_POINT-1)*(DATA_MAT_POINT-2)/2;
    SLOPE_NUM = DMP_CHOOSE_2*SLOPE + TOTAL_DM*NEW_DATA_SLOPE;
    DENOM = DMP_CHOOSE_2+TOTAL_DM;
    SLOPE = SLOPE_NUM/DENOM;
    RANGE_NUM = DMP_CHOOSE_2*RANGE + TOTAL_DM*NEW_DATA_RANGE;
    RANGE = RANGE_NUM/DENOM;
end
DATA_MAT_POINT = N+1;
SV_DATA = [SLOPE;
            RANGE;
            CALC_SV;
            DATA_MAT_POINT];
else
    VARIANCE = RANGE*SLOPE;
end

%*****
% Determine the NUM_EST_POINTS that are closest (in coded space) to the point
% being estimated (in coded space)
%*****
BEST_MAT = zeros(NUM_EST_POINTS, (P+3));
PREV_BEST_DIST = 0;
EST_VEC_CODE = (EST_VEC - CENTER_VEC)./HALF_RANGE_VEC';
for I = 1:NUM_EST_POINTS
    NEXT_BEST_DIST = +inf;
    NEXT_BEST_ROW = -1;
    for J = 1:N
        DIFF_VEC = (REG_DATA(J,2:(P+1)) - EST_VEC_CODE);
        DISTANCE = sqrt (DIFF_VEC * DIFF_VEC');
        if (DISTANCE < NEXT_BEST_DIST) & (DISTANCE >= PREV_BEST_DIST)
            PREV_PICK = 0;
            for B = 1:NUM_EST_POINTS % Check to make sure points has not previously
                                     % been selected as a closest point
                if J == BEST_MAT(B, (P+3))
                    PREV_PICK = 1;
                    break;
                end
            end
            if PREV_PICK == 0
                NEXT_BEST_DIST = DISTANCE;
                NEXT_BEST_ROW = J;
            end
        end
    end
end

BEST_MAT(I,1:P) = REG_DATA(NEXT_BEST_ROW,2:(P+1));
BEST_MAT(I, (P+1)) = DATA_MAT(NEXT_BEST_ROW, (P+1));
BEST_MAT(I, (P+2)) = NEXT_BEST_DIST;
BEST_MAT(I, (P+3)) = NEXT_BEST_ROW;
PREV_BEST_DIST = NEXT_BEST_DIST;
end

%*****

```

```

% if the acceptance criteria is average coded distance from known points to
% the point being estimated, we have enough information to determine if
% the kriged estimate will be accepted.
%*****
if QUALITY_MEASURE == 0
    if mean(BEST_MAT(:,(P+2))) < KRIGING_TOL
        GOOD_EST = 1;
    else % Estimate deemed "no good" -- exit function
        GOOD_EST = 0;
        KRIGED_RESP = +inf;
        return;
    end
end

%*****
% Now that we have the NUM_EST_POINTS closest points (in coded space), generate
% the matrices required to make kriging calculations. These matrices are
% the [D] matrix, which is actually a (NUM_EST_POINTS + 1) x 1 vector of
% spatial variances between the known points and the point to be estimated
% (the last element of the vector is a one). The [C] matrix is a
% (NUM_EST_POINTS + 1) x (NUM_EST_POINTS + 1) square matrix of the spatial
% variances between all of the points being used for the
% estimation (the last row and column of [C] is 1's, with the bottom, right
% corner element being a "0" -- see Isaaks et al. text for details). We
% will be solving for the vector of weights that will be linearly combined with
% the response values of all of the points to develop the estimate and variance
% of the point to be estimated.
%
% Terms in [C] and [D] are obtained by evaluating the variogram at the
% distance (h), which is the Euclidean distance between the
% two points in question.
%*****
for I = 1:NUM_EST_POINTS
    H = BEST_MAT(I,(P+2));
    if H < RANGE
        D(I,1) = (H / RANGE) * VARIANCE;
    else
        D(I,1) = VARIANCE;
    end
end
D((NUM_EST_POINTS+1),1) = 1.0;

for I = 1:NUM_EST_POINTS
    for J = 1:NUM_EST_POINTS
        DIFF_VEC = (BEST_MAT(I,1:P) - BEST_MAT(J,1:P));
        H = sqrt (DIFF_VEC * DIFF_VEC');
        if H < RANGE
            C(I,J) = (H / RANGE) * VARIANCE;
        else
            C(I,J) = VARIANCE;
        end
        C(J,I) = C(I,J);
    end
    C(I,(NUM_EST_POINTS+1)) = 1.0;
end

for I = 1:NUM_EST_POINTS
    C((NUM_EST_POINTS+1),I) = 1.0;
end

C((NUM_EST_POINTS+1),(NUM_EST_POINTS+1)) = 0.0;

W = inv(C) * D;

%*****
% Calculate the kriging estimate
%*****
KRIGED_RESP = W(1:NUM_EST_POINTS,1)' * BEST_MAT(:,(P+1));

%*****
% if the acceptance criteria is estimate variance, determine if the estimate
% meets the acceptance threshold.
%*****

```

```
if QUALITY_MEASURE == 1
  EST_VAR = W' * D;
  if EST_VAR < KRIGING_TOL
    GOOD_EST = 1;
  else % Estimate deemed "no good"
    GOOD_EST = 0;
    KRIGED_RESP = +inf;
  end
end
end
```


Check_Feasible.m Algorithm:

```
function [FEASIBLE, SUB_DIVS, CHECK_DATA] = Check_Feasible(EST_VEC, ...
    MAXMIN_MAT, FEAS_PTS, INFEAS_PTS,...
    SUB_DIVS, GRID_RES, CHECK_DATA)

%-----
%
% FILENAME: Check_Feasible
% PURPOSE : To map known feasible and infeasible points into a Q-dimensional
%           array in order to predict if new points are likely to be feasible.
%
% CODED BY : Paul Millhouse
% DATE    : 1 March 1998
%
% INPUTS
%
%   EST_VEC      - (Q x 1) column vector of independent variables to be
%                  tested for feasibility. Note that only variables that
%                  have infeasible ranges need to be input in this function.
%                  Variables with no infeasible values should be excluded
%                  to maximize the efficiency of this function. Q is the
%                  number of variables with infeasible values (i.e. the
%                  application has P independent variables, but only Q of
%                  of these P variables have infeasible values -- such that
%                  Q <= P)
%
%   MAXMIN_MAT   - (Px2) matrix of minimum and maximum possible values for
%                  the P variables. Structure is as follows:
%
%                  [ Var(1)_Min    Var(1)_Max ;
%                    Var(2)_Min    Var(2)_Max ;
%                    .             .
%                    Var(P)_Min    Var(P)_Max ]
%
%   FEAS_PTS     - (N x Q) array of known feasible point coordinates that
%                  will be used to establish the feasible region. N is
%                  the number of known feasible points. Q is the
%                  number of variables with infeasible values.
%
%   INFEAS_PTS   - (M x Q) array of known infeasible point coordinates that
%                  will be used to establish the infeasible region. M is
%                  the number of known infeasible points. Q is the
%                  number of variables with infeasible values.
%
%   SUB_DIVS     - The 2-D array that holds the mapping information for
%                  feasible and infeasible points. From this map, new
%                  points can be deemed feasible or infeasible. This
%                  2-D array actually holds the information for all Q
%                  dimensions. This is done because Matlab 4.2c (in which
%                  this was programmed) does not allow for higher dimension
%                  arrays. Nevertheless, this array portrays the design
%                  space sub-divisions and whether current data implies that
%                  a sub-division is feasible, infeasible, "not sure" or
%                  no information.
%
%                  If a SUB_DIVS matrix has not previously been created by
%                  this function, set SUB_DIVS = [].
%
%   GRID_RES     - (Q x 1) column vector of the number of grid sub-divisions
%                  to be applied in each of the Q dimensions. This
%                  information defines the resolution of SUB_DIVS.
%
%   CHECK_DATA   - (2 x 1) column vector of information with format:
%
%                  [ FEAS_POINTER ;
%                    INFEAS_POINTER ]
%
%                  where:
%
%                  FEAS_POINTER - Indicates the FEAS_PTS row to begin
```

```

%
% mapping information into SUB_DIVS.
% Proper management of this value can
% avoid repetitively processing known
% data points that can hinder efficient
% operation.
%
% INFEAS_POINTER - Indicates the INFEAS_PTS row to begin
% mapping information into SUB_DIVS.
%
%
% OUTPUTS
%
% FEASIBLE - (2 x 1) vector of feasibility information:
%           [DEF_FEASIBLE;
%           FEAS_CODE ]
%
% where:
%
% DEF_FEASIBLE - Equals 1 if EST_VEC lies in a
%                sub-division into which only feasible
%                points have been mapped. Otherwise,
%                equals 0.
%
% FEAS_CODE - Feasibility code associated with the sub-
%             division in which EST_VEC is located.
%             Codes are as defined in the LIMITATIONS
%             section below.
%
% SUB_DIVS - Updated feasible/infeasible array of information. This
%            array can be passed into this function on subsequent runs
%            to prevent having to re-process points that have
%            previously been mapped.
%
% CHECK_DATA - Updated FEAS_POINTER and INFEAS_POINTER values. Format
%              same as listed in INPUTS.
%
%
% CALLED UNITS
%
% NONE
%
% GLOBAL DATA
%
% NONE
%
% LIMITATIONS
%
% No assumptions about convexity or holes being in the solution space are
% required for this algorithm to work. This algorithm will sub-divide the
% Q variables axes input based on the numbers of sub-divisions specified
% for each variable. The algorithm will then map the known feasible and
% infeasible point information into SUB_DIVS in order to approximate the
% feasible and infeasible regions. SUB_DIVS will maintain information
% using the following codes:
%
% 0 = NO INFORMATION
% 1 = INFEASIBLE (at least based on current data)
% 2 = NOT SURE
% 3 = FEASIBLE (at least based on current data)
%
% When testing a new point for feasibility, if the test point falls in a
% sub-division with any code other than 3, this function will return a
% FEASIBLE value of 0. This does not necessarily mean the point is
% infeasible, just that it does not fall in a subdivision that is known
% to only contain feasible points. If the new point falls in a sub-division
% with a code of 3, FEASIBLE will be returned equal to 1. Note that this
% is simply an estimation technique and, based on the grid resolution
% defined, the point may or may not actually be feasible. This only states
% that, for the sub-division resolution defined, of the known feasible and
% infeasible points, only feasible points have thus far been found in this
% sub-division.
%
% It is important to note that this algorithm is highly dependent on the

```

```

% resolution of the grid. While making the grid excessively fine would
% help to prevent this algorithm from improperly calling
% an infeasible point feasible, it will also cause many of the feasible
% points to be determined infeasible, simply because there is no "grouping"
% of information from the grid (i.e. if every feasible point fills its own
% grid square, then every future point tested will fall in a grid square that
% has not previously been determined, thus this algorithm will return that
% the point is infeasible).
%
%
% ADDITIONAL INFO
%
% This function is intended to be used repetitively, so that as more
% information is learned about the nature of the feasible and infeasible
% regions, predictions of feasibility will improve. For efficiency,
% the function can avoid re-processing known feasible and infeasible points
% that were previously mapped into SUB_DIVS.
%
%-----
%-----
% GLOBAL CONSTANTS
%-----
%
% NONE
%
%-----
% MAIN CODE
%-----
%*****
% Initialize utility variables and check to ensure that all inputs are logical
%*****
Q = size(MAXMIN_MAT,1); % Determine number of variables

if FEAS_PTS == []
    N = 0;
else
    N = size(FEAS_PTS,1); % Determine number of known feasible points
    if size(FEAS_PTS,2) ~= Q
        error('ERROR: FEAS_PTS not consistent with MAXMIN_MAT');
    end
end

if INFEAS_PTS == []
    M = 0;
else
    M = size(INFEAS_PTS,1); % Determine number of known infeasible points
    if size(INFEAS_PTS,2) ~= Q
        error('ERROR: INFEAS_PTS not consistent with MAXMIN_MAT');
    end
end

if size(CHECK_DATA,1) ~= 2
    error('ERROR: CHECK_DATA input not in proper format');
end

FEAS_POINTER = CHECK_DATA(1,1);
INFEAS_POINTER = CHECK_DATA(2,1);

for I = 1:Q
    if GRID_RES(I,1) <= 0
        GRID_RES(I,1) = 1;
    end
end

MIN_VEC = MAXMIN_MAT(:,1); % Vector of minimum values in SUB_DIVS
RANGE_VEC = MAXMIN_MAT(:,2) - MAXMIN_MAT(:,1); % Vector of value ranges
WIDTHS_VEC = RANGE_VEC./GRID_RES; % Width of each subdivision along each axis

SD_WIDTH = GRID_RES(1,1); % Define width of SUB_DIVS
SD_LENGTH = 1;
for I = 2:size(GRID_RES,1) % Define length of SUB_DIVS

```

```

    SD_LENGTH = SD_LENGTH * GRID_RES(I,1);
end

if (size(EST_VEC,2) == Q) & (size(EST_VEC,1) == 1) % Input is in column format
    EST_VEC = EST_VEC';
end

if (size(EST_VEC,1) ~= Q)
    EST_VEC
    MAXMIN_MAT
    error ('ERROR: EST_VEC not consistent with MAXMIN_MAT');
end

if (FEAS_POINTER > (N+1)) | (FEAS_POINTER <= 0) | (FEAS_POINTER == [])
    FEAS_POINTER = 1;
end

if (INFEAS_POINTER > (M+1)) | (INFEAS_POINTER <= 0) | (INFEAS_POINTER == [])
    INFEAS_POINTER = 1;
end

if (size(SUB_DIVS,2) ~= SD_WIDTH) | (size(SUB_DIVS,1) ~= SD_LENGTH)
    FEAS_POINTER = 1; % SUB_DIVS has changed from previous definition
    INFEAS_POINTER = 1; % Will need to re-initialize
end

%*****
% Create or update SUB_DIVS
%*****
if (FEAS_POINTER == 1) & (INFEAS_POINTER == 1)
    SUB_DIVS = zeros(SD_LENGTH, SD_WIDTH); % Initialize SUB_DIVS to all zeros
end

if N >= FEAS_POINTER % There are new feasible points to map to SUB_DIVS
    for I = FEAS_POINTER:N
        for J = 1:Q
            SD_COORDS(J,1) = floor((FEAS_PTS(I,J) - MIN_VEC(J,1))/WIDTHS_VEC(J,1)) + 1;
            if (SD_COORDS(J,1) < 1) | (SD_COORDS(J,1) > GRID_RES(J,1))
                error ('ERROR: Feasible Point Exceeds MAXMIN_MAT Definition');
            end
            end
            X_COORD = SD_COORDS(1,1);
            if Q == 1 % Only one dimension being mapped
                Y_COORD = 1;
            else % Multiple dimensions being mapped
                Y_COORD = 0;
                for K = 2:Q % Convert multi-dimensions into 2-D
                    if K == 2
                        Y_COORD = Y_COORD + SD_COORDS(2,1);
                    else
                        INC = 1;
                        for L = 2:(K-1)
                            INC = INC * GRID_RES(L,1);
                        end
                        Y_COORD = Y_COORD + ((SD_COORDS(K,1)-1) * INC);
                    end
                end
            end
            if SUB_DIVS(Y_COORD, X_COORD) == 0
                SUB_DIVS(Y_COORD, X_COORD) = 3;
            elseif SUB_DIVS(Y_COORD, X_COORD) == 1
                SUB_DIVS(Y_COORD, X_COORD) = 2;
            end
        end
        FEAS_POINTER = N+1;
    end

if M >= INFEAS_POINTER % There are new feasible points to map to SUB_DIVS
    for I = INFEAS_POINTER:M
        for J = 1:Q
            SD_COORDS(J,1) = floor((INFEAS_PTS(I,J) - MIN_VEC(J,1))/WIDTHS_VEC(J,1)) + 1;
            if (SD_COORDS(J,1) < 1) | (SD_COORDS(J,1) > GRID_RES(J,1))
                error ('ERROR: Infeasible Point Exceeds MAXMIN_MAT Definition');
            end
        end
    end
end

```

```

        end
    end
    X_COORD = SD_COORDS(1,1);
    if Q == 1 % Only one dimension being mapped
        Y_COORD = 1;
    else % Multiple dimensions being mapped
        Y_COORD = 0;
        for K = 2:Q % Convert multi-dimensions into 2-D
            if K == 2
                Y_COORD = Y_COORD + SD_COORDS(2,1);
            else
                INC = 1;
                for L = 2:(K-1)
                    INC = INC * GRID_RES(L,1);
                end
                Y_COORD = Y_COORD + ((SD_COORDS(K,1)-1) * INC);
            end
        end
    end
    if SUB_DIVS(Y_COORD, X_COORD) == 0
        SUB_DIVS(Y_COORD, X_COORD) = 1;
    elseif SUB_DIVS(Y_COORD, X_COORD) == 3
        SUB_DIVS(Y_COORD, X_COORD) = 2;
    end
end
INFEAS_POINTER = M+1;
end

% Update CHECK_DATA to reflect newly processed points
CHECK_DATA = [FEAS_POINTER;
              INFEAS_POINTER];

%*****
% Determine sub-division in which EST_VEC is located
%*****
for J = 1:Q
    SD_COORDS(J,1) = floor((EST_VEC(J,1) - MIN_VEC(J,1))/WIDTHS_VEC(J,1)) + 1;
    if (SD_COORDS(J,1) < 1) | (SD_COORDS(J,1) > GRID_RES(J,1))
        error('ERROR: EST_VEC Exceeds MAXMIN_MAT Definition');
    end
end
X_COORD = SD_COORDS(1,1);
if Q == 1 % Only one dimension being mapped
    Y_COORD = 1;
else % Multiple dimensions being mapped
    Y_COORD = 0;
    for K = 2:Q % Convert multi-dimensions into 2-D
        if K == 2
            Y_COORD = Y_COORD + SD_COORDS(2,1);
        else
            INC = 1;
            for L = 2:(K-1)
                INC = INC * GRID_RES(L,1);
            end
            Y_COORD = Y_COORD + ((SD_COORDS(K,1)-1) * INC);
        end
    end
end
end

%*****
% Assess EST_VEC feasibility and assign EST_VEC's feasible code
%*****
if SUB_DIVS(Y_COORD, X_COORD) == 3
    DEF_FEASIBLE = 1;
else
    DEF_FEASIBLE = 0;
end
FEAS_CODE = SUB_DIVS(Y_COORD, X_COORD);
FEASIBLE = [DEF_FEASIBLE;
            FEAS_CODE];

```

Appendix B: Standard Genetic Algorithm Operation

This appendix provides discussion about the processes of the Standard Genetic Algorithm (SGA). Specifically, the method by which a set of floating point variables is encoded into a string of binary digits (which constitutes the point's genetic chromosome) is discussed. Additionally, the inner-workings of the selection, crossover and mutation processes are presented.

Creating the Chromosome. The SGA performs all of its survival-of-the-fittest operations on strings of binary digits, which are analogous to genetic chromosomes. Very often in optimization, however, the objective function requires floating point variable inputs. It is therefore necessary for the SGA to be able to encode a set of floating point variables into a chromosome for SGA manipulations, and then decode the chromosome back into a set of floating point variables for evaluation in the objective function. This encoding/decoding scheme is simply a modification of standard floating point/binary conversion that is used in all digital processors.

In order to generate a chromosome, it is first necessary to define a numerical *precision* (represented by ϵ) to which all of the variables need to be represented. In other words, we must set the number of decimal places that are significant in the solutions to be obtained. Say, for example, that our optimal set of variables need only be represented to three decimal places. We would therefore set our SGA numerical precision to $\epsilon = 10^{-3}$.

Encoding a set of variables into a chromosome involves converting each individual floating point number into a binary number (using standard digital conversion techniques) and then concatenating each of the binary numbers sequentially to form a single string of binary digits. For example, say that two variables, x_1 and x_2 , have values of 5.000 and 10.000,

respectively, and are to be converted into a chromosome with the defined working precision set to $\varepsilon=10^{-3}$. To convert this (x_1, x_2) point into a chromosome, each variable is divided by ε and the result converted into binary.

$$\frac{5.000}{10^{-3}} = 5000$$

5000 converted to binary:

1001110001000

$$\frac{10.000}{10^{-3}} = 10000$$

10000 converted to binary:

10011100010000

Thus the chromosome is formed by combining the two binary numbers together:

$$\begin{array}{ccc} \underline{(x_1, x_2)} & \Rightarrow & \underline{\text{Chromosome}} \\ (5.000, 10.000) & \Rightarrow & 100111000100010011100010000 \end{array}$$

During genetic algorithm operation, it is necessary to ensure that all chromosomes in a population are the same length. For this reason, we fix the size of the chromosome to the number of bits required to represent the largest possible values of each of the variables. Each variable being explored by the SGA will have a user defined upper bound that the SGA will not exceed when searching for optimal solutions. By sizing the chromosome to the largest possible variable values, we ensure that all smaller values can also be represented in the chromosome's binary structure. While we are sizing the chromosome to the maximum variable values, we store the number of bits allocated to represent each variable. When decoding from the chromosome back into floating point numbers, this will allow us to partition the chromosome bits and identify the portions of the chromosome that correspond to the original variables (the decoding process is presented later in this section).

Returning to our example, say that for this SGA optimization, the range of x_1 values being searched is from 1.000 to 5.000 and the range of x_2 values being searched is from 6.000 to 10.000. Thus, the maximum values of x_1 and x_2 are 5.000 and 10.000, respectively. As was shown in the conversion example above, the chromosome needs to contain 13 bits for x_1 and 14 bits for x_2 (27 total bits) to represent (5.000, 10.000). Sizing the chromosome to these largest variable values ensures that enough bits exist in the chromosome structure in order to represent all possible values of both of these variables. When combinations of smaller values of these variables are experienced by the SGA, leading zeros are inserted to maintain the chromosome length. For example, when converting (1.300, 5.800) into a chromosome:

(1.300, 5.800) \Rightarrow Chromosome

$$\frac{1.300}{10^{-3}} = 1300$$

$$\frac{5.800}{10^{-3}} = 5800$$

1300 converted to binary:

5800 converted to binary:

10100010100

1011010101000

Add Leading Zeros to Maintain Chromosome Length

0010100010100

01011010101000

Thus:

(1.300, 5.800) \Rightarrow 001010001010001011010101000

Decoding from a chromosome back into floating point numbers is simply the reverse of this process. Because the number of bits representing each variable are fixed and because the chromosome has been pieced together in the same order as the original floating point variables, we simply partition the chromosome appropriately and perform binary to floating point

conversion. For example, say that the SGA has created the following chromosome for evaluation in the objective function:

$$011110101010001111000000100 \Rightarrow (x_1, x_2)$$

Using the known number of binary digits being used for each variable (13 bits for x_1 and 14 bits for x_2), we partition the chromosome and convert each section of the chromosome back into floating point numbers:

0111101010100 01111000000100	
0111101010100	01111000000100
converted to decimal:	converted to decimal:
3924	7684
3924 · 10 ⁻³ = 3.924	7684 · 10 ⁻³ = 7.684

Thus:

$$011110101010001111000000100 \Rightarrow (3.924, 7.684)$$

Genetic Operations. With an understanding of how the chromosome structures are created, we now turn our attention to the genetic operations that are used in the Standard Genetic Algorithm. Selection, crossover and mutation are the genetic operations used to generate various chromosome bit patterns (which correspond to points throughout the design space) and enable the genetic algorithm to locate globally optimal solutions.

Selection. Selection is the process by which the most-fit (the best) members of a population are chosen for mating (crossover). Selection enforces the survival-of-the-fittest

concept in the SGA. Once a population has been evaluated (via the objective function) for fitness, the selection process ensures that the chromosome bit structures with the best fitness values are used to create the next population of chromosomes. While various selection methods exist, they all ensure that the most fit members of a population are most likely to be selected for crossover. Often times, *elitism* is introduced into the selection process. Elitism is a rigid enforcement of survival-of-the-fittest in that it does not leave selection of the most-fit population member to chance. It guarantees that the best member of the population will be replicated into the subsequent population and that this best member will be used for crossover with other chromosome patterns chosen in the selection process. This process ensures that the most-fit member of a current population is at least as fit as the most-fit member of any previous population.

Crossover. Crossover causes new regions of the design space to be explored by creating new chromosome structures from existing chromosome structures. Many techniques can be used to perform crossover, but they all involve a blending of chromosome bit patterns from two parent chromosomes (chosen during selection) to create new children chromosome patterns. A common crossover technique is called *simple crossover*, which is demonstrated below.

Two parents are chosen for mating (crossover):

Parent #1	001010001010001011010101000
Parent #2	011110101010001111000000100

Both chromosomes have the same number of total bits (27 bits). A break-point is randomly selected in the chromosome bit structure. For example, a break at bit 6 could be randomly chosen as shown:

Parent #1	001010 001010001011010101000
Parent #2	011110 101010001111000000100

Crossover is performed by swapping the parents' bits to the right of the break-point, thus producing two new children chromosome patterns:

Parent #1	001010 001010001011010101000
	↑↓
Parent #2	011110 101010001111000000100

produces

Child #1	001010101010001111000000100
Child #2	011110001010001011010101000

Looking at the parents and children in terms of the floating point coordinates that their chromosomes represent, we see how crossover causes new regions of a design space to be searched.

Parent #1 = (1.300, 5.800)	create	Child #1 = (1.364, 7.684)
Parent #2 = (3.924, 7.684)		Child #2 = (3.860, 5.800)

Mutation. As the SGA performs selection and crossover, subsequent populations begin to have more homogenous chromosome bit structures. This is because the best members of a population are being selected for crossover, and after processing several

populations, all of the best population members will be located in the region surrounding the current most-fit solution. In order to inhibit premature convergence to a point that may not be the global optimum, the SGA uses mutation to introduce new chromosome patterns into the population. This causes new points in the design space to be tested for fitness and allows the SGA to search beyond the local optimum it has located.

A common mutation technique, known as *binary mutation*, causes bits in population members' chromosome patterns to flip with some random probability. For example, given a probability of mutation of 0.05, a chromosome undergoes mutation by picking a uniform random number (between 0 and 1) for each bit in the chromosome. If the random number associated with a bit is less than 0.05, then the bit is flipped from 0 to 1 or from 1 to 0. Thus, on average, about 5% of the bits will be randomly flipped via the mutation process. The mutation process is demonstrated below:

Initial Chromosome Structure 001010001010001011010101000

Randomly select bits to mutate:

↓↓ ↓↓
 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0

Final Chromosome Structure 00 0 0100010100 01 1 11010101000

Assessing the affect of this mutation in terms of floating point coordinates, we see how mutation causes new regions of the design space to be searched:

Initial = (1.300, 5.800)

mutates to

Final = (0.276, 7.848)

Bibliography

- Branham, Richard. United States Air Force Officer and Air Force Institute of Technology Student, Wright-Patterson Air Force Base OH. Personal interview. May 1997.
- Clark, Isobel. *Practical Geostatistics*. Barking, Essex, England: Elsevier Applied Science Publishers Ltd, 1987.
- Hill, Philip G. and Peterson, Carl R. *Mechanics and Thermodynamics of Propulsion*. Reading MA: Addison-Wesley Publishing Company, 1970.
- Isaaks, Edward H. and Srivastava, R. Mohan. *An Introduction to Applied Geostatistics*. New York: Oxford University Press, 1989.
- Krishnakumar, Kalmanje. "Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization," *SPIE Vol. 1196 Intelligent Control and Adaptive Systems*: 289-296 (1989).
- Mattingly, Jack D. And others. *Aircraft Engine Design*. New York: American Institute of Aeronautics and Astronautics, 1987.
- Mattingly, Jack D. *On-Design and Off-Design Aircraft Engine Cycle Analysis Programs*. Washington D.C.: American Institute of Aeronautics and Astronautics, 1990.
- Mattingly, Jack D. *Elements of Gas Turbine Propulsion*. New York: McGraw-Hill, Inc., 1996.
- Nadon, Luc J.J.P. *Multidisciplinary and Multiobjective Optimization in Conceptual Design for Mixed-Stream Turbofan Engines*. MS thesis, AFIT/GAE/ENY/96D-6. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1996.
- Pirlot, Marc. "General Local Search Methods," *European Journal of Operations Research*, 92: 493-511 (1996).

Wright Laboratories. *Turbine Engine Reverse Modeling Aid Program (TERMAP)*. Provided by
Wright Laboratories / POTA. Wright-Patterson Air Force Base OH, March 1997.

Vita

Captain Paul T. Millhouse was born on 4 November 1970 in Charleston, South Carolina. He graduated from Middleton High School in 1987 and entered undergraduate studies at Auburn University in Auburn, Alabama. He graduated with a Bachelor of Science degree in Aerospace Engineering in August 1991. He received his commission on 31 August 1991 and entered military service in July 1992.

His first assignment was at Falcon AFB, located in Colorado Springs, Colorado. While there, he served as a satellite operations officer and chief of operations for the Defense Satellite Communications System, Phase III (DSCS III) satellite program. In September, 1996, he entered the Operations Research graduate program at the School of Engineering, Air Force Institute of Technology. Upon completion of his Master of Science degree, he will be assigned to the National Reconnaissance Office, located in the Washington D.C. metropolitan area.

Paul was married to his wife, Kimberly B. Millhouse, in November 1991. They currently have two children, Clayton and Cooper.

Permanent Address: 55 Mueller Drive
Charleston, SC 29407

email: pmillhouse@aol.com

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE Improving Algorithmic Efficiency Of Aircraft Engine Design For Optimal Mission Performance		5. FUNDING NUMBERS		
6. AUTHOR(S) Paul T. Millhouse, Capt, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, 2950 P Street WPAFB OH 45433-7765		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GOR/ENY/98M-02		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Mr. Glenn Blevins PRTA, Bldg #18 1950 5 th Street WPAFB OH 45433		10. SPONSORING / MONITORING AGENCY REPORT NUMBER Phone: (937)255-2121		
11. SUPPLEMENTARY NOTES Lt Col Stuart Kramer				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited		12b. DISTRIBUTION CODE A		
13. ABSTRACT (Maximum 200 words) Automated techniques for selecting jet engines that minimize overall fuel consumption for a given aircraft mission have recently been developed. However, the current techniques lack the efficiency required by Wright Laboratories. Two noted dependencies between turbine engine fan pressure ratio, bypass ratio, high pressure compressor pressure ratio and overall engine mass flow allows for a reduction in the number of independent design variables searched in the optimization process. Additionally, through the use of spatial statistics (specifically kriging estimation), it is possible to significantly reduce the number of time-consuming response function evaluations required to obtain an optimal combination of engine parameters. A micro-Genetic Algorithm (μ GA) is employed to perform the non-linear optimization process with these two computation-saving techniques. Optimal engine solutions were obtained in 25% of the time required by previous automated search algorithms.				
14. SUBJECT TERMS Jet Engine Optimization, Kriging, Spatial Statistics, Black-Box Optimization, Feasible Region Estimation, Problem Dimension Reduction, Estimation, Micro-Genetic Algorithm, Genetic Algorithm, Estimation Accuracy, Jet Engine Component Optimization, Non-Linear Optimization, Implicit Constraints			15. NUMBER OF PAGES 154	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	