3-2007

Surrogate Strategies for Computationally Expensive Optimization Problems with CPU-Time Correlated Functions

Raymond Magallanez Jr.

SURROGATE STRATEGIES FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS WITH CPU-TIME CORRELATED FUNCTIONS

THESIS

Raymond Magallanez Jr., Captain, USAF

AFIT/GOR/ENC/07-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENC/07-01

SURROGATE STRATEGIES FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS WITH CPU-TIME CORRELATED FUNCTIONS

THESIS

Raymond Magallanez Jr.
Captain, USAF

AFIT/GOR/ENC/07-01

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense or the United States Government.

AFIT/GOR/ENC/07-01

SURROGATE STRATEGIES FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS WITH CPU-TIME CORRELATED FUNCTIONS

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Raymond Magallanez Jr., B.S.

Captain, USAF

March 2007

AFIT/GOR/ENC/07-01

SURROGATE STRATEGIES FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS WITH CPU-TIME CORRELATED FUNCTIONS

Raymond Magallanez Jr., B.S.

Captain, USAF

Approved:

Lt. Col. Mark A. Abramson, Ph.D. Chairman	Date
James W. Chrissis, Ph.D. Member	Date
Marcus B. Perry, Ph.D. Member	Date

AFIT/GOR/ENC/07-01

Abstract

This research focuses on numerically solving a class of computationally expensive optimization problems that possesses a unique characteristic: as the optimal solution is approached, the computational time required to compute an objective function value decreases. This is motivated by an application in which each objective function evaluation requires both a numerical fluid dynamics simulation and an image registration and comparison process. The goal is to find the parameters of a predetermined image by comparing the flow dynamics from the numerical simulation and the predetermined image through the image comparison process. The generalized pattern search and mesh adaptive direct search methods were applied in a way that employs surrogate functions in the search step to reduce the number of costly function evaluations. The surrogate functions are formed, based on either previous function values or their computational times, or both. The solution to the surrogate optimization problem can be solved easily and provides an improved solution quickly. A time cut-off parameter was also added to the objective function to allow its termination during the comparison process if the computational time exceeds a specified threshold. The approach was tested on two problems using the NOMADm and DACE MATLAB® software packages, and results are presented.

Dedication

I dedicate this thesis to my entire family for their constant support throughout my life. I especially give thanks to my wife for her patience and strength, along with the wonderful announcement that we are going to have our first child.

Acknowledgements

I would like to thank the members of my thesis committee, Lt. Col. Abramson, James W. Chrissis, and Marcus B. Perry, for their efforts in accomplishing this thesis. Lt. Col. Abramson's expertise and knowledge provided an exceptional foundation to learn from. I would also like to thank those at Los Alamos National Laboratory, Thomas J. Asaki and Matthew J. Sottile who helped and supported this thesis. Their diligent effort was not unnoticed.

Raymond Magallanez Jr.

Table of Contents

List of Figures

List of Tables

List of Symbols

Symbol Page

SURROGATE STRATEGIES FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS WITH CPU-TIME CORRELATED FUNCTIONS

1. Introduction

The optimization problem considered in this research is to evaluate,

$$\min_{x \in \Omega} f(x), \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$ is computationally-expensive, $\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ and $l, u \in (\mathbb{R} \cup \{\pm\infty\})^n$ for $l < u$. The objective function f will be treated as a "black box" where f can contain different properties to include nonsmoothness, discontinuity, unknown derivatives, and may fail to return a value for $x \in \Omega$. Furthermore, the objective function also possesses a unique property: the computational time required to compute the objective function decreases as objective function values decrease.

1.1 Motivation

This class of problems is motivated by an application in which a single function evaluation consists of a numerical simulation and image registration process. The two processes estimate parameters of experimental data through the numerical simulation of fluid dynamics and then compare two images through a metric measuring image registration. This chapter reviews these processes and reveals why the central processing unit (CPU) time decreases as the solution is approached.

1.1.1 Numerical Simulation of Fluid Dynamics

A numerical simulation involves several steps in an attempt to predict the state of all points in space continuously throughout time. From the observation of a process, a mathematical set of equations can be developed to describe the outcome of the process. The equations can then be solved approximately at a finite number of points, or discretized [17], at any given time. Furthermore, from these sets of known equations one can compile sufficient information to forecast the outcome of an unknown situation.

Fluids, both liquids and gases, are defined as substances that cannot resist shear stress when at rest [17]. The movement of fluids in a given region $\Omega \subset \mathbb{R}^n, n \in \{2, 3\}$ is governed by the well-known Navier-Stokes equations, which are comprised of three equations, namely, momentum of the fluid (1.2), conservation of energy (1.3), and continuity (1.4), given as follows (see Griebel *et al.* [17]):

$$\frac{\partial}{\partial t}\vec{u} + (\vec{u} \cdot \nabla)\vec{u} + \nabla\, p = \frac{1}{Re}\Delta\vec{u} + (1 - \beta T)\vec{g}, \tag{1.2}$$

$$\frac{\partial T}{\partial t} + \vec{u} \cdot \nabla\, T = \frac{1}{Re}\frac{1}{Pr}\Delta T + q''', \tag{1.3}$$

$$div\ \vec{u} = 0, \tag{1.4}$$

where $\vec{u} \in \mathbb{R}^n$ is the velocity field, $p \in \mathbb{R}$ is the pressure in the region, $\vec{g} \in \mathbb{R}^n$ indicates body forces, $Re \in \mathbb{R}$ is the Reynolds number of the flow, $Pr \in \mathbb{R}$ is Prandtl number of the flow, $\beta \in \mathbb{R}$ is the coefficient of thermal expansion, q''' is the heat source, and T is the temperature.

The Navier-Stokes equations cannot be solved analytically. To solve numerically, both space and time are discretized, and a finite differences scheme is applied. For a two-dimensional case, let $\vec{x} = (x_{i,j}^{(k)}, y_{i,j}^{(k)})$, $\vec{u} = (u_{i,j}^{(k)}, v_{i,j}^{(k)})$ and $\vec{g} = (g_x, g_y)$ where $i = 1, \ldots, m$ and $j = 1, \ldots, m$ represent m^2 spacial points in the region at iteration

k. The time increment is denoted by δt, and x and y are spatially incremented by δx and δy respectively. The Laplace operator is denoted by

$$\triangle f = \sum_{i=1}^{d} \frac{\partial^2 f}{\partial x_i^2}.$$

Using central differences and first-order difference quotients, the discretization of (1.2)–(1.4) yields the equations:

$$F_{i,j}^{(k)} = u_{i,j} + \delta t \left(\frac{\triangle u_{i,j}}{Re} - \left[\frac{\partial (u^2)}{\partial x} \right]_{i,j} - \left[\frac{\partial \left(uv \right)}{\partial y} \right]_{i,j} + g_x \right) \tag{1.5}$$

$$G_{i,j}^{(k)} = v_{i,j} + \delta t \left(\frac{\triangle v_{i,j}}{Re} \left[\frac{\partial (uv)}{\partial x} \right]_{i,j} - \left[\frac{\partial (v^2)}{\partial y} \right]_{i,j} + g_y \right) \tag{1.6}$$

$$\tilde{F}_{i,j}^{(k)} = F_{i,j}^{(k)} - \beta \frac{\delta t}{2} \left(T_{i,j}^{(n+1)} + T_{(i+1),j}^{(n+1)} \right) g_x \tag{1.7}$$

$$\tilde{G}_{i,j}^{(k)} = G_{i,j}^{(k)} - \beta \frac{\delta t}{2} \left(T_{i,j}^{(n+1)} + T_{i,(j+1)}^{(n+1)} \right) g_y \tag{1.8}$$

$$u_{i,j}^{(k+1)} = \tilde{F}_{i,j}^{(k)} - \frac{\delta t}{\delta x} \left(p_{(i+1),j}^{(k+1)} - p_{i,j}^{(k+1)} \right) \tag{1.9}$$

$$v_{i,j}^{(k+1)} = \tilde{G}_{i,j}^{(k)} - \frac{\delta t}{\delta y} \left(p_{i,(j+1)}^{(k+1)} - p_{i,j}^{(k+1)} \right) \tag{1.10}$$

$$\begin{aligned} & \left[\frac{\partial T}{\partial t} \right]_{i,j}^{(n+1)} + \left[\frac{\partial (uT)}{\partial x} \right]_{i,j}^{n} + \left[\frac{\partial (vT)}{\partial y} \right]_{i,j}^{n} \\ = \; & \frac{1}{Re} \frac{1}{Pr} \left(\left[\frac{\partial^2 T}{\partial x^2} \right]_{i,j}^{n} + \left[\frac{\partial^2 T}{\partial y^2} \right]_{i,j}^{n} \right) + q_{i,j}''' \\ & \frac{p_{(i+1),j}^{(k+1)} - 2p_{i,j}^{(k+1)} + p_{(i-1),j}^{(k+1)}}{(\delta x)^2} + \frac{p_{i,(j+1)}^{(k+1)} - 2p_{i,j}^{(k+1)} + p_{i,(j-1)}^{(k+1)}}{(\delta y)^2} \end{aligned} \tag{1.11}$$

$$= \frac{1}{\delta t} \left(\frac{F_{i,j}^{(k)} - F_{(i-1),j}^{(k)}}{\delta x} + \frac{G_{i,j}^{(k)} - G_{i,(j-1)}^{(k)}}{\delta y} \right) \tag{1.12}$$

Equations (1.11)–(1.12) lead to a numerical algorithm for simulating fluid flow under different settings. The time step is chosen consistent with Griebel *et al.* [17]

$$\delta t = \tau \ \min\left(\frac{Re}{2}\left(\frac{1}{\delta x^2}+\frac{1}{\delta y^2}\right)^{-1}, \frac{PrRe}{2}\left(\frac{1}{\delta x^2}+\frac{1}{\delta y^2}\right)^{-1}, \frac{\delta x}{|u_{max|}}, \frac{\delta y}{|v_{max}|}\right) \quad (1.13)$$

where $\tau \in [0,1]$ to maintain stability and prevent oscillations. The resulting numerical algorithm can be seen in Figure 1.1, followed by Figure 1.2 representing different flow properties extrapolated from the algorithm under certain parameters and conditions.

1.1.2 Image Registration and Comparison

Image registration is the transformation of an image into a related image [24], while image comparison is a measurement of the transformation to determine the level of similarity between images [6]. Modersitzki [24] shows different types of geometric transformations possible, to include both parametric and non-parametric image registration. Consider the inner product space of squared Lebesque-integrable functions,

$$L_2(\Omega) = \{f : \Omega \to \mathbb{R} | \int_\Omega |f(x)|^2 dx < \infty\},$$

with the inner product $\langle \cdot\, , \cdot \rangle$ defined by

$$\langle f, g \rangle_{L_2(\Omega)} = \int_\Omega f(x)g(x)dx, \quad (1.14)$$

and consider the transformation of an image T,

$$T_u(x) = T(x - u(x)),$$

• INITIALIZATION: Let $t = 0$, $k = 0$, choose δt according to (1.13), and assign initial values to t_{end}, u, v, p, T.
• WHILE $t < t_{end}$
Compute $T_{i,j}^{(k+1)}$ according to (1.11).
Compute $F_{i,j}^{(k)}$ and $G_{i,j}^{(k)}$ according to (1.5) and (1.6).
Compute $\tilde{F}_{i,j}^{(k)}$ and $\tilde{G}_{i,j}^{(k)}$ according to (1.7) and (1.8).
Solve simultaneously $p_{i,j}^{(k+1)}$ according to (1.12) for all i, j.
Compute $u_{i,j}^{(k+1)}$ and $v_{i,j}^{(k+1)}$ according to (1.9) and (1.10).
$t = t + \delta t$
$k = k + 1$

Figure 1.1 Numerical Simulation Algorithm

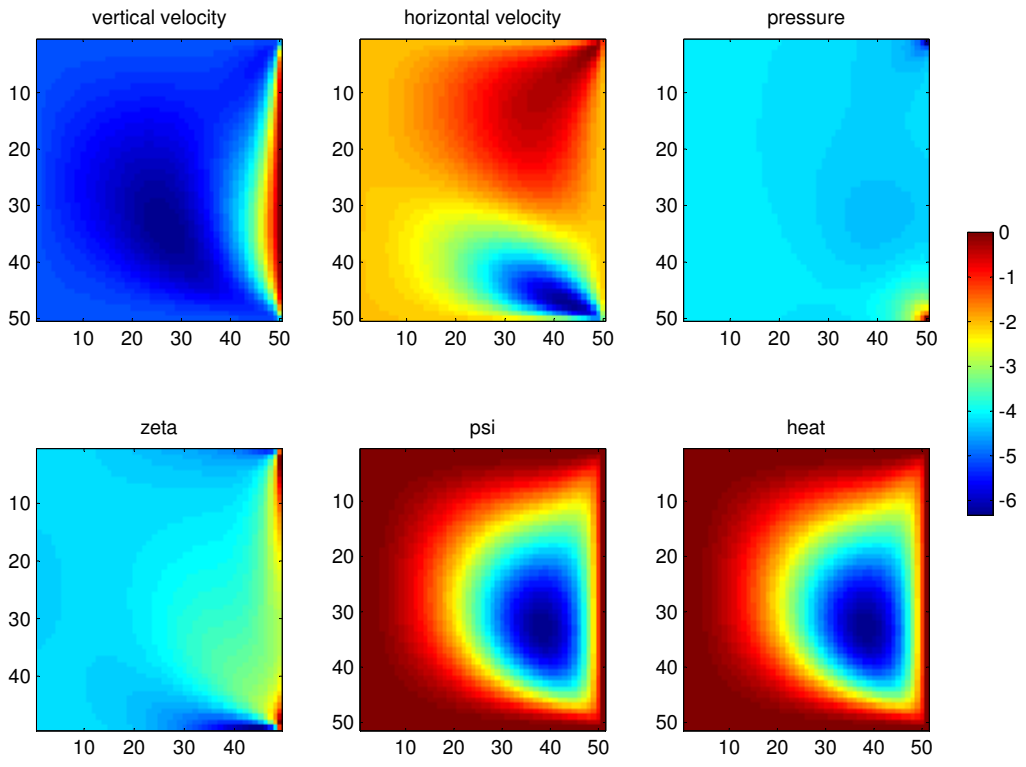

Figure 1.2 Numerical Simulation Image

where $u(x)$ is the displacement of the point x. The objective is to minimize the distance between a *reference image* R and a *template image* T through an optimal warp transformation T_u, as defined by some distance measurement D, and a smoothing or regularizing term S. This problem is given by

$$\min_u D[R, T_u] \quad + \quad \alpha S[u], \tag{1.15}$$

where $\alpha > 0$, and

$$D[R, T_u] \quad = \quad f(x, u(x)) \tag{1.16}$$

$$S[u] \quad = \quad A[u](x), \tag{1.17}$$

for a force measurement f and partial differential operator A. The distance measurement uses a force f to warp T into R, creating the image warp transformation T_u. The regularizing term is added to the objective function to differentiate between possible transformations, since the minimum distance may not be unique and one type of transformation may be preferred over another. Applying the Euler-Lagrange equations to (1.15)–(1.17) yields the system of nonlinear differential equations,

$$A[u](x) \quad - \quad f(x, u(x)) = 0,$$

from which the iteration scheme,

$$A[u^{k+1}](x) \quad - \quad f(x, u^k(x)) = 0, \tag{1.18}$$

is constructed. In particular, the following choices for D, S and A are used consistently with Modersitzki [24],

$$\begin{aligned} D[R, T_u] &= \frac{1}{2}\|T_u - R\|_{L_2(\Omega)} \\ S[u] &= \frac{1}{2}\sum_{l=1}^{d}\int_{\Omega}(\triangle u_l)^2 dx \\ A[u] &= \triangle^2 u, \end{aligned}$$

where $\|\cdot\|_{L_2(\Omega)}$ is the norm induced by (1.14). From Asaki [6], the iterative scheme constructed by (1.18), the optimal warp found can then be measured to compare the level of similarity between the *reference image* and *template images* dependant on the warp transformation.

Figure 1.3 shows an example of an image registration of two different simulated flows of heat for a fluid. The top left picture is the *reference image* R for a given set of parameter values, the top right is the *template image* T for a different set of parameter values, the bottom left is the warped template image T_u, and the bottom right is the difference between the reference image and the warped template image.

1.1.3 Objective Function

The goal is to find the parameters of a predetermined image among a range of choices from a numerical simulation. The objective function (1.1) accounts for both the numerical simulation and the numerical image registration and comparison methods described above through the following process. The objective function uses certain input parameters (e.g., Reynolds number) in the numerical simulation, resulting in an image that can be compared to a predetermined image. Based on the simulation and the predetermined image, the results of (1.15) enable the determination of the difference between the two images. Each possible image from the numerical simulation results in a distance value (**??**). The smallest distance value

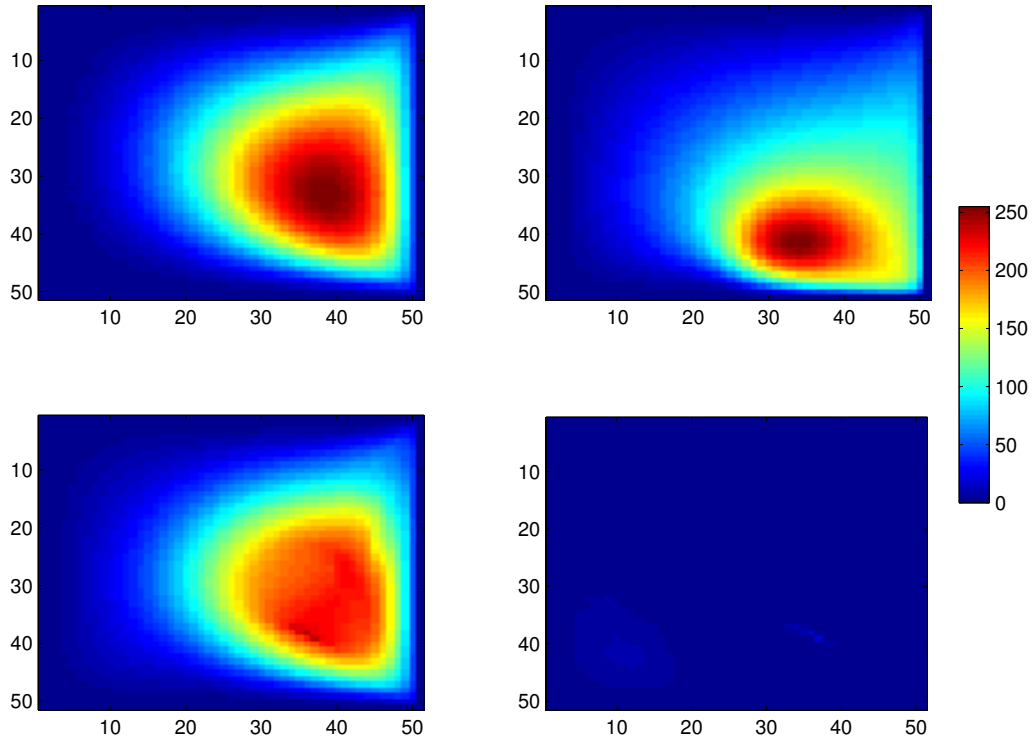

Figure 1.3 Image Registration of a Fluid Simulated Flow

implies that the parameter set for the numerical simulation match those of the predetermined image. For the numerical scheme presented for image registration (1.18), if the images are very similar, only a few iterations are necessary to transform the template image into the reference image resulting in a small distance value. If the images are farther apart, more iterations are necessary, which would increase the computational time and distance value. From the numerical scheme and distance value output, a direct correlation exists between the objective function value and the amount of computational time.

1.2 Purpose

The intent of this research is to develop an efficient strategy for numerically solving the class of optimization problems in which function values are expensive to evaluate, but become less so as a solution is approached. The method utilizes a less expensive surrogate function and a direct search method to find the optimal parameter values. A surrogate function can be thought of as a replacement for f that exhibits similar behavior. Surrogates are typically used when a function f is computationally expensive to evaluate. The nonlinear optimization problem will be solved by means of a direct search method, which does not rely on derivative information.

1.3 Overview

Chapter 2 outlines relevant literature on surrogates and direct search methods while Chapter 3 develops an approach using both methods. Chapter 4 describes the application of the method on two test problems and their results. Chapter 5 finishes this research with conclusions and recommendations for future research.

2. Relevant Literature

This chapter reviews literature related to the method developed to solve the optimizations problem (1.1). The first section introduces surrogate functions, their composition, and possible implementations. The next section examines pattern search algorithms as a solution method for nonlinear optimization.

2.1 Surrogate Functions

The idea for surrogates first appeared in the work of Schmit and Miura [29], who refer to surrogates as "approximation concepts." Booker, *et al.* [13] characterize a class of problems for which surrogate functions would be an appropriate approach, suggest a surrogate composition, and set forth a general framework, called the Surrogate Management Framework (SMF), for using surrogates to solve optimization problems numerically. Most surrogates are one of two types: simplified physics or response-based.

2.1.1 Simplified Physics: Low-Fidelity

A simplified physics model, also known as a low-fidelity model, makes certain physical assumptions that allow for a reduction in the computational cost but with less accuracy in the solution and parameter values. The assumptions not only reduce the computational cost through estimating or eliminating complex equations, but may also reduce the number of variables. These models are typically problem-dependent since they are based on specific characteristics of the problem.

Robinson *et al.* [26], take an approach for using a low-fidelity model as a surrogate through two transformations. For $x \in \mathbb{R}^n$, $\tilde{x} \in \mathbb{R}^{\tilde{n}}$, $\tilde{n} \leq n$, let $g(\tilde{x})$ be a low-fidelity model for a computationally expensive function $f(x)$. The surrogate $\hat{y}(x)$ is the composition of two transformations linking $f(x)$ to $g(\tilde{x})$. The first transformation must correct the surrogate to associate the difference between evaluations of $f(x)$

and $g(\tilde{x})$. The correction transformation can use an additive term (2.1) to update the surrogate, i.e.,

$$A(x) = f(x) - g(\tilde{x}) \quad \Rightarrow \quad f(x) = g(\tilde{x}) + A(x). \tag{2.1}$$

The term $A(x)$ is approximated by $\alpha(x)$ using a second-order Taylor series around a point x_c,

$$\begin{aligned} \alpha(x) &= A(x_c) + \nabla A(x_c) + \frac{1}{2}(x - x_c)^T \nabla^2 A(x_c)(x - A(x_c)) \\ &= [f(x_c) - g(\tilde{x_c})] + [\nabla f(x_c) - \nabla g(\tilde{x_c})] \\ &\quad + \frac{1}{2}(x - x_c)^T [\nabla^2 f(x_c) - \nabla^2 g(\tilde{x_c})](x - x_c). \end{aligned} \tag{2.2}$$

Then from (2.1),

$$\begin{aligned} \hat{y}(x) &= g(\tilde{x}) + \alpha(x) \\ &= g(\tilde{x}) + [f(x_c) - g(\tilde{x_c})] + [\nabla f(x_c) - \nabla g(\tilde{x_c})] \\ &\quad + \frac{1}{2}(x - x_c)^T [\nabla^2 f(x_c) - \nabla^2 g(\tilde{x_c})](x - x_c). \end{aligned} \tag{2.3}$$

The second transformation is a space mapping function P that handles the change in variable dimensions (see Bandler *et al.* [11] for details). It is defined by

$$\tilde{x} = P(x) \tag{2.4}$$

such that

$$\|g(\tilde{x}) - f(x)\| \leq \varepsilon$$

where $\|\cdot\|$ is a suitable norm and $\varepsilon > 0$. Combining transformations (2.3) and (2.4), the surrogate function becomes

$$\hat{y}(x) \;=\; g(P(x)) + \alpha(x).$$

2.1.2 Response-Based Model: Design and Analysis of Computer Experiments (DACE)

A response-based model makes no assumptions on the physics involved and is constructed from known responses of the function. For a model to be based on the responses of the function, a basic structure of the surrogate must be assumed. Sacks *et al.* [27] discuss a way to build a model using both a generalized least-squares regression model and interpolation. In regression analysis, an output is treated as a realization of a stochastic process, since for the same input parameters, a different output value can occur. A computer experiment, however, has the same output value for the same input parameters. Using the deterministic response of the computer experiment, it is treated as if it is a realization of a stochastic process from which to build a regression model. The variance associated with the generalized least squares regression model is used to form a correlation matrix to interpolate the regression model through the deterministic responses at known data sites. The generalized least squares model can then act as a predictor of the response at an unknown point, based on the known points and their known function values.

Following [27] and [22], the deterministic function $\hat{y}(z)$ can be modeled as a realization of a stochastic process $Y(z)$, which is composed of a regression model with coefficients $\beta \in \mathbb{R}^n$ and a random variable $Z : \mathbb{R}^n \longrightarrow \mathbb{R}$, namely,

$$Y(z) = \sum_{j=1}^{n} \beta_j f_j(z) + Z(z) = \beta^T f(z) + Z(z). \tag{2.5}$$

The random function $Z(\cdot)$ is assumed to have a mean of zero and covariance

$$V(w,z) \;=\; \sigma^2 R(\theta,w,z) \tag{2.6}$$

between $Z(w)$ and $Z(z)$, where σ^2 is the process variance and $R(\theta,w,z)$ is the correlation between w and z.

Kriging produces an approximate function value at an unknown point using weights on known responses. Given a set of known data points $\{x_i\}_{i=1}^k \subset \mathbb{R}^n$ and their response $\{y_x\}_{x=1}^k \subset \mathbb{R}^m$, a function value at an unknown point $z \in \mathbb{R}^n$ can be approximated from

$$\hat{y}(z) = c(x)^T y_x \tag{2.7}$$

where $c(x) \in \mathbb{R}^m$ is a vector of weights. The best kriging weights are obtained by minimizing the mean squared error (MSE) between the true model (2.5) and the approximate model (2.7), given by

$$\begin{aligned} MSE[\hat{y}(z)] &= E[c(x)^T y_x - Y(z)]^2 \\ c(x)^T y_x - Y(z) &= c(x)^T(\beta^T f(x) + Z(x)) - (\beta^T f(z) + Z(z)) \\ &= c(x)^T Z(x) - Z(z) + (f(x)^T c(x) - f(z))^T \beta. \end{aligned}$$

To ensure that the linear predictor is unbiased, the condition

$$f(x)^T c(x) - f(z) = 0 \tag{2.8}$$

must hold. In this case, it follows from (2.8) that

$$c(x)^T y_x - Y(z) = c(x)^T Z(x) - Z(z),$$

and

$$\begin{aligned} MSE[\hat{y}(z)] &= E[c(x)^T Z(x) - Z(z)]^2 \\ &= E[Z(z)^2 + c(x)^T Z(x) Z(x)^T c(x) - 2c(x)^T Z(x) Z(z)]. \end{aligned} \quad (2.9)$$

From (2.6),

$$\begin{aligned} E[Z(z)] &= \sigma^2 \\ E[Z(x)Z(z)] &= V(x,z) = \sigma^2 R(\theta, x, z) \\ E[Z(x)Z(x)^T] &= V(x,x) = \sigma^2 R(\theta, x, x), \end{aligned}$$

which simplifies (2.9) to

$$MSE[\hat{y}(z)] = \sigma^2 + \sigma^2 c(x)^T R(\theta, x, x) c(x) - 2\sigma^2 c(x)^T R(\theta, x, z). \quad (2.10)$$

The optimal weights $c(x)$ can be found by solving an optimization problem formulated from (2.8) and (2.10), namely,

$$\begin{aligned} \min_{c} \quad & \sigma^2(1 + c(x)^T R(\theta, x, x) c(x) - 2c(x)^T R(\theta, x, z)) \\ \text{s.t.} \quad & f(x)^T c(x) - f(z) = 0. \end{aligned}$$

To solve this problem, the Lagrangian function $L(c, \lambda)$, with multiplier $\lambda \in \mathbb{R}$ is formulated, as

$$\begin{aligned} L(c,\lambda) = \sigma^2(1 + c(x)^T R(\theta, x, x) c(x) - 2c(x)^T R(\theta, x, z)) \\ -\lambda^T (f(x)^T c(x) - f(z)). \end{aligned}$$

Taking the gradient with respect to $c(x)$ and λ, the first-order necessary conditions for optimality become

$$\begin{aligned} R(\theta,x,x)c(x)+\widetilde{\lambda}f(x) &= R(\theta,x,z), && (2.11)\\ \widetilde{\lambda} &= \lambda/2\sigma^2, \\ f(x)^T c(x) &= f(z). && (2.12) \end{aligned}$$

Solving the system of equations in (2.11) and (2.12) yields

$$\begin{aligned} \widetilde{\lambda} &= (f(x)^T R(\theta,x,x)^{-1}f(x))^{-1}(f(x)R(\theta,x,x)^{-1}R(\theta,x,z)-f(z)) && (2.13)\\ c(x) &= R(\theta,x,x)^{-1}(R(\theta,x,z)-f(x)\widetilde{\lambda}). && (2.14) \end{aligned}$$

Substituting (2.13) and (2.14) into the predictor (2.7),

$$\begin{aligned} \hat{y}(z) &= c(x)^T y_x \\ &= R(\theta,x,x)^{-1}(R(\theta,x,z)-f(x)\widetilde{\lambda})^T y_x \\ &= R(\theta,x,z)^T R(\theta,x,x)^{-1}y_x - (f(x)^T R(\theta,x,x)^{-1}R(\theta,x,z)-f(z))^T \\ &\quad (f(x)^T R(\theta,x,x)^{-1}f(x))^{-1}f(x)^T R(\theta,x,x)^{-1}y_x \\ &= f(z)^T\beta^* + R(\theta,x,z)^T\gamma^*, && (2.15) \end{aligned}$$

where

$$\begin{aligned} \beta^* &= (f(x)^T R(\theta,x,x)^{-1}f(x))^{-1}f(x)R(\theta,x,x)^{-1}y_x, && (2.16)\\ \gamma^* &= R(\theta,x,x)^{-1}(y_x - f(x)\beta^*). && (2.17) \end{aligned}$$

The correlation matrices in (2.15)–(2.17) are based on the distance between all points and can be expressed as

$$R(\theta,x,x) = \begin{pmatrix} R(\theta,x_1,x_1) & R(\theta,x_1,x_2) & \dots & R(\theta,x_1,x_n) \\ R(\theta,x_2,x_1) & R(\theta,x_2,x_2) & \dots & R(\theta,x_2,x_n) \\ \vdots & \vdots & \vdots\ddots & \dots \\ R(\theta,x_n,x_1) & R(\theta,x_n,x_2) & \dots & R(\theta,x_n,x_n) \end{pmatrix}$$

$$R(\theta,x,z) = \begin{pmatrix} R(\theta,x_1,z) & R(\theta,x_2,z) & \dots & R(\theta,x_n,z) \end{pmatrix}^T$$

for known points $x_i \in \mathbb{R}^n$, $i = 1 \dots k$ and unknown point, $z \in \mathbb{R}^n$. Then $R(\theta,a,b)$ can be specified as

$$\begin{aligned} R(\theta,a,b) &= \prod_{j=1}^{n} R_j(\theta_j,|d_j|) \\ d_j = a_j - b_j &\qquad j = 1 \dots n \end{aligned} \tag{2.18}$$

for any points $a,b \in \mathbb{R}^n$. Many choices for the function $R_j(\theta_j,|d_j|)$ have been proposed to allow a user to determine the amount of influence that known points should exert on an unknown point. Common choices include the following [22]:

$$\begin{aligned} R_j(\theta_j,|d_j|) &= \\ \textit{exponential} &: \exp(-\theta_j|d_j|) \\ \textit{general exponential} &: \exp(-\theta_j|d_j|^{\theta_{n+1}}) \quad 0 < \theta_{n+1} \le 2 \\ \textit{gaussian} &: \exp(-\theta_j d_j^2) \\ \textit{linear} &: \max\{0,\ 1-\theta_j|d_j|\} \\ \textit{spherical} &: 1 - 1.5\xi_j + 0.5\xi_j^3 \quad \xi_j = \min\{1,\ \theta_j|d_j|\} \\ \textit{cubic} &: 1 - 3\xi_j^2 + 2\xi_j^3 \qquad \xi_j = \min\{1,\ \theta_j|d_j|\} \\ \textit{spline} &: \varsigma(\xi_j) \qquad\qquad \xi_j = \theta_j|d_j| \end{aligned} \tag{2.19}$$

$$\varsigma(\xi_j) = \begin{cases} 1 - 15\xi_j^2 + 30\xi_j^3, & 0 \le \xi_j \le 0.2 \\ 1.25(1-\xi_j)^3, & 0.2 < \xi_j < 1 \\ 0, & 1 \le \xi_j. \end{cases}$$

The optimal choice for θ is the maximum likelihood estimator θ^* that solves

$$\min_{\theta} |R(\theta, x, x)|^{1/m} \hat{\sigma}^2, \tag{2.20}$$

where

$$\begin{aligned} |R(\theta, x, x)| &= det(R) \\ \hat{\sigma}^2 &= \frac{1}{m}(y_x - f(x)\beta^*)^T R(\theta, x, x)^{-1}(y_x - f(x)\beta^*). \end{aligned}$$

2.2 Generalized Pattern Search

Generalized Pattern Search (GPS) is a class of direct search methods that generates a sequence of iterates with nonincreasing function values to numerically solve optimization problems without utilizing derivative information. Torczon [30] introduced pattern search for unconstrained optimization problems as a generalization of several well-known methods, including the method of Hooke and Jeeves [18] and the multidirectional search algorithm of Dennis and Torczon [16], and showed if the objective function f is continuously differentiable and all iterates lie in a compact set, then a subsequence of iterates converges to a first-order stationary point. Lewis and Torczon later extended GPS to problems with simple bounds [19] and linear constraints [20]. Audet and Dennis [7], developed a hierarchy of convergence results for GPS that depends on the smoothness properties of f, while second-order convergence behavior was studied by Abramson [3].

Further extensions include work to manage generally constrained optimization and mixed variables. For nonlinearly constrained optimization, Audet and Dennis

introduced a filter GPS method [9], while Lewis and Torczon use an augmented Lagrangian approach [21], both of which allow infeasible points in the iteration sequence. Audet and Dennis [8] also introduced a mixed variable GPS algorithm for bounded constrained problems with continuous and categorical variables, which was extended by Abramson [2] using filters for problems with nonlinear constraints and mixed variables.

Audet and Dennis [7] describe GPS as a two step process in generating a sequence of nonincreasing function values. At each iteration, GPS executes a SEARCH and POLL, which are executed on a mesh. The optional search step is very general in that it simply evaluates a finite number of mesh points, and can be implemented in a variety of ways. This allows the user to choose a specific heuristic suited for the optimization problem. Common choices include the use of surrogates for expensive objective functions or a random search of the region if nothing is known about the objective function. The search step contributes nothing to the convergence theory, but is often successful in aiding the algorithm find a quick improvement.

An example of a random search method is the Latin hypercube design which consists of a random set of "space filling" points. This is done by dividing each of n dimensions into m intervals of equal length, where m is the number of points desired. This creates m^n sectors for the given space. Random points within the m random sectors are chosen such that each column for each dimension is chosen only once, as described by Santner *et al.* [28]. On the region $[0,1] \times [0,1]$, Figure 2.1 shows an example of $m = 4$ random points chosen on a two-dimensional Latin hypercube design. From the 16 sectors, 4 random points are chosen so that no row or column is repeated.

The search step continues until no further improvement in f can be found, at which, point the poll step is invoked. Polling consists of an examination of the points

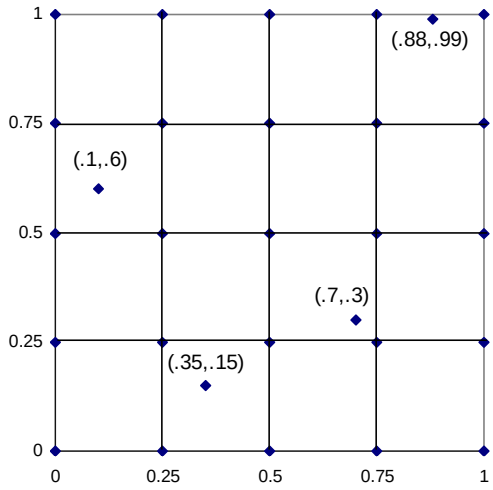

Figure 2.1 Latin Hypercube Design (2-D, 4-Points)

P_k neighboring the current solution x_k on a mesh M_k, which is defined by

$$M_k = \{x_k + \Delta_k^m Dz : z \in \mathbb{Z}_+^{|D|}\} \tag{2.21}$$

where Δ_k^m is the mesh size and D is a set of positive spanning directions (defined by Davis [15]). The set D must be constructed so that each direction $d_j \in D$ is required to be the product Gz_j for some fixed nonsingular generating matrix $G \in \mathbb{R}^{n\times n}$ and $z_j \in \mathbb{Z}^n$.

The set of points on the mesh neighboring the current solution x_k is called the *poll set* and can be expressed as

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\}$$

where $D_k \subseteq D$ is also a positive spanning set.

From the poll set, points are evaluated until an improvement in the objective function is found or until all points in P_k have been evaluated. If an improvement is found, the improved point becomes the new iterate and the mesh size can be relaxed

according to the rule

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m, \tag{2.22}$$

where $0 < \tau^{w_k} < 1$, for $\tau \in \mathbb{Q}$, $w_r \leq -1$ and $w_k \in [w_r, -1] \cap \mathbb{Z}$. If no improvement is found, the current solution is retained and the mesh is tightened according to

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m, \tag{2.23}$$

where $0 < \tau^{w_k} < 1$, for $\tau \in \mathbb{Q}$, $w_t \geq 0$ and $w_k \in [0, w_t] \cap \mathbb{Z}$.

2.2.1 Mesh Adaptive Directed Search

Mesh Adaptive Directed Search (MADS) was introduced by Audet and Dennis [10] as a generalization of GPS that extends to nonlinear constraints without the use of a penalty function or filter. The search step in MADS is the same as in GPS; the difference lies within the poll step. MADS adopts the idea of a *frame* from Coope and Price [14], which is the *poll set* in GPS, and generates an asymptotically dense set of refining directions. The increased number of directions used by MADS leads to a stronger convergence theory than that of GPS. Audet and Dennis [10] provided convergence to a first-order stationary point for general nonlinearly constrained optimization problems, even in the nonsmooth case. Abramson [4] gives reasonable conditions under which convergence to a local solution is ensured. In GPS, the mesh size parameter Δ_k^m dictates the direction and magnitude of the mesh are equal for each positive spanning set. MADS overcomes this rule by introducing a poll size parameter Δ_k^p such that $\Delta_k^m \leq \Delta_k^p$ for updating Δ_k^m for all k,

$$\lim_{k \in K} \Delta_k^p = 0 \quad \Rightarrow \quad \lim_{k \in K} \Delta_k^m = 0. \tag{2.24}$$

The updated POLL SET of frame becomes,

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\} \tag{2.25}$$

where D_k is a positive spanning set such that each $d \in D_k$ must have three properties (see Audet and Dennis [10]): d can be written as a nonnegative integer combination of the directions in D: $d = Du$ for some vector $u \in \mathbb{N}^{n_{D_k}}$ that may depend on the iteration number k, $\Delta_k^m \|d\| \leq \Delta_k^p \max\{\|d'\| : d' \in D\}$ and limits of the normalized sets D_k are a positive spanning set, Audet and Dennis [10].

Figures 2.2 and 2.3 (from Audet and Dennis [10]) illustrate the difference between GPS and MADS frames. For GPS, $\Delta_k^p = \Delta_k^m$ and for MADS, $\Delta_k^p = n\sqrt{\Delta_k^m}$, n is the dimension. In Figure 2.2, the equal frame and mesh sizes limit the number of directions GPS can investigate. Figure 2.3, however, demonstrates how MADS allows polling in increasingly different directions and magnitudes as the mesh size decreases faster than the poll size. This yields an algorithm similar to GPS, as seen in Figure 2.4 [10].

2.3 Conclusion

This chapter reviewed the relevant literature and provided an introduction to surrogates, as well as MADS, to solve an optimization problem. In the next chapter, a method is developed using these ideas to numerically solve the optimization problem (1.1).

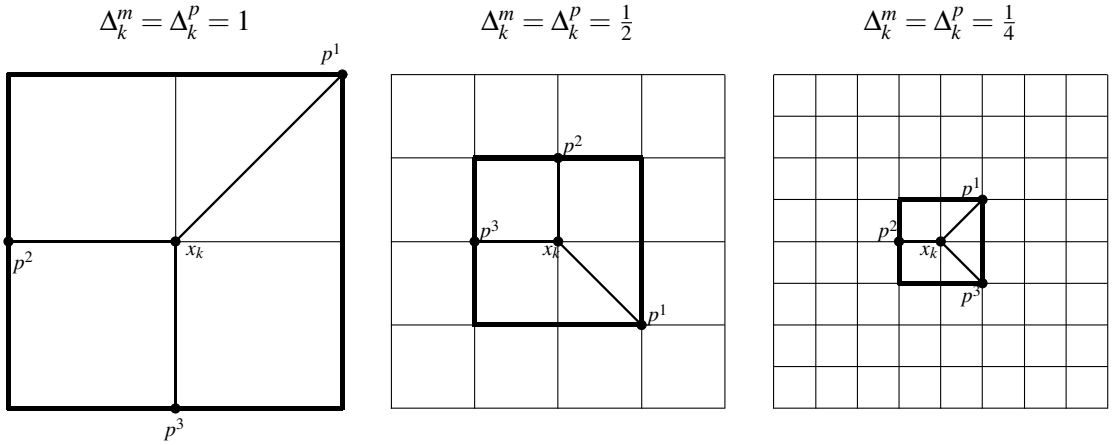

Figure 2.2 GPS Frame

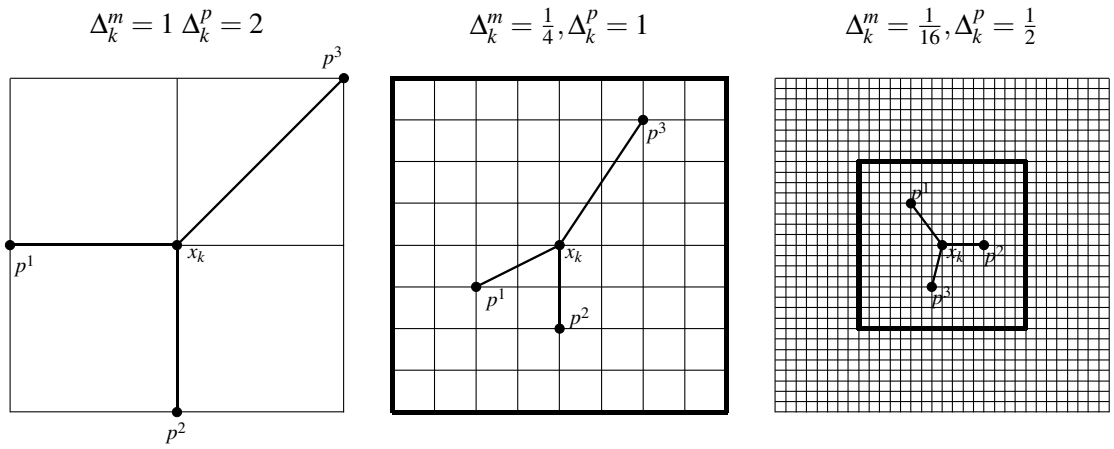

Figure 2.3 MADS Frame

A GENERAL MADS ALGORITHM

• INITIALIZATION: Let $x_0 \in \Omega$, $\Delta_0^m \leq \Delta_0^p$, D, G, τ, w_r, and w_t satisfy the requirements above. Set the iteration counter $k \longleftarrow 0$.

• SEARCH AND POLL: Perform the search and possibly the poll steps (or only part of them) until an improvement mesh point x_{k+1} is found on the mesh, M_k (See (2.21)).

OPTIONAL SEARCH: Evaluate f_Ω on a finite subset of trial points on the mesh, M_k.

LOCAL POLL: Evaluate f_Ω on the from P_k. (See (2.25)).

• PARAMETER UPDATE: Update Δ_{k+1}^m according to (2.22) or (2.23) and Δ_{k+1}^p according to (2.24). Set $k \longleftarrow k+1$ and go back to the SEARCH AND POLL step.

Figure 2.4 MADS Algorithm

3. Methodology

This chapter describes the approach for solving the optimization problem presented in Chapter 1 with the goal of solving the problem as quickly as possible. This chapter outlines an arrangement between the use of surrogates within the application of a pattern search algorithm.

3.1 Optimization Problem and Notation

The optimization problem from Chapter 1 is a computationally expensive, black box function,

$$\min_{x \in \Omega} f(x) \tag{3.1}$$

for $f : \mathbb{R}^n \to \mathbb{R} \cup \{\pm\infty\}$, $\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ and $l, u \in (\mathbb{R} \cup \{\pm\infty\})^n$ for $l < u$. The overall approach in solving (3.1) numerically is the implementation of GPS or MADS with a barrier approach [10]. The application of the barrier forces $f(x) = \infty$, whenever $x \notin \Omega$. To treat (3.1) when f is computationally expensive and the CPU time required to evaluate f at a point x decreases as x approaches a solution, a new notation is first introduced, in which the time to compute a function value is added as part of the input and output; i.e., $[z, t] = f(x, t_{cut})$, where $x \in \Omega$ is a trial point, t_{cut} is a user-specified CPU time threshold, z is the function value at x and t is the computational time needed to compute z. Once the computational time of a function evaluation exceeds the value specified by t_{cut}, then evaluation of f is aborted. This is done with the expectation that a lower objective value will not be produced for the increased time required for the evaluation. Given the current minimal solution $\hat{z}_s$ at iteration n with a computational time of t_s, let $t_{cut} = t_s$ for x_{n+1}. The value for t_{cut} can be changed after each iteration to help reach a solution quickly.

3.2 Search: Optimization Surrogate

The search step for a pattern search method also make use of the time feature. From the points evaluated, the function values and times can be used individually as responses, to form surrogates $F(x)$ and $T(x)$, respectively. The resulting surrogate optimization problem is then solved cheaply at each search step to find a point at which to evaluate f. The application of the two surrogates are tested in four different configurations:

$$1. \quad \min_{x \in \Omega} F(x), \tag{3.2}$$

$$2. \quad \min_{x \in \Omega} F(x) \quad \text{s.t.} \quad T(x) \leq t_{cut} + \varepsilon, \tag{3.3}$$

$$3. \quad \min_{x \in \Omega} T(x), \tag{3.4}$$

$$4. \quad \min_{x \in \Omega} T(x) \quad \text{s.t.} \quad F(x) \leq \hat{z}_s, \tag{3.5}$$

where the constant offset ε is added to the constraint in (3.3) to allow for variability in computational time. The first surrogate is a typical application of using the function values to construct the surrogates, while the second adds a constraint surrogate based on time. The idea is to make sure the function surrogate and time surrogate both see a decline in value for possible input values. The third and fourth configurations switch the roles of the two surrogates with the same intent of finding a decline.

To solve the surrogate optimization problems presented, the same pattern search method is applied. The method to solve the surrogate problem also uses the barrier approach along with a nonlinear closed constraint in the case of (3.3) and

(3.5). The nonlinear closed constraint is treated using the same barrier approach described in Section 3.1. A function value of ∞ is assigned for all infeasible points. The expectation is that an improved solution may be found quickly by solving the surrogate problem, rather than spending the time in the poll step.

The combination of a pattern search method with a barrier approach and the use of the parameter t_{cut} in the original optimization problem causes a dilemma when using surrogate functions. When the parameter t_{cut} stops the evaluation of f at a point $x \in \Omega$, the function value is unavailable, and a value of ∞ is typically assigned. However, when constructing a surrogate, a value of ∞ is not viable as a response. Therefore, a different value is imposed when t_{cut} is reached, for $[z_i, t_i] = f(x_i, t_{cut})$, if $t_i = t_{cut}$ then set $z_i = \max\{z_1, \ldots, z_{i-1}\}$.

3.3 Surrogate Composition

For the surrogate problems developed in Section 3.2, DACE surrogates from Section 2.1.2 were implemented, since the objective function (3.1) is treated as a black box. The next subsections discuss the initial points used to generate the DACE surrogates, the order of the regression polynomial, and the correlation function chosen to model the responses.

3.3.1 Initial Points

A set of initial points must be evaluated and then used to generate a surrogate. From a global perspective, a desirable property for the initial points is that they be "space-filling". The idea is to sample enough points in the given space to construct a reasonably accurate initial surrogate. Santner *et al.* [28] discuss two main types of space-filling techniques: experimental and Latin hypercube designs.

Latin hypercube designs were discussed in Section 2.2 as a possible step in the GPS algorithm to generate a set of random points in a region. The points generated

could then be used to construct the initial surrogate. However, when considering the structure of DACE surrogates, the order of the regression model dictates more than just random space-filling points.

In an experimental design approach, Myers and Montgomery [25] assert that there are several characteristics to consider when choosing an initial set of points or design. To minimize a DACE surrogate and obtain a proper estimate of $\hat{y}(z)$ for the unknown point z, a second-order polynomial,

$$\hat{y}(z) = \beta_o + \sum_{i=1}^{k} \beta_i z_i + \sum_{i=1}^{k} \beta_{ii} z_i^2 + \sum_{i=1}^{k} \sum_{j<i}^{k} \beta_{ij} z_i z_j + R(\theta, x, z)^T \gamma^*$$

is used where $\beta_{ij}, i, j = 0, 1, \ldots, k$, γ^*, and $R(\theta, x, z)$ are defined in (2.16)–(2.18). A second-order polynomial is chosen given Myers and Montgomery [25] state it as a more accurate predictor of $\hat{y}(z)$ than a polynomial of a lower order from. To properly estimate the coefficients β_{ij}, $i, j = 0, 1, \ldots k$, of the linear and quadratic terms, a central composite design (CCD) is to determine for the initial set of points. Figure 3.1 is an illustration of a 2-dimensional CCD in which the axial and center points allow for an estimation of quadratic coefficients, while the $(\pm 1, \pm 1)$ points enable approximation of the linear terms and two-factor interactions. Normally, a CCD includes three replications at the center point, but this is not done here because there is no randomness in the responses, and function evaluations are computationally expensive.

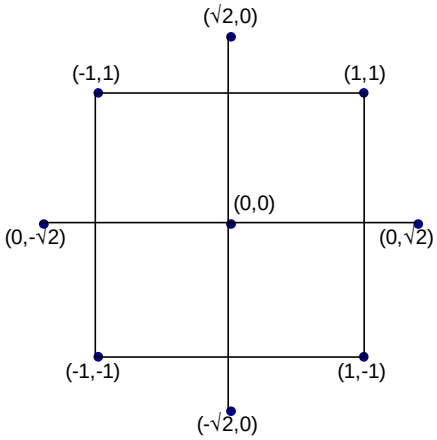

Figure 3.1 Central Composite Design (2-D)

3.3.2 Correlation Matrix

The choice of correlation function affects the performance of the algorithm. A common choice is the Gaussian process given by

$$R(\theta, a, b) = \prod_{j=1}^{n} R_j(\theta_j, |d_j|) \tag{3.6}$$

$$\begin{aligned} R_j(\theta_j, |d_j|) &= \exp(-\theta_j d_j^2) \\ d_j &= a_j - b_j \end{aligned} \tag{3.7}$$

for any points $a, b \in \mathbb{R}^n$. Lophaven *et al.* [22] have found that in practice, the Gaussian process often shows the same behavior of the desired function as the number of known points increases. However, when solving for $R(\theta, x, x)^{-1}$ (see (2.16) and (2.17)), Booker [12] demonstrates that $R(\theta, x, x)^{-1}$ can become ill-conditioned as points cluster together during the convergence process of a pattern search method. The increase in the condition number, $\kappa(R)$ of $R(\theta, x, x)$, results in a loss of significant digits which can greatly impact the "optimal" θ found (see 2.20) or prevent the calculation of β altogether. For specific values of θ and data sites, $R(\theta, x, x)$ is a

symmetric matrix that can be factored via Cholesky decomposition:

$$R = CC^T$$

and

$$\kappa(R) = \kappa(C)^2 = \frac{\lambda_{max}^2(C)}{\lambda_{min}^2(C)},$$

where λ_{max} and λ_{min} are the largest and smallest eigenvalues of C, respectively, obtained from the diagonal of C. The condition number can be reviewed periodically for different possible values of θ. If the condition number is too large the SEARCH step is skipped and only polling is executed.

3.4 Implementation

The algorithm presented in Figure 3.2 incorporates the ideas developed in this chapter to solve (3.1) efficiently. MADS–TIME executes a pattern search method with surrogates based on function values and computational times with the addition of a check on the surrogates condition numbers $\kappa(R)$ and an updating scheme for the time cut-off parameter using the current solutions $\hat{z}_s$ computational time. The algorithm will be applied to two test problems in the next chapter.

• INITIALIZE

Given an initial set of points $\{x_i\}_{i=1}^n$, set $t_{cut} = \infty$, evaluate $[z_i, t_i] = f(x_i, t_{cut})$, $i = 1, 2 \ldots, n$. Set $k = n$, $s = 1$, $[\hat{z}_s, t_{cut}] = [\min\{z_i\}_{i=1}^k, t_i]$, and M_s satisfying (2.21) for given parameters Δ_s^p, Δ_s^m, D, τ and w. Set mesh size stop criteria Δ_{stop} and surrogate condition number threshold κ.

• SEARCH

For all known responses $\{x_i, t_i, z_i\}_{i=1}^k$, build surrogate optimization problem.

- ◦ If $\kappa(R) > \kappa$ proceed to POLL.
- ◦ Else $\kappa(R) \leq \kappa$, solve surrogate problem yielding x_{k+1}, and evaluate $[z_{k+1}, t_{k+1}] = f(x_{k+1}, t_{cut})$.
  - • If $t_{k+1} = t_{cut}$, set $z_{k+1} = \max\{z_i\}_{i=1}^k$, $k = k + 1$, proceed to POLL.
  - • Else $t_{k+1} \neq t_{cut}$.
    - ◦ If $z_{k+1} < \hat{z}_s$ an improvement has been found. Set $k = k + 1$, $s = s + 1$, $\hat{z}_s = z_k$, $t_{cut} = t_k$, update Δ_s^m (2.22), Δ_s^p (2.24), and return to SEARCH.
    - ◦ Else $z_{k+1} \geq \hat{z}_s$ and no improvement has been found, set $k = k + 1$, proceed to POLL.

• POLL

- ◦ If $z_{k+1} > \hat{z}_s \ \ \forall\, x_{k+1} \in P_s$ Set $s = s + 1$, $\hat{z}_s = \hat{z}_{s-1}$ and update Δ_s^m (2.23), Δ_s^p (2.24), and proceed to CONVERGENCE.
- ◦ Else evaluate a point $x_{k+1} \in P_s$ (2.25), $[z_{k+1}, t_{k+1}] = f(x_{k+1}, t_{cut})$.
  - • If $t_{k+1} = t_{cut}$, set $z_{k+1} = \max\{z_i\}_{i=1}^k$, $k = k + 1$, return to POLL.
  - • Else $t_{k+1} \neq t_{cut}$.
    - ◦ If $z_{k+1} < \hat{z}_s$ an improvement has been found. Set $k = k + 1$, $s = s + 1$, $\hat{z}_s = z_k$, $t_{cut} = t_k$, update Δ_s^m (2.22), Δ_s^p (2.24), and return to SEARCH.
    - ◦ Else $z_{k+1} \geq \hat{z}_s$ and no improvement has been found, set $k = k + 1$, return to POLL.

• CONVERGENCE

- ◦ If $\Delta_s^p < \Delta_{stop}$, stop convergence criteria has been met.
- ◦ Else $\Delta_s^p > \Delta_{stop}$, return to SEARCH.

Figure 3.2 MADS–TIME

4. Implementation

This chapter focuses on implementing the numerical algorithm presented in Chapter 3 on a suite of test problems, and reporting numerical results. For each scenario, different variations of the algorithm are applied and compared to a base case. The base case implementation uses GPS with no search, a single initial point or CCD set of points, and $t_{cut} = \infty$. This allows for a full evaluation of all points and a comparative analysis of the proposed algorithm. The other cases are the partial and full implementation of the algorithm presented in Figure 3.2. Depending on the solution and effectiveness of the algorithm presented in Figure 3.2, MADS may also be implemented.

4.1 Processing and Coding

From Section 3.4, the algorithm presented was run on a Linux operating system using two MATLAB® software packages, NOMADm [4] for the implementation of GPS and MADS algorithms and DACE [23] to build the surrogates, along with custom search files. NOMADm requires five files to run the optimization problem, four of which set up the parameters, objective function, variable bounds, and initial points. The fifth file sets up a custom search for optimizing the surrogate problem. The surrogate problem is solved by a recursive call to the NOMADm optimizer from within the search step using four surrogate files. Each file can be seen in Appendix A.

4.2 Test 1: Lid-Driven Cavity

The first test problem considered is known as the lid-driven cavity problem. For a given two-dimensional square domain, the Navier-Stokes equations describe a fluid flow with a horizontal velocity force on one edge. At time zero, the fluid is

at rest. Once time starts, a constant horizontal velocity is asserted along the top edge, causing a circular pattern of flow to appear within the fluid over time. For different Reynolds numbers and simulation lengths, the velocity and viscosity of the fluid form a different circular heat pattern throughout the region. At one particular Reynolds number and simulation length, a reference image of the heat pattern is captured and then noise is added into the image. The goal is to run a simulation for different Reynolds numbers and simulation lengths, capture the template image, and compare the template and reference images of heat in an attempt to determine the original Reynolds number and simulation length set for the reference image. Figure 1.3 shows the reference image and a template image with their comparison.

4.2.1 Initial Runs

The base case of GPS with no search was applied and reached a solution, but initial attempts using the MADS–TIME failed. After studying the problem in detail, the reason for failure became evident. Figure 4.1 shows the relationship between the function values at each trial point and the CPU time required to compute it. From Figure 4.1, it is clear that the main assumption does not hold. The computational time does decrease as the optimal solution is reached, but only below a certain function value. If the function values are too far from that of the optimal solution (i.e. too high), the computational time starts to decrease. Since the initial point(s) produced function values above 200, t_{cut} does not allow the computation of a value between 25 and 200, leading the GPS algorithm to terminate as though the optimal solution is around 200 without finding the true optimum. This situation was remedied by changing the t_{cut} parameter. Given a current minimal solution with its computational time $[z_s, t_s]$, let $t_{cut} = 2t_s$.

The preliminary runs of the surrogate optimization problem provided useful information. However, as GPS progressed, the surrogate became ill-conditioned, due to the clustering of trial points as the mesh size is reduced. Even though measures

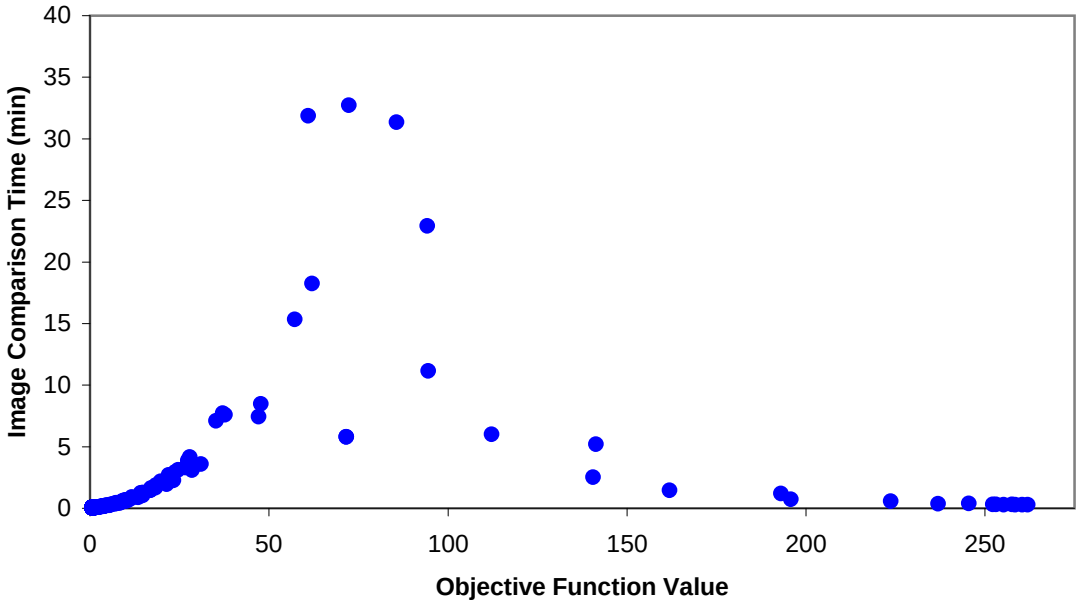

Figure 4.1 Test Problem 1: Function and Time Correlation

were in place to deter the possibility of ill-conditioning, it could not be prevented, and the loss of significant digits was overwhelming. To combat this, a measure was added to stop solving the surrogate problem once computational times fell below some threshold or if the condition number was too high. This makes sense for this class of problems, since the objective function evaluations are no longer expected to be expensive.

4.2.2 Results-GPS

The results for each case are shown in Table 4.1. The first three are base cases with no search step and $t_{cut} = \infty$. The first has one initial point at the center of the constraint region, the second chooses a random initial feasible point, and the third generates a set of initial points from a CCD. The last seven cases are different variations of the MADS–TIME algorithm. The first three are similar to the base case (no search step) except for the use of the t_{cut} parameter to abort expensive function evaluations. The random point is the same random initial point that was

Full Time	Search	Initial Point(s)	Objective Value	Reynolds Number	Simulation Length	Number of Iterations	Number of Evaluations	Total CPU Time (min)
	None	Center	0.60	134.13	4.76	56.00	123.00	126.73
	None	Random	0.60	134.12	4.76	58.00	118.00	178.31
	None	CCD	0.60	134.13	4.76	40.00	107.00	196.58
Cut Time								
	None	Center	0.60	134.13	4.76	80.00	157.00	257.67
	None	Random	0.60	134.12	4.76	92.00	162.00	796.40
	None	CCD	0.60	134.13	4.76	40.00	107.00	109.73
	F(x) s.t. T(x)	CCD	0.60	134.19	4.76	73.00	162.00	135.78
	F(x)	CCD	0.60	134.19	4.76	72.00	165.00	182.89
	T(x) s.t. F(x)	CCD	0.60	134.19	4.76	52.00	133.00	74.00
	T(x)	CCD	0.60	134.19	4.76	49.00	127.00	74.48

Table 4.1 GPS: Lid-Driven Cavity Results

used for the base case. The final four employ one of the surrogate optimization problems (3.2), (3.3), (3.4), or (3.5). For each implementation, certain information was collected, including the optimal parameters found with the function value, the number of iterations and function evaluations executed, and the overall time it took to find the solution.

All runs found the optimal solution at the same parameter values. The quickest convergence occurred when the surrogate time function was minimized, subject to a surrogate constraint based on function values (3.5). Closely behind is the surrogate time function with no constraint (3.4). Analysis of the runs provide interesting insight and shows how each component of the proposed algorithm affected the computational time.

Analysis of the first three runs (provided in Figure 4.1) yields the following observations. The initial center point took more function evaluations than with a random initial point, but the former took significantly less time. This illustrates the importance of getting close to the solution quickly so that the time to evaluate the objective function decreases quickly and why the number of function calls is not a good measure of the effectiveness of the different implementations.

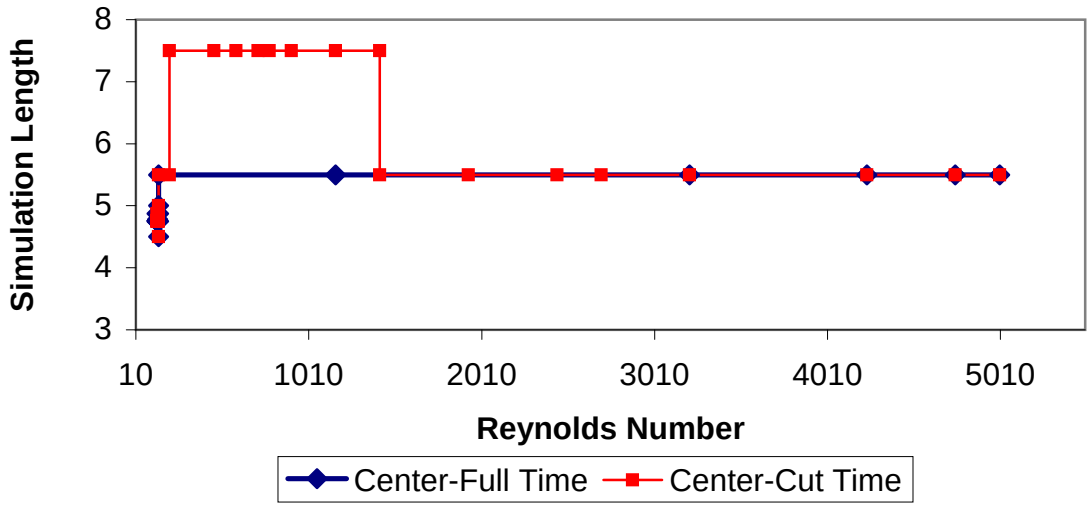

Figure 4.2 Center Comparison

The results with no search demonstrate the importance of finding an appropriate initial point to set up the t_{cut} parameter. With only one initial point, the t_{cut} parameter needs several iterations to build enough slack to allow the sequence of points to overcome the phenomenon seen in Figure 4.1. The extra iterations result in a different path to a solution, which requires significantly more time. Figure 4.2 illustrates the differences for the two-dimensional case with simulation length and Reynolds number as variables. The blue line representing the path, and blue diamonds for a function evaluation with no t_{cut}, and the red line and squares represent with t_{cut}. So that there are 34 more function evaluations and the paths are not the same when the t_{cut} parameter was implemented. However, the results using initial CCD points and no search shows an appropriate point for t_{cut}, resulting in a faster convergence. Both cases that use the CCD initial points take the same path to the solution, but using the t_{cut} parameter saves almost 90 minutes of time saved compared to not using it.

Analysis of the search surrogates showed useful information was provided in finding the optimal solution. The use of the constraint in both surrogate searches provides an additional level of information to the search, leading to a different path

Cut Time	Search	Initial Points	Number of Search Surrogates Implemented	Number of Lower Objective Function Value	Percentage
	F(x) s.t. T(x)	CCD	44	23	52.27%
	F(x)	CCD	34	15	44.12%
	T(x) s.t. F(x)	CCD	15	7	46.67%
	T(x)	CCD	13	5	38.46%
	Overall		106	50	47.17%

Table 4.2 GPS: Search Surrogates Usage

and quicker convergence time. However, the different search surrogates also show that quality is better than quantity. The search surrogate based on function values with a time constraint (3.3) provided a lower objective function value more often than the surrogate function based on time with a function value constraint (3.5), but (3.5) converges about 60 minutes faster. Table 4.2 shows the ratio between the number of times the surrogate was used and the number of times it resulted in a point with a lower function value.

Figure 4.3 illustrates the quality in the decrease of the objective function. Each color in Figure 4.3 represents a different search surrogate and each shape demonstrates whether the poll or search step found the improved function value. The figure shows that the surrogate objective function based on computational time provides a lower function value quicker than the surrogate objective function based on function values, leading to a faster convergence sequence.

4.3 Test 2: Barrier Flow

The second test problem applies the Navier-Stokes equations to a vertical flow of fluid with a barrier near the top of the given two-dimensional region. At time zero, the fluid is at rest. When time starts, an initial vertical force is applied to the top of the fluid forcing a vertical downward reaction. The vertical force on the fluid passes a horizontal barrier plane, causing the fluid to move around the barrier and to sway horizontally after passing the barrier, while maintaining the downward

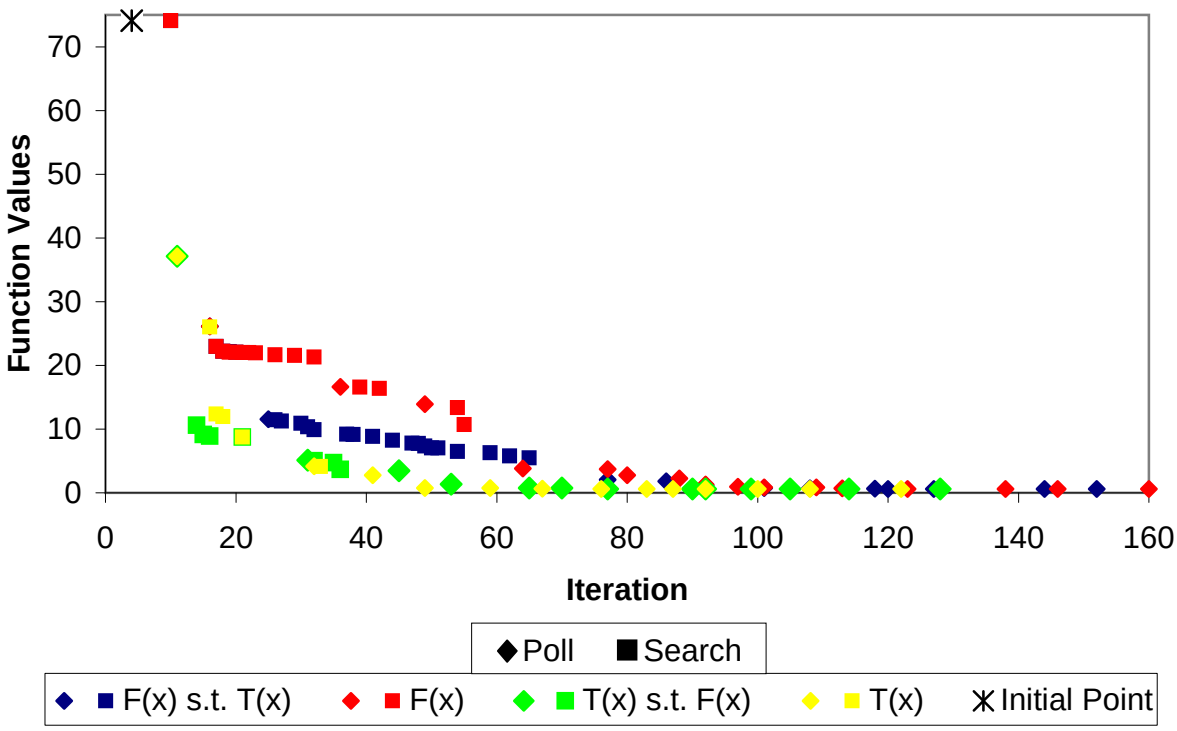

Figure 4.3 Decreasing Function Value

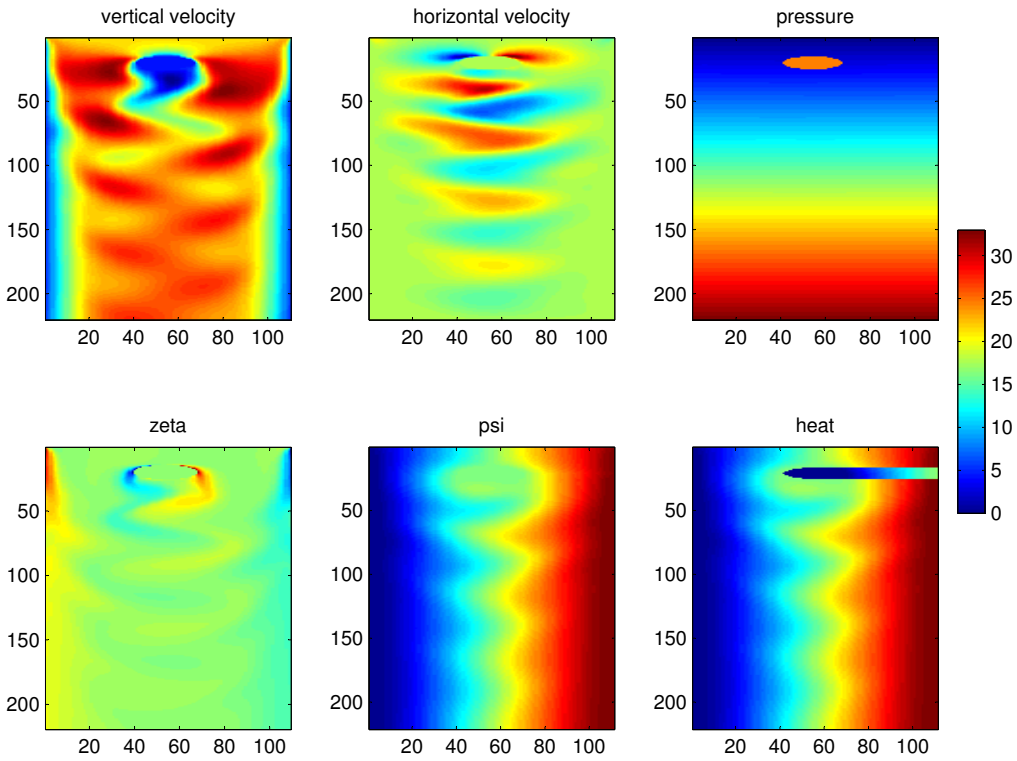

Figure 4.4 Numerical Simulation of Fluid Barrier

reaction. Figure 4.4 illustrates this reaction. For different input parameters, the fluid's vertical and horizontal flow in the region demonstrates different states that can be compared. For this problem, in addition to the two variables of Reynolds number and simulation length, the Prandtl number and initial vertical velocity are added as variables. The goal again is to determine the parameters of a reference image.

4.3.1 Results–GPS

The results for each case are shown in Table 4.3. The runs were conducted in a similar manner as the previous problem, but the results do not exhibit similar behavior as seen in the previous test problem. The base case using a center initial

Full Time	Search	Initial Point(s)	Objective Value	Initial Velocity	Reynolds Number	Prandtl Number	Simulation Length	Number of Iterations	Number of Evaluations	Total CPU Time (min)
	None	Center	1074.07	1.88	140.00	0.16	76.00	32.00	160.00	540.13
	None	CCD	1310.82	1.56	140.00	0.16	72.00	22.00	155.00	476.00
Cut Time										
	None	Center	1074.07	1.88	140.00	0.16	76.00	32.00	160.00	541.05
	None	CCD	1310.82	1.56	140.00	0.16	72.00	22.00	155.00	475.12
	F(x) s.t. T(x)	CCD	1310.82	1.56	140.00	0.16	72.00	22.00	170.00	529.86
	F(x)	CCD	1310.82	1.56	140.00	0.16	72.00	22.00	170.00	529.62
	T(x) s.t. F(x)	CCD	1222.01	1.56	175.56	0.16	87.88	42.00	254.00	778.11
	T(x)	CCD	1121.07	2.00	154.86	0.20	100.00	26.00	170.00	538.85
		Solution	0.00	1.00	160.00	0.10	40.00	-	-	-

Table 4.3 GPS: Barrier Flow Results

point discovered the lowest function value and took the least amount of time. None of the evaluations found the optimal parameters.

Figure 4.5 illustrates a partial correlation between the function value and time for the function evaluations using no search and $t_{cut} = \infty$. It also illustrates the small room for improvement in the overall computational time with image comparison times between a fraction of a second to six seconds. A comparison of GPS with no search shows that with a center initial point, t_{cut} takes an additional 1 minute to converge, but with a CCD set of initial points, t_{cut} saves 1 minute. The variability in computational time overcame any time saved by the t_{cut} parameter. The variability in the computational time can also be seen with the search surrogates using an objective surrogate function based on function values with and without a constraint. In these cases the search and poll steps evaluated the objective function at exactly the same points during the runs, but the computational time was off by only 0.2 minutes.

The effectiveness of the surrogates in finding improved points is shown in Table 4.4. Only two points provided an improved function value and both used a surrogate objective function based on time. The points led to a lower objective function value than if they were not used, demonstrating the use of surrogate functions based on

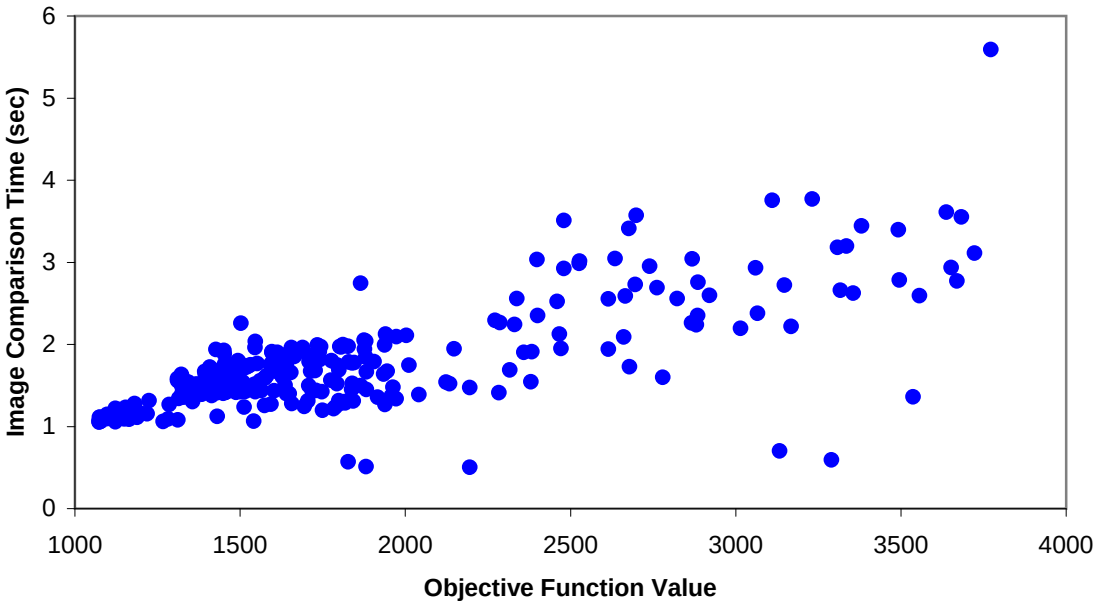

Figure 4.5 Test Problem 2: Function and Time Correlation

Cut Time	Search	Initial Points	Number of Search Surrogates Implemented	Number of Lower Objective Function Value	Percentage
	F(x) s.t. T(x)	CCD	16	0	0.00%
	F(x)	CCD	16	0	0.00%
	T(x) s.t. F(x)	CCD	15	1	6.67%
	T(x)	CCD	16	1	6.25%
	Overall		63	2	3.17%

Table 4.4 GPS: Search Surrogate Usage

time may be more appropriate. Surrogate functions based on objective function values never provided an improved point and thus merely wasted computational time.

4.3.2 Results–MADS

Table 4.5 shows similar results using the MADS algorithm, which are no better than those of GPS. The lowest objective function value was from the MADS case which used function values for the objective surrogate with no constraint (3.2).

Full Time	Search	Initial Point(s)	Objective Value	Initial Velocity	Reynolds Number	Prandtl Number	Simulation Length	Number of Iterations	Number of Evaluations	Total CPU Time (min)
	None	Center	1307.81	1.63	200.00	0.19	84.00	16.00	84.00	227.32
	None	CCD	1261.07	1.69	204.00	0.16	64.00	12.00	92.00	225.77
Cut Time										
	None	Center	1307.81	1.63	200.00	0.19	84.00	16.00	84.00	228.08
	None	CCD	1261.07	1.69	204.00	0.16	64.00	12.00	96.00	226.11
	F(x) s.t. T(x)	CCD	1401.52	1.56	140.00	0.16	72.00	8.00	83.00	246.67
	F(x)	CCD	1219.01	1.56	174.00	0.14	72.19	12.00	112.00	330.50
	T(x) s.t. F(x)	CCD	1399.52	1.56	144.16	0.16	71.77	10.00	86.00	287.32
	T(x)	CCD	1392.17	1.56	132.00	0.16	71.25	10.00	89.00	273.12
		Solution	0.00	1.00	160.00	0.10	40.00	-	-	-

Table 4.5 MADS: Barrier Flow Results

Cut Time	Search	Initial Points	Number of Search Surrogates Implemented	Number of Lower Objective Function Value	Percentage
	F(x) s.t. T(x)	CCD	8	0	0.00%
	F(x)	CCD	12	0	0.00%
	T(x) s.t. F(x)	CCD	10	1	10.00%
	T(x)	CCD	10	0	0.00%
	Overall		40	1	2.50%

Table 4.6 MADS: Search Surrogate Usage

Unfortunately, the random polling directions generated by MADS was the determining factor in finding an improved solution, instead of the surrogates. The random polling directions for each MADS evaluation were set to a specific random seed, but the recursive call of MADS during the surrogate optimization process prevented the use of the same directions for each polling step. Table 4.6 shows that the only surrogate to provide an improved function value at any time was the constrained time surrogate (3.5), and it turned out to be of little benefit to the overall process. The other surrogates were only different versions of MADS with a CCD set of initial set of points and no search. The quicker convergence occurred because the mesh size parameter is reduced at twice the rate of GPS, which is necessary for the MADS convergence theory.

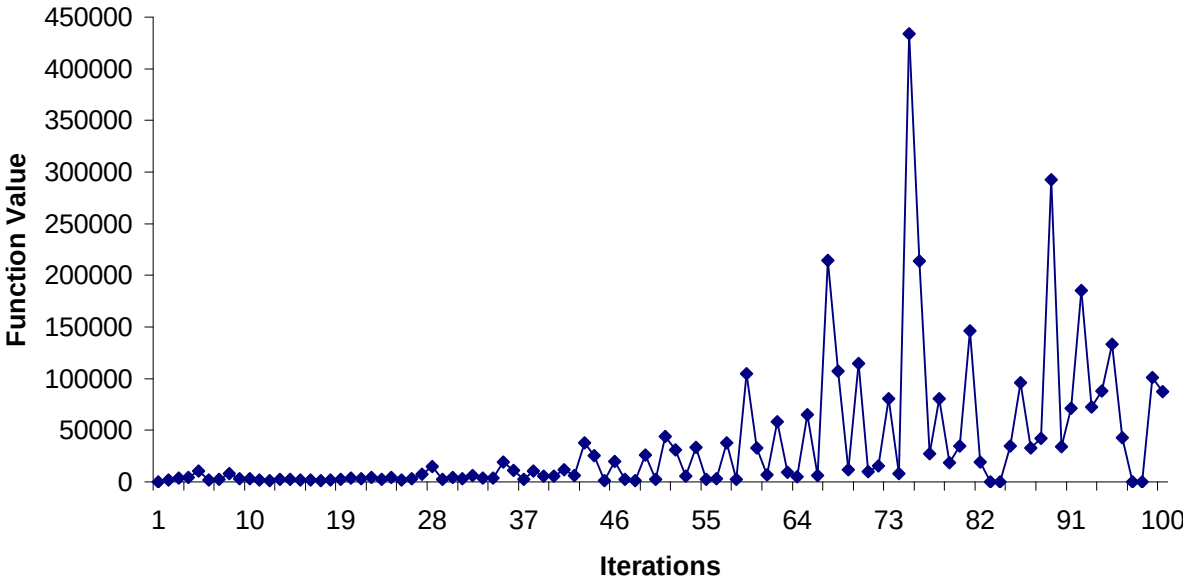

Figure 4.6 GPS: Problem 2 Function Values

4.3.3 Results–Solution

The results of the barrier flow test problem using both GPS and MADS call into question the effectiveness of the MADS–TIME algorithm. To investigate this further, an additional run was made using the base case scenario, but started at the solution. This allows for an evaluation of points neighboring the solution to analyze the fluctuations in the function values and computational time.

Figure 4.6 illustrates how impractical it is to find the optimal solution to this problem. As the mesh size decreases, the function values dynamically increase even though the neighboring points are approaching the solution. This trend leads the algorithm away from the optimal solution. Figure 4.7 illustrates that the image registration times do not follow the same pattern. As the iterations continue, the image registration times become less dynamic.

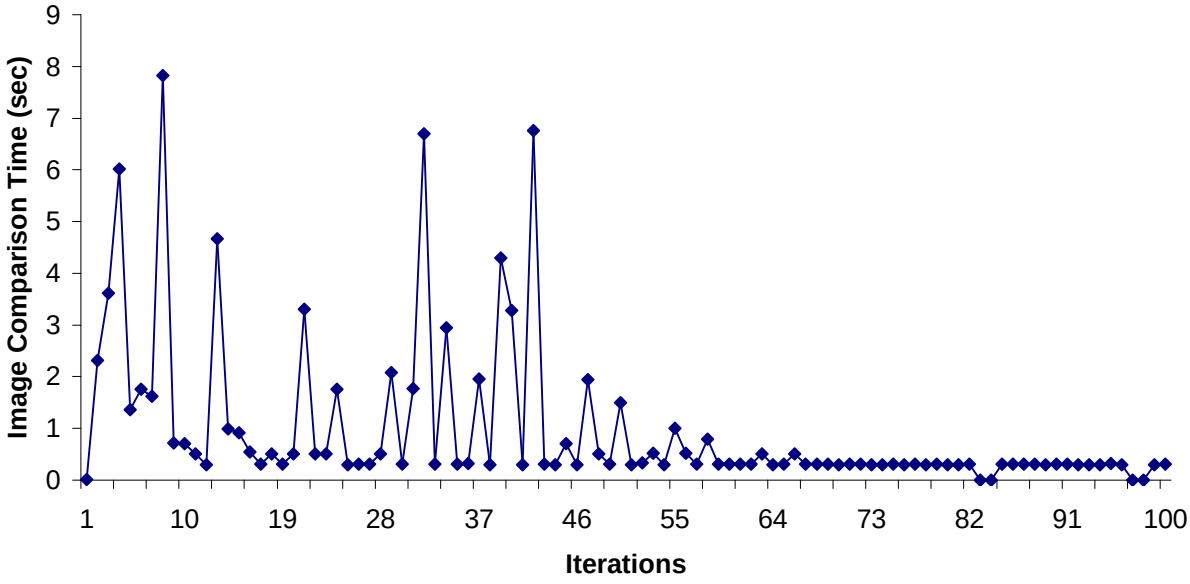

Figure 4.7 GPS: Problem 2 Time

4.4 Review

The algorithm presented in Chapter 3 was applied to two test problems with mixed results. The first problem was successfully solved while the second was not solved. The second problem ,however, turned out to be an extremely difficult problem. Chapter 5 concludes with comments on solving nonlinear optimization problems and suggestions for future work on the implementation of the algorithm.

5. Conclusions and Future Research

The research in this thesis provides an initial investigation in solving a optimization problems in which the objective function values and computational times correlated. Further study into the solution and parameters established can lead to improvement in the application of the algorithm. This chapter offers some concluding remarks and suggests some potential areas of future research.

5.1 Conclusion

The two problems studied in Chapter 4 show an underlying complexity in nonlinear optimization; surrogates and pattern search methods are highly influenced by the behavior, or fluctuations, in the objective function. The test problems possessed the function value and time correlation, but only to a point (see Figures 4.1 and 4.5). The effectiveness of the algorithm on two test problems demonstrates mixed results.

The implementation of the t_{cut} parameter was a useful way to reduce the time to solve the optimization problem. The first test problem showed that this parameter can hinder the performance if the time assumption does not hold, but saves time if the time assumption does hold. The amount of time saved appears to be determined by the problem's time property. The majority of the time required to execute the objective function for the first problem is used by the image comparison algorithm, but for the second problem, the fluid simulation time contains the majority of the computational time. While t_{cut} can be used to stop the image comparison algorithm, it cannot stop the numerical simulation, since the image comparison requires the image obtained from the full simulation.

Dynamic volatility and fluctuations in the nonlinear objective function can greatly impact pattern search methods, not allowing the optimal solution to be found. Surrogate functions were implemented in the search step to help lead the

iterative process quickly to a region containing the optimal solution. However, with dynamic fluctuations, surrogate functions require more points in strategically located positions to effectively mimic the behavior of the objective function. The first test problem does not contain too many fluctuations, since the optimal solution was found by GPS with and without the different search surrogates. The time function seems less dynamic and generates a quicker path to the region containing the optimal solution, with or without the function value surrogate constraint.

The second problem does not seem to have the same simple behavior. For GPS and MADS, the surrogate objective function based on function values never provides an improved function value. The surrogates based on time did provide improved function values, and a study of the iteration sequence shows evidence that the objective function value and time surrogate functions can lead the iteration sequence in completely different paths. While the surrogate constraint based on time rarely impedes the iteration sequence to solve the surrogate based on function values, the opposite cannot be implied. That is for both GPS and MADS, the surrogate constraint based on objective function values frequently restricts the directions available to the surrogate based on time. The implementation of the two surrogates in different variations questions which value, objective function values or computational time, can better represent the true behavior of the objective so that quicker progress can be achieved toward the optimal solution. The main reason for using one surrogate over the other should be based on whether or not the fluctuations are expected to be less for one output than the other, while still correctly mimicking the behavior of the true objective function. The surrogate based on the less dynamic output has a better chance of correctly identifying a better point. A poor surrogate objective function fails to generate good trial points, while a poor surrogate constraint may hinder the adequate exploration of the domain.

5.2 Future Research

The merging of different methods to synthesize a new technique for solving this class of problems has provided new areas of research. Different parameter settings and modifications to the algorithm could provide a better solution in less computational time. Simple changes, such as solving all the proposed surrogate optimization problems, (3.2)–(3.5), in the search step could lead to quicker convergence. Each choice for the algorithm was made based on the current literature, but due to the unique application, the most effective parameter may have not been used.

The parameter t_{cut} provided an effective and efficient method to reduce the time required to find a solution. This parameter is controlled by the user and the approach used in this thesis was simple and based on trial and error. An enhancement to the algorithm can be made by developing a more systematic was of using this parameter to overcome or detect deficiencies illustrated in Figure 4.1 to minimize the time to solve the optimization problem. Since function values and computational times are stored in order to construct surrogates, they can also be used to measure the correlation.

The DACE surrogates implemented had many choices, including the degree of the polynomial, initial points, and the correlation function. These choices led to a surrogate yielding varied results. A higher degree polynomial may have improved the effectiveness of the surrogate, but the degree of the polynomial dictates how may initial points are needed and the number of evaluations to solve the surrogate, both of which take time to evaluate. A balance between the number of the initial points evaluated and the degree of the polynomial may be reached through different experimental designs such as a face center cube or Plackett-Burman (see Myers and Montgomery [25]).

The choice of a Gaussian correlation function $R(\theta_j, |d_j|)$ to determine the amount of influence between known and unknown points was chosen based on recom-

mendations from the literature, but one of the other choices may be more appropriate for this class of problems. Particular choices of the correlation function, such as the exponential function, have a high influence factor on an unknown point based on known points and may enhance the DACE surrogate's ability to imitate the desired behavior.

Not using the surrogates when ill-conditioning, occurs as a result of the clustering of points during the pattern search method was a simple tactic. However, a more effective means to combat this problem may be the use of a trust region in which the search for an improved point is limited to a subregion, $\hat{\Omega} \in \Omega$. A framework for using trust regions for search approximation models can be seen in the work of Alexandrov *et al.* [5]. Similar to this approach is the idea of limiting the number of points used to generate the DACE surrogate to a trust region based on the frame size Δ_k^p or mesh size Δ_k^m. A relationship between the number of points within a region and the distance between the points would dictate the size the trust region. These two factors are considered because the number of points affects the desire in mimic the true function and the distance between points creates the ill-conditioning for $R(\theta, x, x)^{-1}$ (see Section 3.3.2). Eliminating points extremely far apart through the use of the trust region would ease the ill-conditioning.

Including a trust region or different parameter settings could potentially improve performance for this class of problems, but it could also be applied to problems with the opposite behavior; i.e. when the computational time increases as the optimal solution is approached. Implementations such as this, or research into different parameters, may possibly lead to great strides in solving these types of CPU-time correlated functions.

Bibliography

1. Abramson, M.A. "NOMADm optimization software". http://www.afit.edu/en/ENC/Faculty/MAbramson/NOMADm.html. 2007.
2. Abramson, M. A. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems.* Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, August 2002.
3. Abramson, M. A. "Second-order behavior of pattern search". *SIAM Journal on Optimization*, 16(2):315–330, 2005.
4. Abramson, M. A. "Second-order convergence of mesh adaptive direct search". *SIAM Journal on Optimization*, 17(2):606–619, 2006.
5. Alexandrov, N., J. E. Dennis, Jr., R. M. Lewis, V. Torczon. "A Trust Region Framework for Managing the Use of Approximation Models in Optimization" NASA/CR 201745 ICASE Report No 97-50, 1997.
6. Asaki, T. "Elasticity-Based TSWarp Cost Functions" Los Alamos National Laboratory Report, 2004.
7. Audet, C. and J. E. Dennis, Jr. "Analysis of Generalized Pattern Searches". *SIAM Journal on Optimization*, 13(3):889–903, 2003.
8. Audet, C. and J. E. Dennis, Jr. "Pattern Search Algorithms for Mixed Variable Programming". *SIAM Journal on Optimization*, 11(3):573-594, 2000.
9. Audet, C. and J. E. Dennis, Jr. "A Pattern Search Filter Method for Nonlinear Programming without Derivatives". *SIAM Journal on Optimization*, 14(4):980-1010, 2004.
10. Audet, C. and J. E. Dennis, Jr. "Mesh Adaptive Direct Search Algorithms for Constrained Optimization". *SIAM Journal on Optimization*, 17(2):188–217, 2004.
11. Bandler, J. W., Q. Cheng, S. Dakroury, A. S. Mohamed, M.H. Bakr, K. Madsen, J. Søndergaard. "Space Mapping: The State of The Art". *IEEE Transactions on Microwave Theory and Techniques*, 52(1): 337–361, 2004.
12. Booker, A.J. "Well-Conditioned Kriging Models for Optimization of Computer Simulations". Technical Report, M&CT-TECH-00-002, The Boeing Company, 2000.
13. Booker, A.J., J.E. Dennis, Jr, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. "A Rigorous Framework for Optimization of Expensive Functions by Surrogates". *Structural Optimization*, 17(1): 1–13, 1998.

14. Coope, I. D. and C. J. Price. "On the convergence of grid-based methods for unconstrained optimization". *SIAM Journal on Optimization* 11(4): 859–869, 2001

15. Davis, C. "Theory of Positive Linear Dependence". *American Journal of Mathematics*, 76(4): 733–746, 1954

16. Dennis, Jr., J. E. and V. Torczon. "Direct Search Methods on parallel machines". *SIAM Journal Optimization*, 1 448–474, 1991.

17. Griebel, M., T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics: a practical introduction.* SIAM, 1998

18. Hooke, R. and T. A. Jeeves. "Direct Search Solution of Numerical and Statistical Problems". *Journal of the Association for Computing Machinery*, 8(2):212229, 1961.

19. Lewis, R. M. and V. Torczon. "Pattern Search Methods for bounded constrained minimization". *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

20. Lewis, R. M. and V. Torczon. "Pattern Search Methods for Linearly Constrained Minimization". *SIAM Journal on Optimization*, 10(3):917-941, 2000.

21. Lewis, R. M. and V. Torczon. "A globally convergent augmented Lagrangian Pattern search algorithm for optimization with general constraints and simple bounds". *SIAM Journal on Optimization*, 12(4):1075–1089, 2002.

22. Lophaven, S.N., H.B. Nielsen, and J. Søndergaard. "Aspects of the MATLAB Toolbox DACE". Report IMM-REP-2002-13, Informatics and Mathematical Modelling, Danish Technical University, 2002.

23. Lophaven, S. N., Nielsen, H. B., and J. Søndergaard. "DACE Surrogate Models". http://www2.imm.dtu.dk hbn/dace.

24. Modersitzki, J. *Numerical Methods for Image Registration*, Oxford University Press, 2004.

25. Myers, R. H., and D. C. Montgomery. *Response Surface Methodology.* John Wiley and Son, New York, NY, 2002.

26. Robinson, T. D., Eldred, M. S., Willcox, K. E., R. Haimes *Strategies for Multifidelity Optimization with Variable Dimernsional Hierarchical Models* Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (2nd AIAA Multidisciplinary Design Optimization Specialist Conference), Newport, Rhode Island, May, 2006.

27. Sacks, J., W. J. Welch , T. J. Mitchell and H. P. Wynn. "Design and Analysis of Computer Experiments". *Statistical Science*, 4(4):409–435, 1998.

28. Santner, T. J., B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments.* Springer-Verlag, New York, NY, 2003.

29. Schmit Jr., L. A., and H. Miura. "Approximation concepts for efficient structural synthesis". Technical Report CR-2552, NASA, 1976.

30. Torczon, V. "On the Convergence of Pattern Search Algorithms". *SIAM Journal on Optimization*, 7(1):125, 1997.

Appendix A. Appendix 1

The following Appendix contains MATLAB® code for executing the MADS–TIME algorithm (3.2). The files work in conjunction with the NOMADm [4] and DACE [23] software packages.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION PROBLEM: Parameter File setup for fixed and recorded
% variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  Param
%    .count          : Iterate Number
%    .sear           : Number of Times the Search Surrogate is used
%    .nosear         : Number of Times the Search Surrogate is not used
%    .data           : Fixed Parameter Input
%    .x              : Variable Parameter Input
%    .tc             : Time Cut Off Marker
%    .LB             : Parameter Lower Bound
%    .UB             : Parameter Upper Bound
%    .thetaint       : Initial Theta estimate
%    .reg            : Degree of regression polynomial for surrogates
%    .corr           : Correlation matrix for surrogates
%    .stop           : Time Threshold
%    .kon            : Condition Number
%    .konnum         : Condition Number Threshold
%    .f              : Output Function Value
%    .twarp          : Output Time Value
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Param = cpuproblem_Param

% Initialize Data
Param.count=0;
Param.sear=0;
Param.nosear=0;
Param.data=struct2cell(load('refimage0.mat'));
Param.x=0;
Param.tc=Inf;
Param.LB = [];
Param.UB = [];
Param.f=0;
Param.twarp=0;
Param.cfd=0;
Param.stop=0;
Param.konnum=10000;
Param.kon=0;
Param.thetaint=10*ones(1,length([Param.LB]));
Param.reg=@regpoly2;
Param.corr=@corrgauss;

setappdata(0,'PARAM',Param);
return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION PROBLEM: Objective Function File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  x                : Variable Parameter Input
%  fx               : Output Function Value
%  tx               : Output Time Value
%  f_max            : Largest Function Value
%  Param
%   .count          : Iterate Number
%   .data           : Fixed Parameter Input
%   .x              : Variable Parameter Input
%   .tc             : Time Cut Off Marker
%   .f              : Output Function Value
%   .twarp          : Output Time Value
%   .f_min          : Current Lowest Function Value
%   .go             : Current Time of Lowest Function Value
%   .initial        : Number of Initial Points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [fx] = cpuproblem(x)
Param = getappdata(0,'PARAM');
Param.count=Param.count+1;

% Setup Time Cut Off Marker if all initial points already evaluated
if Param.count>Param.initial
    [f_min,call]=min(Param.f(1:(Param.count-1)));
    Param.go=(Param.twarp(call));
    Param.tc=2*Param.go+.1;
    f_max=max(Param.f(1:Param.initial));
end

% User Supplied Objective Function
[fx,tx]=costfunc(Param.data,Param.tc,x);

% Negative Time Value is
if tx < 0
    fx=f_max;
    tx.warp=Param.tc;
end

% Record Necessary Information and Store
Param.f(Param.count)=fx;
Param.x(Param.count)=x;
Param.twarp(Param.count)=tx;
Param.fmin(Param.count)=f_min;

setappdata(0,'PARAM',Param);
return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION PROBLEM: Initial Points File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  Param
%    .LB             : Parameter Lower Bound
%    .UB             : Parameter Upper Bound
%    .initial        : Number of Initial Points
%  iterate           : List of Initial Points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function iterate = cpuproblem_x0
Param = getappdata(0,'PARAM');

% Central Composite Design for n Dimensions
[n,m]=size(Param.LB);
W=zeros(1,n);
X=sqrt(n)*eye(n);
Y=-sqrt(n)*eye(n);

Z=[-1 -1; 1 -1; -1 1; 1 1 ];
for k=1:n-2
    Z=[Z,ones(size(Z,1),1)*-1;Z,ones(size(Z,1),1)];
end
DOE=[W;X;Y;Z];

% Normailze Data to Upper and Lower Bounds of Problem
[a,b]=size(DOE);
for j=1:a
    for k=1:n
        iterate(j).x(k,1)=((Param.UB(k,1)-
Param.LB(k,1))/(2*sqrt(n)))*((DOE(j,k)+sqrt(n)))+Param.LB(k,1);
        iterate(j).p={};
    end
end
Param.initial=length([iterate.x]);

setappdata(0,'PARAM',Param);
return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION PROBLEM: Bounds File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  n                 : Problem Dimension
%  Param
%    .LB             : Parameter Lower Bound
%    .UB             : Parameter Upper Bound
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [A,l,u] = cpuproblem_Omega(n)
Param = getappdata(0,'PARAM');

A = eye(n);
l = Param.LB;
u = Param.UB;
return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SEARCH SURROGATE: MADS Search Surrogate File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  Problem          : Input Parameters Required for MADS
%  Options          : Input Parameters Required for MADS
%  Param
%   .sear           : Number of Times the Search Surrogate is used
%   .nosear         : Number of Times the Search Surrogate is not used
%   .go             : Current Time of Lowest Function Value
%   .stop           : Time Threshold
%   .kon            : Condition Number
%   .konnum         : Condition Number Threshold
%   .thetaint       : Current Surrogate Theta Solution
%  pcenter          : Current Solution Variables
%  BestF            : Solution to Search Surrogate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function S = searchsurrogate(pcenter);
Param = getappdata(0,'PARAM');

% If the Solution Time and Condition Number are within Thresholds,
% execute Search Surrogate

if (Param.go>Param.stop) && (Param.kon<Param.konnum)
    Param.sear=Param.sear+1;

    Defaults = mads_defaults('Surrogate');
    Options  = Defaults.Options;
    Problem.nameCache   = 'SCACHE';
    Problem.typeProblem = 'STRUTH';
    Problem.File.F = 'mysearch';
    Problem.File.O = 'mysearch_Omega';
    Problem.File.X = 'mysearch_X';
    Problem.File.I = 'mysearch_x0';
    Problem.File.N = 'mysearch_N';
    Problem.File.P = 'mysearch_Param';
    Problem.File.C = 'mysearch_Cache.mat';
    Problem.File.S = 'mysearch_Session.mat';
```

```
Problem.File.H = 'mysearch_History.txt';
Problem.File.D = 'mysearch_Debug.txt';
Problem.fType  = 'M';
Problem.nc     = 0;

% Specify Choices for SEARCH
Options.nSearches          =0;
Options.pollStrategy    = 'Standard_2n';
Options.pollOrder       = 'Consecutive';
Options.pollCenter      = 0;
Options.pollComplete    = 0;
Options.NPollComplete   = 0;
Options.EPollComplete   = 0;

% Specify Termination Criteria
Options.Term.delta      = 1e-4;
Options.Term.nIter      = Inf;
Options.Term.nFunc      = 50000;
Options.Term.time       = Inf;
Options.Term.nFails     = Inf;

% Choices for Mesh Control
Options.delta0          = 0.1;
Options.deltaMax        = Inf;
Options.meshRefine      = 0.5;
Options.meshCoarsen     = 2.0;

% Choices for Filter management
Options.hmin            = 1e-8;
Options.hmax            = 1.0;

% MADS flag parameter values
Options.loadCache        = 1;
Options.countCache       = 1;
Options.runStochastic    = 0;
Options.scale            = 2;
Options.useFilter        = 1;
Options.degeneracyScheme = 'random';
Options.removeRedundancy = 1;
Options.computeGrad      = 0;
Options.saveHistory      = 0;
Options.plotHistory      = 0;
Options.plotFilter       = 0;
Options.plotColor        = 'k';
Options.debug            = 3;
Options.Sur.Term.delta   = 0.01;
```

```
    % Create DACE Surrogates
    [midthetaint,minkon] = mysearch_Param;
    Param.thetaint=midthetaint;
    Param.kon=minkon;

    iterate0.x = pcenter.x;
    iterate0.p={};

    % Execute MADS to Solve Search Surrogate
    [BestF,BestI,RunStats,RunSet] = mads(Problem,iterate0,Options);

    S.x=BestF.x;
    S.p={};
else
    S.x=pcenter.x;
    S.p={};
    Param.nosear=Param.nosear+1;
end

setappdata(0,'PARAM',Param);
return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SEARCH SURROGATE: Create DACE Surrogates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  kon              : Condition Number
%  Param
%   .x              : Variable Parameter Input
%   .f              : Output Function Value
%   .twarp          : Output Time Value
%   .LB             : Parameter Lower Bound
%   .UB             : Parameter Upper Bound
%   .thetaint       : Initial Theta estimate
%   .reg            : Degree of regression polynomial for surrogates
%   .corr           : Correlation matrix for surrogates
%   .kon            : Condition Number
%   .theta          : Initial Theta estimate
%   .lob            : Theta Lower Bound
%   .upb            : Theta Upper Bound
%   .dmodelF        : DACE Function Value Surrogate
%   .dmodelT        : DACE Time Value Surrogate
%       .C          : Cholesky Decomposition
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [midthetaint,minkon] = mysearch_Param
Param = getappdata(0,'PARAM');
kon=0;
```

```
% Determine Lower Bound on Theta
while kon < 10e3
    [dmodel]=dacefit(Param.x, Param.f, Param.reg, Param.corr,
Param.thetaint);
    kon= condest(dmodel.C);
    Param.thetaint=Param.thetaint/2;
end
theta0=Param.thetaint;
Param.lob=2*theta0;

% Determine Lower Bound on Theta
dLnorm = inf;
h = 0.1;
while dLnorm > 1e-2
    theta1=theta0+h*ones(1,length(theta0));
    dmodel0=dacefit(Param.x,Param.f,Param.reg, Param.corr, theta0);
    dmodel1=dacefit(Param.x,Param.f,Param.reg, Param.corr, theta1);
    C0=dmodel0.C;
    R0=C0*C0';
    R1=dmodel1.C*dmodel1.C';
    dR = (R1 - R0)./h;
    G=dmodel0.gamma';
    S=dmodel0.sigma2;
    dL=(G'*dR*G)/(2*S)-trace(C0'\(C0\dR));
    dLnorm = norm(dL);
    theta0=2*theta0;
end
Param.upb=theta0/2;
Param.theta = (Param.lob+Param.upb)./2;

% Compute DACE models for Function Value and Time
[Param.dmodelF]=dacefit(Param.x,Param.f,Param.reg,Param.corr,Param.thet
a,Param.lob,Param.upb);
[Param.dmodelT]=dacefit(Param.x,Param.twarp,Param.reg,Param.corr,Param.
theta,Param.lob,Param.upb);
midthetaint=Param.dmodelF.theta;

% Compute the Condition Numbers of the Surrogates
kon1= condest(Param.dmodelF.C);
kon2= condest(Param.dmodelT.C);
minkon=min(kon1,kon2);

setappdata(0,'PARAM',Param);
return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SEARCH SURROGATE: Surrogate Objective Function File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Variables:
%  x                : Surrogate Variable Parameter Input
%  F                : Surrogate Output Function Value
%  Param
%   .dmodelF        : DACE Function Value Surrogate
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function F=mysearch(x)
Param = getappdata(0,'PARAM');

% Evaluate DACE Surrogate for Variable
[F]=predictor(x,Param.dmodelF);

setappdata(0,'PARAM',Param);
return


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SEARCH SURROGATE: Surrogate Constraint File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variables:
%  x                : Surrogate Variable Parameter Input
%  T                : Surrogate Constraint Output Function Value
%  Param
%   .dmodelT        : DACE Time Value Surrogate
%   .go             : Current Time of Lowest Function Value
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function isfeasible=mysearch_X(x)
Param = getappdata(0,'PARAM');

% Evaluate DACE Surrogate for Variable
T=predictor(x,Param.dmodelT);

% Determine if point meets constraint
if T <= Param.go*2+.1
    isfeasible=1;
else
    isfeasible=0;
end

setappdata(0,'PARAM',Param);
return



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SEARCH SURROGATE: Surrogate Bounds File
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Variables:
%  n                : Problem Dimension
%  Param
%   .LB             : Parameter Lower Bound
%   .UB             : Parameter Upper Bound
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [A,l,u] = mysearch_Omega(n)
Param = getappdata(0,'PARAM');

A = eye(n);
l = Param.LB;
u = Param.UB;

return
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
xx–xx–2007	Master's Thesis	Mar 2005 — Mar 2006

4. TITLE AND SUBTITLE

SURROGATE STRATEGIES FOR COMPUTATIONALLY EXPENSIVE OPTIMIZATION PROBLEMS WITH CPU-TIME CORRELATED FUNCTIONS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Magallanez Jr, Raymond Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENC/07-01

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Los Alamos National Labaratory
Bikini Atoll Rd, SM 30
Los Alamos, NM 87545

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research focuses on numerically solving a class of computationally expensive optimization problems that possesses a unique characteristic: as the optimal solution is approached, the computational time required to compute an objective function value decreases. This is motivated by an application in which each objective function evaluation requires both a numerical fluid dynamics simulation and an image registration and comparison process. The goal is to find the parameters of a predetermined image by comparing the flow dynamics from the numerical simulation and the predetermined image through the image comparison process. The generalized pattern search and mesh adaptive direct search methods were applied in a way that employs surrogate functions in the search step to reduce the number of costly function evaluations. The surrogate functions are formed, based on either previous function values or their computational times, or both. The solution to the surrogate optimization problem can be solved easily and provides an improved solution quickly. A time cut-off parameter was also added to the objective function to allow its termination during the comparison process if the computational time exceeded a specified threshold. The approach was tested on two problems using the NOMADm and DACE MATLAB® software packages, and results are presented.

15. SUBJECT TERMS

Time Correlation, Generalized Patter Search, Mesh Adaptive Directed Search, Surrogates, Design and Analysis of Computer Experiments

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt. Col. Mark A. Abramson, Phd, (ENC)
U	U	U	UU	74	19b. TELEPHONE NUMBER *(include area code)* (937) 785-3636; e-mail: mark.abramson@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18