THE DIFFUSION EQUATION METHOD FOR GLOBAL OPTIMIZATION AND ITS APPLICATION TO MAGNETOTELLURIC GEOPROSPECTING

BY

REBECCA JEAN HARTMAN–BAKER

B.S., University of Kentucky, 1998

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana–Champaign, 2005

Urbana, Illinois

To Jeff, with love.

Acknowledgments

I owe a debt of gratitude to several institutions and countless people. First, this work was partially supported by National Computational Science Alliance under grants CCR030004N and CCR030002, and utilized the platinum IA-32 and tungsten Xeon clusters at NCSA. In addition, most of the development of the three-dimensional objective function was performed on the CSE Turing Cluster, an Apple G5 cluster machine. As for financial support, I entered the University with the College of Engineering SURGE fellowship, generously funded by several corporate sponsors, to whom I am grateful for their commitment to diversity. In addition, I was supported during my final year of research by funds from the Fulton Watson Copp endowed chair, held by my advisor, and I am appreciative to Mrs. Copp for her generosity.

I owe my success in this endeavor to more people than I can name in this short space. A special thank you to my first family: John, Gale, Rachel, and Laura Hartman. Thanks especially to my dad John for encouraging my interest in mathematics, and for being my number one fan when I was growing up. Thanks also to my babysitter Alice, who was always encouraging and inspiring.

To the teachers who encouraged me to explore mathematics, I owe a debt of gratitude. These teachers include Mrs. Chamberlain, my 5th/6th grade teacher; Mrs. Rummel, my 7th grade Algebra teacher; Mr. Koetke, my 10th grade Pre-Calculus and Computer Science teacher; and Mrs. Caldwell, my 12th grade Calculus II teacher. And I am grateful to Dr. Paul Eakin of the University of Kentucky Department of Mathematics, who helped me to survive my first exposure to calculus. He's the first person whom I ever saw talk about math in an informal manner, with as much ease as talking about the weather. Thanks to him for making mathematics accessible. Other teachers who encouraged my scientific side include Mr. Barber, Mr. Hutchison, and Mrs. Kikuchi. Also Dr. Chappelle of the University of Kentucky Agronomy Department, with whom I did my senior

research project, was incredibly encouraging and positive to me.

As an undergraduate at the University of Kentucky, I majored in Physics. The encouragement of Professors Christopher, Subbaswamy, and MacAdam gave me the courage to pursue summer research at SRI International with Dr. Xiao-An Shen, where I discovered the love of my academic life, scientific computing. Thank you, Xiao-An, for your patience and help. Also thank you to Professor Greg Wasilkowski, with whom I had my first numerical analysis course, who first gave me the idea to pursue computer science in graduate school, and to Professor Robert Reams, who taught me nearly everything I know about linear algebra.

In graduate school, many professors greatly impacted my life. First and foremost, my advisor, Mike Heath, is due a lot of credit for putting up with me through both the high and the low points in my journey. I am grateful for his endless patience and unwaivering support. Thanks also to Professor Paul Saylor, one of the kindest people I know. Professor Skeel's honesty and support have strengthened my confidence. Professor Bond's kindness and genuine interest in my work have been very encouraging, too. And Professor Braatz's lofty standards have encouraged me to aim high. Thanks to all the professors, from whom I have learned so much.

My fellow grad students have been an invaluable resource too. Thanks to my favorite office-mate of all time, Ali Pinar, for the academic support and friendship. Thanks also to Vanessa López for her unwavering and honest friendship. Thanks to Mitch Harris for his role in my survival of CS 373, and his subsequent friendship; to David Bunde for being here as long as I have, giving us a chance to form a strong friendship; to Dan Cranston for the fun conversations and the help with finding that pattern for 1-D diffusion; to Chris Siefert, Mike Parks, Hanna VanderZee, Bill Cochran, David Alber, Eric Cyr, Wei Wang, David Hardy, Gaurav Kalmady, Dan Bullok, Ava Jarvis, Robert Engle, Jim Jiao, Ryan Szypowski, Jeff Naisbitt, Tony Hursh, Alison Noble, and all the others who have been so supportive of me.

A long paragraph should be devoted to my advisor's supportive staff, and to his assistant Jodi Gritten–Dorsett in particular. Jodi is the model of efficiency and reliablility, and a great friend. Without her my life in graduate school would not have run so smoothly. She always went above and beyond the call of duty, at times literally feeding me, clothing me, and helping me to straighten

out my life. She saved me from my own cluelessness more than once. Thanks so much, Jodi!

Thanks to David Day and Greg Newman from Sandia for giving me the initial ideas upon which my thesis is based, and the opportunity to experience the national lab environment. Thanks to Sudhakar Pamidighantam and Bruce Loftis for the assistantship at NCSA which sharpened my computing skills, and the friendship which continues beyond my tenure at NCSA.

Thanks also to my karate instructor, Chris Johnston, and my Weight Watchers leader, Shirley Skaggs, who both helped me learn to believe in myself by doing things I never thought I could do. The confidence I gained through karate and weight loss have carried over into the rest of my life, and I appreciate their positive impact.

My life in graduate school was marred by personal tragedy. Without the assistance of professionals, I firmly believe that I would not be here today. So I would like to give my thanks to Dr. Rakhi Sen, Mr. Timothy Vance, Dr. Robert Basler, Connie, Martin, Susanna, Nirmal, Brian, Rachel, Malia, and many more. Also thanks to my husband for taking such good care of me and staying with me despite all the chaos and turmoil; my Dad for never abandoning me and loving me when I was unloveable; my sisters for their unconditional love and helpful natures; Marvis for being a great "bonus mom;" "bonus brother" Austin for listening to my ceaseless ramblings about supercomputers, and actually being interested; Barb, Glen, Julian, and Glenna for always being there for me; Sally for the cheer and moral support; my in-laws for their love and support; and the rest of the family for their good will and positive thoughts.

But most of all, I want to thank the love of my life, Jeffery Allen Hartman–Baker. His encouragement has been invaluable, and without him, I could not have finished. He made the whole journey worth it. Thanks, Jeff. You mean the world to me.

Table of Contents

List of Tables . . . . . **ix**

List of Figures . . . . . **x**

List of Abbreviations . . . . . **xi**

Chapter 1 Magnetotelluric Geoprospecting Problem **1**
1.1 Introduction 1
1.2 Stabilization of Inverse Problem 2
1.2.1 Stabilization Using Tikhonov Regularization 3
1.2.2 Stabilization Using Selection Methods 5

Chapter 2 Solution of Magnetotelluric Geoprospecting Problem **8**
2.1 Formulation of Problem in Two Dimensions 8
2.1.1 Solution of Forward Problem in Two Dimensions 10
2.2 Formulation of Problem in Three Dimensions 10
2.2.1 Solution of Forward Problem in Three Dimensions 12
2.3 Solution of Inverse Problem 13
2.4 Optimization 13
2.4.1 Global Optimization Methods 14

Chapter 3 Diffusion Equation Method for Global Optimization **19**
3.1 Background 19
3.2 Properties of Diffusion 21
3.2.1 Effect of Diffusion on Periodic Functions 22
3.2.2 Effect of Diffusion on Coercive Polynomials 25
3.2.3 Effect of Diffusion on Arbitrary Coercive Functions 27
3.3 Robustness of DEM 36
3.4 Implementation 37
3.4.1 Numerical Implementation 39
3.5 Initialization of DEM 42
3.6 Continuation 45
3.6.1 Continuation Theory 45
3.6.2 Behavior of Extrema under Diffusion 47
3.6.3 Heuristics for Continuation 49
3.7 Local Methods in Inner Loop 50

3.7.1 BFGS . . . 50
3.7.2 Nelder–Mead . . . 50
3.8 Performance of DEM . . . 51
3.8.1 Performance of DEM on Test Functions . . . 51
3.8.2 Comparison of DEM and Brute Force Methods . . . 53

Chapter 4 Implementation of Solution to Geoprospecting Problem . . . 56
4.1 Overview of Algorithm . . . 56
4.2 Computation of Objective Function in Two Dimensions . . . 57
4.3 Computation of Objective Function in Three Dimensions . . . 57
4.3.1 Monte Carlo Integration . . . 58
4.4 Parallelization of Computation of Objective Function . . . 62
4.4.1 Parallelization in Two Dimensions . . . 62
4.4.2 Parallelization in Three Dimensions . . . 63
4.5 Strategies for DEM . . . 64
4.5.1 Determining Optimal Distribution of Work . . . 64
4.6 Parallel Performance Modeling . . . 71
4.6.1 Efficiency of DEM Computation of 2-D Objective Function . . . 71
4.6.2 Model of DEM Computation of 2-D Objective Function . . . 72
4.6.3 Efficiency of DEM Computation of 3-D Objective Function . . . 74
4.6.4 Model of DEM Computation of 3-D Objective Function . . . 75
4.7 Performance on Magnetotelluric Test Problem . . . 79

References . . . 81

Author's Biography . . . 85

List of Tables

3.1 Number of function evaluations required 42
3.2 Comparison of DEM and MAPS on test functions 53

4.1 Comparison of optimal and heuristic work distribution 66
4.2 Pseudoefficiency (2-D objective function) 72
4.3 Pseudoefficiency (3-D objective function) 74
4.4 Performance of DEM on geoprospecting objective function 79

List of Figures

2.1 Example conductivity function $\sigma(x, y)$ 9
2.2 Example conductivity function $\sigma(x, y, z)$ 12
2.3 Example objective function 14

3.1 Example of smoothing 20
3.2 Bernstein Polynomial approximation to Griewank function 31
3.3 Function for which DEM fails 37
3.4 Insufficient diffusion 38
3.5 Recursive finite difference dependencies 39
3.6 Multi-dimensional finite difference stencil 40
3.7 Creating the finite difference matrix 41
3.8 Ratio $D(N, p)/(2N + 1)^p$. 43
3.9 Diffusion causing convexity 43
3.10 Agglomeration of extrema with increased diffusion 48

4.1 Concentration of hypercubes in VEGAS 61
4.2 Three-tier parallelism 65
4.3 Determining optimal work distribution 67
4.4 Scalability of DEM (2-D objective function) 69
4.5 Scalability of DEM (3-D objective function) 70
4.6 Performance model fit (2-D objective function) 73
4.7 Performance model fit (3-D objective function) 76
4.8 Performance model fit (3-D objective function, constant work per processor) 77
4.9 Performance model fit (3-D objective function, constant work per processor, detail) . 78
4.10 Solution of objective function using DEM 80

List of Abbreviations

BFGS Broyden–Fletcher–Goldfarb–Shanno Optimization Method.

DEM Diffusion Equation Method.

ODE Ordinary Differential Equation.

Chapter 1

Magnetotelluric Geoprospecting Problem

1.1 Introduction

In geoprospecting, the conductivity of rock is determined from its effect on electromagnetic waves. From the conductivity, we can deduce the type of rock (or oil or water) and the nature of the geological formation. The forward problem is to solve Maxwell's system of partial differential equations

$$\nabla \times \nabla \times E + \mathrm{i}\omega\mu_0\sigma E = J \tag{1.1}$$

for the electric field E, given the electrical conductivity σ and source vector J, where ω is the angular frequency, and μ_0 is the magnetic permeability of free space. The conductivity could be complex, but in our study we restrict it to have real values. The inverse problem is formulated from the forward problem. Given J and some information about E, we must determine σ. Electromagnetic inverse problems arise in many geological applications, including magnetotelluric geoprospecting, hydrological modeling, and reservoir and environmental characterization [25].

A simpler forward problem is Poisson's equation

$$\nabla \cdot \sigma \nabla \phi = \nabla \cdot J. \tag{1.2}$$

This equation is essentially the conservative components of (1.1) ($E = \nabla\phi$), and is a scalar equation rather than a vector equation. Thus $\nabla\phi$ can still be thought of as an electromagnetic field, but

the mathematics is simpler. Here, we begin with (1.2) in two space dimensions: given $J(x, y)$ and some measurements of $\phi(x, y)$, we must determine $\sigma(x, y)$.

Electromagnetic inverse problems of this kind are *ill-posed,* meaning that the solution is not unique or does not depend continuously on the data [7, p. 4]. Ill-posed problems arise in many fields, including optics, signal processing, acoustics, astronomy, and thermodynamics. If a problem has more than one solution, then the issue is how to select a desired solution from the solution space. One approach is to stabilize the problem by adding information such as smoothness constraints to the problem. Regularization, one such class of stabilizing methods, has commonly been used for these problems.

1.2 Stabilization of Inverse Problem

There are many types of additional information that we could use to stabilize the problem. In least squares methods, a residual and its norm are defined,

$$\rho(\sigma) = \|z - L\sigma\|_2,$$

where L is the operator and z is the right-hand side, and then one of the following methods is used to regularize [7, pp. 10–11]:

1. Minimize $\rho(\sigma)$ subject to the constraint that σ belongs to a specified subset, $\sigma \in S_\sigma$.

2. Minimize $\rho(\sigma)$ subject to the constraint that some measure of the "size" of σ, $\omega(\sigma)$, is less than an upper bound δ, *i.e.,* $\min \rho(\sigma)$ subject to $\omega(\sigma) < \delta$.

3. Minimize $\omega(\sigma)$ subject to a constraint on $\rho(\sigma)$, *i.e.,* $\min \omega(\sigma)$ subject to $\rho(\sigma) < \alpha$.

4. Minimize a linear combination of $\rho(\sigma)^2$ and $\omega(\sigma)^2$, *i.e.,* $\min (\rho(\sigma)^2 + \lambda^2\omega(\sigma)^2)$. This is the most famous method and is known as *Tikhonov regularization.*

Tikhonov regularization is often used in magnetotelluric problems [23, 24, 25, 32, 34]. Newman and Hoversten describe the technique at a high level in [25].

1.2.1 Stabilization Using Tikhonov Regularization

Suppose that we can compute discrete solutions of Maxwell equations for a real and arbitrary 3-D electrical conductivity distribution $\mathbf{m}$, using a linear complex forward mapping operator $\mathbf{f}[\mathbf{m}]$. Then the field observations can be viewed as a function of the model:

$$d^{\text{obs}} = \mathbf{f}[\mathbf{m}] + \epsilon,$$

where d^{obs} and ϵ are complex vectors of length n that comprise the observed data values and measurement noise. The mapping $\mathbf{f}[\mathbf{m}]$ takes the real vector $\mathbf{m}$ of length m and maps it to the complex space of dimension n. Because the inverse problem is underdetermined ($m \gg n$), the process of recovering the model $\mathbf{m}$ is ill-posed. Thus to recover the model, extra information must be added to the problem. Typically, the technique used is to minimize the Tikhonov functional

$$\phi = \frac{1}{2}(\mathbf{F}[\mathbf{m}] - \mathbf{d}^{\text{obs}})^H(\mathbf{F}[\mathbf{m}] - \mathbf{d}^{\text{obs}}) + \frac{\lambda}{2}\|\mathbf{W}(\mathbf{m} - \mathbf{m}_{\text{ref}})\|^2, \tag{1.3}$$

where H denotes the Hermitian transpose operator, $\mathbf{W} \in \mathbb{R}^{m\times m}$ is a smoothing matrix that does not depend on $\mathbf{m}$, $\mathbf{F}[\mathbf{m}]$ and $\mathbf{d}^{\text{obs}}$ are weighted versions of $\mathbf{f}[\mathbf{m}]$ and d^{obs}, $\mathbf{m}_{\text{ref}}$ is the reference model, and λ is the regularization parameter.

$\mathbf{F}[\mathbf{m}]$ and $\mathbf{d}^{\text{obs}}$ are obtained by multiplying a weighting matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ by $\mathbf{f}[\mathbf{m}]$ and d^{obs}, respectively. Typically, $\mathbf{D}$ is diagonal and weights the entries based on the inverse of the standard deviations of the measurements. $\mathbf{W}$ is usually based upon a finite-difference approximation to the Laplacian operator, so minimizing the second term in the Tikhonov functional controls the model curvature.

The regularization parameter $\lambda \geq 0$ determines the degree of smoothing imposed upon the solution. The trade-off is between data fit and model smoothness. A smaller λ yields a better data fit but a rougher model, while a larger λ will result in a smoother model but poorer data fit. Selecting the regularization parameter is a difficult problem with no single best approach. Newman and Hoversten report success with using a continuation approach to the problem: Begin with a large λ, which results in a nearly quadratic objective function, and then reduce λ (thereby placing more

weight on the data fitting requirement) and solve a new minimization problem, reducing λ until the data misfit no longer exceeds a given tolerance [25]. The rationale for using a continuation approach is that a large regularization parameter stabilizes the problem at the outset of the procedure, guarding against spurious, nonphysical models.

Computing the gradient $\nabla\phi$ with respect to the model parameters and setting the resulting expression equal to zero results in a nonlinear equation that must be satisfied for a minimum of (1.3). The kth component of the gradient can be expressed as

$$\frac{\partial\phi}{\partial m_k} = \mathrm{Re}\left\{\frac{\partial\mathbf{F}[\mathbf{m}]^T}{\partial m_k}\left\{\mathbf{F}[\mathbf{m}] - \mathbf{d}^{\mathrm{obs}}\right\}^*\right\} + \lambda\mathbf{w}_k^T\mathbf{W}(\mathbf{m} - \mathbf{m}_{\mathrm{ref}}),$$

where $\mathbf{w}_k$ denotes the kth column of $\mathbf{W}$, T denotes the transpose operator, and Re refers to the real part of the complex argument. Thus we can write $\nabla\phi = 0$ as

$$\mathrm{Re}\left\{\mathbf{J}^T(\mathbf{m})(\mathbf{F}[\mathbf{m}] - \mathbf{d}^{\mathrm{obs}})^*\right\} + \lambda\mathbf{W}^T\mathbf{W}(\mathbf{m} - \mathbf{m}_{\mathrm{ref}}) = 0, \tag{1.4}$$

where $\mathbf{J}^T(\mathbf{m}) = \partial\mathbf{F}[\mathbf{m}]/\partial\mathbf{m} \in \mathbb{C}^{n\times m}$ is the sensitivity matrix, and * represents the complex conjugate operation.

Typically, (1.3) is minimized by using Newton's method to solve (1.4). Newton's method converges quadratically in the neighborhood of a minimum, and for a quadratic (1.3), it will solve (1.4) in one step. In the first steps of continuation, (1.3) is nearly quadratic, and as the regularization parameter is reduced, the starting model obtained from the previous step should be in the neighborhood of the minimum. If this is not the case, it may be necessary to perform a line search with backtracking.

The metric $\omega(\sigma)$ is usually chosen to measure the smoothness of the second derivative. While this choice of metric certainly stabilizes the problem, it does not seem physically realistic for this application, since different types of rock have distinct conductivities and strata have definite boundaries.

Portniaguine and Zhdanov [29] propose a different stabilizing functional. Instead of minimizing the second derivative, they minimize the area where strong model parameter variations and dis-

continuities occur with their *minimum gradient support* (MGS) functional. Using this functional, we seek the minimum of

$$\phi = \frac{1}{2}(\mathbf{F}[\mathbf{m}] - \mathbf{d}^{\text{obs}})^H(\mathbf{F}[\mathbf{m}] - \mathbf{d}^{\text{obs}}) + \frac{\lambda}{2}\|\mathbf{m}\|^2_{w_e}, \tag{1.5}$$

where $\|\cdot\|^2_{w_e}$ is a weighted norm with weighting function

$$w_e(\mathbf{m}) = \frac{f_{\beta\text{MGS}}(\mathbf{m})}{((\mathbf{m},\mathbf{m}) + \epsilon^2)^{1/2}},$$

where ϵ is a small number, $(\mathbf{x},\mathbf{y})$ is the inner product, and

$$f_{\beta\text{MGS}}(\mathbf{m}) = \frac{\nabla\mathbf{m}}{(\nabla\mathbf{m}\cdot\nabla\mathbf{m} + \beta^2)^{1/2}},$$

where β is also a small number.

Using the MGS functional in the regularization results in a more focused image. Portniaguine and Zhdanov report promising results in [29]. In particular, their method is able to reproduce a model containing two small bodies of anomalous conductivity, whereas traditional regularization produces an incorrect solution.

1.2.2 Stabilization Using Selection Methods

Another class of stabilizing methods are known as *selection methods* [33]. The general idea behind selection methods is to restrict the solution of $L\sigma = z$ to some set of possible solutions S, and then minimize the residual in a meaningful manner. If $z \notin LS$, then we are searching for a *quasisolution*. Furthermore, if we search for a solution in a finite dimensional space, then the solution is said to be an *approximate quasisolution* [33].

Other researchers have investigated selection methods for geoprospecting. Marcuello–Pascual, Kaikkonen, and Pous have proposed an inversion algorithm that takes into account the varying geometry in magnetotelluric models by parameterizing the boundaries between strata with curves [17]. They use finite elements with linear shape functions to compute a coefficient matrix, and a

statistical iterative method to determine the parameterization.

M. S. Zhdanov and N. G. Golubev have proposed the Finite Functions Method [36]. In this method, the anomalous conductivity is described by a function of the form

$$f(x) = \begin{cases} 0 & \text{if } x \notin [a,b] \\ A \cdot \phi_{\alpha,\beta}\{t[\tau(x;a,b);p,q]\} & \text{if } x \in [a,b] \end{cases},$$

where A is the amplitude of the function,

$$\phi_{\alpha,\beta}(t) = (1-t)^{\alpha}(1+t)^{\beta}$$

is a smooth function with a maximum equal to 1 within the interval $[-1,1]$,

$$t(\tau;p,q) = \mathrm{Re}\left[-\frac{i}{\pi}\ln\left(\frac{(1+\gamma^*)(e^{i\pi\tau}-\gamma)}{(1+\gamma)(1-\gamma^* e^{i\pi\tau})}\right)\right],$$

and $-1 \le \tau \le 1, \gamma = p+iq, \gamma^* = p-iq, |\gamma| < 1$. The function t creates a continuous mapping of $[-1,1]$ onto itself and is what allows great variation in the form of $\phi_{\alpha,\beta}$. The function $\tau(x;a,b) = (2x-(a+b))/(b-a)$ maps $x \in [a,b]$ onto the segment $[-1,1]$.

The finite function is completely defined by seven parameters. The two-dimensional version of this function is simply the product of two one-dimensional finite functions of two different variables (*e.g.*, x and z), and turns out to be completely defined by nine parameters.

Placing the finite functions $f_1(x)$ and $f_2(z)$ inside a rectangle P ($a_1 \le x \le b_1, a_2 \le z \le b_2$) enclosing the inhomogeneity, we obtain a function

$$\widetilde{\Delta\sigma}(x,z) = A f_1(x) f_2(z) \tag{1.6}$$

with a unique maximum amplitude A that represents the excess electrical conductivity contained within P. The magnitude of A depends on the conductance $\bar{S}$ of the inhomogeneity, and

$$\bar{S} = \int_P\int \widetilde{\Delta\sigma}(x,z)dx\,dz = A\int_P\int f_1(x)f_2(z)dx\,dz.$$

We might instead take $\bar{S}$ as a free parameter, solve for A and obtain

$$A = \frac{\bar{S}}{\int_P\int f_1(x) f_2(z) dx\, dz}.$$

Substituting this expression for A into (1.6), we obtain an expression for the finite function Ψ_P approximating $\widetilde{\Delta\sigma}(x, z)$:

$$\widetilde{\Delta\sigma}(x, z) \approx \Psi_P(\bar{S}, \alpha_1, \beta_1, \alpha_2, \beta_2, p_1, q_1, p_2, q_2) = \frac{\bar{S} f_1(x) f_2(z)}{\int_P\int f_1(x) f_2(z)} dx\, dz.$$

We now have a well-posed problem of minimizing the misfit with respect to a small number of parameters. Zhdanov and Golubev use the Nelder–Mead optimization method (see Section 3.7.2) to find the minimum.

We propose to solve the geoprospecting problem using an approximate quasisolution selection method. Our approach is described in the next chapter.

Chapter 2

Solution of Magnetotelluric Geoprospecting Problem

In the first chapter, we outlined the ill-posed problem that must be solved and some solution methods that have been used by other researchers. Traditionally, Tikhonov regularization has been the choice of practitioners, despite its great expense and the blurriness of results. Other researchers have investigated selection methods, in which the anomalous conductivity is described by a function of a certain form. Likewise, we search for an approximate quasisolution by following the same basic algorithm:

1. Formulate an initial set S in which $\sigma(x, y)$ is contained. (This set is meant to approximate the actual conductivity function.)

2. Find the best $\sigma(x, y)$ from the set S by minimizing the difference between the solution of the forward problem using σ and the true solution obtained from measurements.

2.1 Formulation of Problem in Two Dimensions

The form of the conductivity function depends on the set S to which $\sigma(x, y)$ is constrained. We constrain the function $\sigma(x, y)$ to the set of piecewise constant functions on two regions having two different values,

$$\sigma(x,y) = \begin{cases} \sigma_{\text{in}} & \text{if } (x,y) \in \Omega_{\text{in}} \\ \sigma_{\text{out}} & \text{if } (x,y) \notin \Omega_{\text{in}}. \end{cases}$$

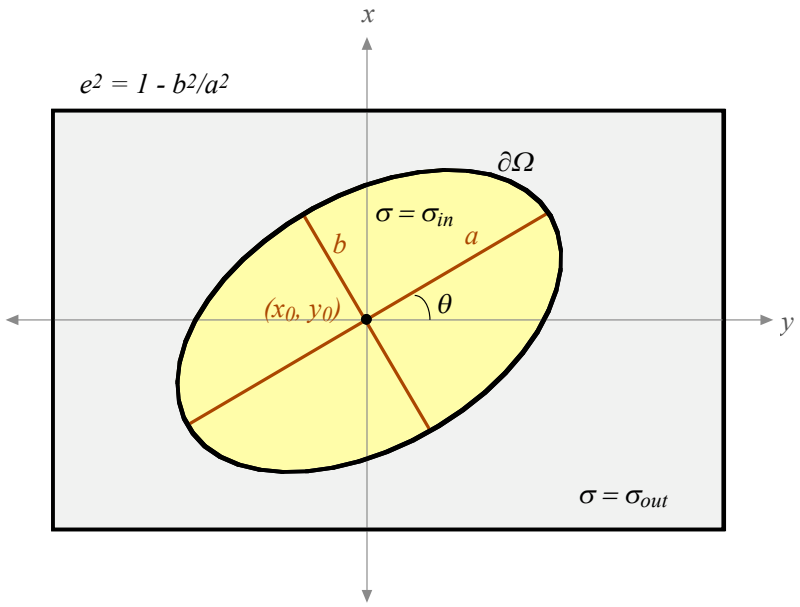


Figure 2.1. Example conductivity function $\sigma(x, y)$.

Furthermore, the interface $\partial\Omega_{\text{in}}$ is taken to be an ellipse parameterized as follows:

$$\begin{aligned} r(x, y, a, e^2, x_0, y_0, \theta) &= (1 - e^2\cos^2\theta)(x - x_0)^2 - 2e^2\sin\theta\cos\theta(x - x_0)(y - y_0) \\ &\quad +(1 - e^2\sin^2\theta) = a^2(1 - e^2) \end{aligned}$$

(see Figure 2.1 for an illustration). This formulation makes sense for the geoprospecting application in particular. An oil deposit has a constant conductivity, and the surrounding rock has a different constant conductivity. The interface can be described as above. We chose the elliptical formulation because relatively few parameters are required to describe an ellipse, but it is still general enough to describe the approximate size, shape, and orientation of an arbitrary region.

We expand $\phi(x, y) : [-\pi, \pi] \times [-\pi, \pi] \mapsto \mathbb{R}$ in a truncated Fourier sine series in two dimensions:

$$\phi(x, y) = \sum_{m=1}^{M}\sum_{n=1}^{N} \alpha_{mn}\sin(mx)\sin(ny).$$

The Fourier formulation of ϕ is quite convenient, as every component is orthogonal to the other components, *i.e.*,

$$\int_{-\pi}^{\pi} \sin(mx)\sin(nx)\,dx = 0 \quad \text{if} \quad m \neq n.$$

2.1.1 Solution of Forward Problem in Two Dimensions

In order to minimize the error, we set up the problem so that the error is orthogonal to the Fourier basis. Thus we select coefficients α_{mn} such that

$$\int_{\Omega} g_{pq}(x,y)\,(\nabla \cdot J)\,\mathrm{d}\Omega - \int_{\Omega} g_{pq}(x,y)\,\nabla \cdot \sigma \nabla(\alpha_{mn}\sin(mx)\sin(ny))\,\mathrm{d}\Omega = 0, \tag{2.1}$$

where $g_{pq}(x,y) = \sin(px)\sin(qy)$. We evaluate the second term of (2.1) for every combination of ϕ_{mn} and g_{pq} to obtain the entries of a matrix A, and we compute the first term of (2.1) to obtain the right-hand side b. Thus we obtain a square system of linear equations

$$A\alpha = b$$

of dimension MN, which we solve for α. We can then calculate an approximation to ϕ based on our approximate σ,

$$\phi(x,y) = \sum_{m,n} \alpha_{mn}\phi_{mn}(x,y).$$

2.2 Formulation of Problem in Three Dimensions

As in the 2-D problem, we constrain the function $\sigma(x,y,z)$ to the set of piecewise constant functions on two regions having two different values,

$$\sigma(x,y,z) = \begin{cases} \sigma_{\text{in}} & \text{if } (x,y,z) \in \Omega_{\text{in}} \\ \sigma_{\text{out}} & \text{if } (x,y,z) \notin \Omega_{\text{in}}. \end{cases}$$

Furthermore, the interface $\partial\Omega_{\text{in}}$ is taken to be an ellipsoid parameterized as follows:

$$\begin{aligned} r(x,y,z,a,e_1^2,e_2^2,x_0,y_0,z_0,\theta_1,\theta_2,\theta_3) &= x^2[c_{11}^2(1-e_1^2)(1-e_2^2)+c_{21}^2(1-e_2)^2+c_{31}^2(1-e_1^2)] \\ &\quad +2xy[c_{11}c_{12}(1-e_1^2)(1-e_2^2)+c_{21}c_{22}(1-e_2)^2+c_{31}c_{32}(1-e_1^2)] \\ &\quad +2xz[c_{11}c_{13}(1-e_1^2)(1-e_2^2)+c_{21}c_{23}(1-e_2)^2+c_{31}c_{33}(1-e_1^2)] \\ &\quad +y^2[c_{12}^2(1-e_1^2)(1-e_2^2)+c_{22}^2(1-e_2)^2+c_{32}^2(1-e_1^2)] \\ &\quad +2yz[c_{12}c_{13}(1-e_1^2)(1-e_2^2)+c_{22}c_{23}(1-e_2)^2+c_{32}c_{33}(1-e_1^2)] \\ &\quad +z^2[c_{13}^2(1-e_1^2)(1-e_2^2)+c_{23}^2(1-e_2)^2+c_{33}^2(1-e_1^2)] \\ &= a^2(1-e_1^2)(1-e_2^2), \end{aligned}$$

where the coefficients

$$\begin{aligned} c_{11} &= \cos\theta_3\cos\theta_2-\cos\theta_2\sin\theta_1\sin\theta_3, \\ c_{12} &= \cos\theta_3\sin\theta_2+\cos\theta_2\cos\theta_1\sin\theta_3, \\ c_{13} &= \sin\theta_3\sin\theta_2, \\ c_{21} &= -\sin\theta_3\cos\theta_2-\cos\theta_2\sin\theta_1\cos\theta_3, \\ c_{12} &= -\sin\theta_3\sin\theta_2+\cos\theta_2\cos\theta_1\cos\theta_3, \\ c_{23} &= \cos\theta_3\sin\theta_2, \\ c_{31} &= \sin\theta_2\sin\theta_1, \\ c_{32} &= -\sin\theta_2\cos\theta_1, \\ c_{33} &= \cos\theta_2, \end{aligned}$$

and e_1^2 and e_2^2 represent the eccentricity between the x and y and the x and z axes, respectively; (x_0,y_0,z_0) is the center point of the ellipsoid; θ_1 is an angle about the z axis; $\theta_2\in[0,\pi]$ is an angle about the x axis; and θ_2 is another angle about the z axis. See Figure 2.2 for an illustration.

Like the two-dimensional case, we expand $\phi(x,y,z):[-\pi,\pi]^3\mapsto\mathbb{R}$ in a truncated Fourier sine

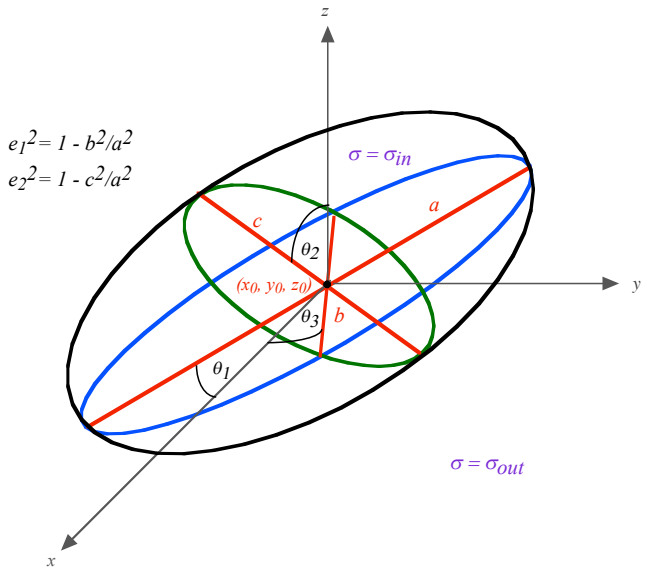


Figure 2.2. Example conductivity function $\sigma(x, y, z)$.

series in three dimensions:

$$\phi(x, y, z) = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{p=1}^{P} \alpha_{mnp} \sin(mx) \sin(ny) \sin(pz).$$

2.2.1 Solution of Forward Problem in Three Dimensions

Like the 2-D case, the error is constructed to be orthogonal to the Fourier basis. Thus we select coefficients α_{mnp} such that

$$\int_\Omega g_{qrs}(x, y, z)(\nabla \cdot J)\mathrm{d}\Omega - \int_\Omega g_{qrs}(x, y, z)\nabla \cdot \sigma \nabla(\alpha_{mnp} \sin(mx) \sin(ny) \sin(pz))\mathrm{d}\Omega = 0, \qquad (2.2)$$

where $g_{qrs}(x, y, z) = \sin(qx) \sin(ry) \sin(sz)$. We evaluate the second term of (2.2) for every combination of ϕ_{mnp} and g_{qrs} to obtain the entries of a matrix A, and we compute the first term of (2.2) to obtain the right-hand side vector b. Thus we obtain a linear system

$$A\alpha = b$$

of dimension MNP, which we solve for α. We can then calcuate an approximation to ϕ based on our approximate σ,

$$\phi(x, y, z) = \sum_{m,n,p} \alpha_{mnp} \phi_{mnp}(x, y, z).$$

2.3 Solution of Inverse Problem

We determine the optimality of a solution by taking the norm of the difference between the computed coefficients and the true coefficients. So to solve the inverse problem, we perform the following optimization:

$$\min_{\alpha} \Psi = \|\alpha - \alpha_{\text{true}}\|_2^2. \tag{2.3}$$

We know the true coefficients because they represent the amplitude of the input electromagnetic field. We choose to perform the optimization with respect to α because it is an easily calculated vector with a simple physical meaning.

We use the Euclidean norm because of its simplicity. This function is not easily differentiated in closed form, in general, and because of the expense of a single function evaluation, finite differencing is costly.

2.4 Optimization

Preliminary exploration of the objective function Ψ in two spatial dimensions was performed. In this initial study, x_0, y_0, and θ were constrained to be zero, while a and e^2 were allowed to vary, yielding a two-dimensional search space. For a given right-hand side, a unique global minimum exists (see example in Figure 2.3).

We tried many local optimization methods, including Model-Assisted Pattern Search (MAPS) [30], LMDIF [5], Praxis [1], Nelder–Mead [21], Sufficient Decrease Nelder–Mead [12], and BFGS [26]. Most of these methods successfully converged to the global minimum if the initial point was located "close enough" to the global minimum. We discovered, however, that there are many local minima to which these optimization methods will converge if the starting point is located "close enough" to them. These local minima are more shallow than the global minimum but still attractive to any

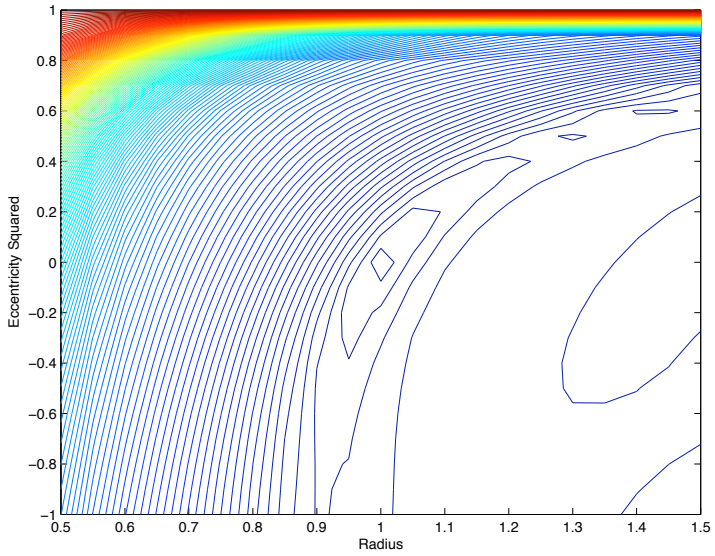


Figure 2.3. Example objective function Ψ, with $a_{\text{true}} = 1, e^2_{\text{true}} = 0$.

starting points outside the neighborhood of the global minimum. Since they are located all around the global minimum, a starting point outside the area would naturally be attracted to one of these local minima. Thus it was necessary to find a global optimization method.

2.4.1 Global Optimization Methods

A standard local optimization problem can be defined as follows. Given a nonempty, closed set $D \subset \mathbb{R}^n$ and a continuous function $f : A \mapsto \mathbb{R}$, where $A \subset \mathbb{R}^n$ is a suitable set containing D, find a point $x^* \in D$ satisfying $f(x^*) \leq f(x)$ for all $x \in D$ in a neighborhood of x^*, or show that such a point does not exist.

A standard *global* optimization problem is more onerous: Given a nonempty, closed set $D \subset \mathbb{R}^n$ and a continuous function $f : A \mapsto \mathbb{R}$, where $A \subset \mathbb{R}^n$ is a suitable set containing D, find at least one point $x^* \in D$ satisfying $f(x^*) \leq f(x)$ for all $x \in D$, or show that such a point does not exist [9].

Let us denote the minimum value of the function $f(x)$ as f^* and the point at which the minimum is located as x^*. If D is compact and $f(x)$ is continuous, then the problem of finding f^* is well-posed. We cannot guarantee that f^* will be found in a finite number of steps, however. To see this,

assume that $f(x)$ has been evaluated at a finite number of points. It is possible that the global minimum of the function has not yet been found. Indeed, we could construct a function fitting those points but for which the global minimum is smaller than any function value found so far. There is no algorithm that will find f^* for every $f(x)$ in a finite number of steps [35].

To make matters worse, the problem of finding x^* is ill-posed. Even if f is continuous and satisfies very stringent smoothness conditions, x^* is not a continuous function of f. For example, consider the functions f_δ, where

$$f_\delta(x) = \cos(\pi x) - \delta x, \qquad x \in [-2, 2].$$

If $\delta > 0$, then $x^* \approx 1$, if $\delta < 0, x^* \approx -1$, and if $\delta = 0$ then the solution is not unique [35]. A very small change in f can result in a very large change in x^* [1].

There are many methods that guarantee that a local minimum will be found, but no corresponding methods for finding global minima. How do we go about finding minima? Even if we find a minimum, how can we be assured that it is the global minimum? While there is no guarantee that any strategy we adopt (short of examining every point in the set D) will find a global minimum, researchers have developed many techniques for global optimization that often work well for certain classes of functions. Most global optimization methods for continuous functions fall into one of three classes: branch and bound methods, stochastic methods, and smoothing methods.

Branch and Bound Methods

The general idea of a branch and bound method is to subdivide the feasible region of a problem into smaller subproblems [6]. In order for this method to work, it must be possible to compute lower and upper bounds on the objective function.

We begin by considering the original problem with its original feasible region (known as the *root problem*). We compute the lower and upper bounds, and if they match, we have found an optimal solution and the procedure is terminated. Otherwise, the feasible region is subdivided into strict subregions (or *child problems*) that together make up the original feasible region. The algorithm is applied recursively to the subproblems, forming a tree of subproblems. While an optimal solution

to a subproblem is a feasible solution to the root problem, it is not necessarily a global optimum. We can use it nonetheless to prune the rest of the tree: If the lower bound of a node is larger than the best feasible solution found so far, then we can eliminate the subspace of that node from the search. The search terminates when all nodes have been solved or pruned, or until some threshold between the best solution and the lower bounds on all unsolved problems is met.

In the worst case, this sort of method requires an exponential amount of work, but in practice most problems do not exhibit this worst case behavior. Neumaier suggests that branch and bound methods, combined with computationally verifiable sufficient conditions for global minima, may permit proof of global optimality of a solution [22].

Stochastic Methods

The two most popular stochastic methods are simulated annealing and evolutionary algorithms. Both types of methods can be proven convergent in a probabilistic sense, and rely on properties of statistics to overcome local barriers that would otherwise force them into local minima. These methods are suitable in particular for objective functions that are cheap to evaluate.

Simulated annealing is based on the idea that heating and then slowly cooling a metal brings it into a relatively uniform crystalline state, in other words, a configuration in which the free energy of the metal is minimized. A temperature parameter allows the configuration to reach higher energy states, so that states beyond a local well can be visited and examined. Simulated annealing is provably convergent in a probabilistic sense using a sufficiently slow annealing schedule. Researchers have developed enhancements that speed the convergence of simulated annealing, but success depends on the implementation used [22].

Evolutionary algorithms take their inspiration from biological evolution. By making small alterations to promising solutions, evolutionary algorithms seek to derive even better solutions at each step. Evolutionary algorithms begin with a "population" of solutions that compete against one another to eliminate poor solutions. The remaining solutions are mutated (each coordinate of a promising solution is allowed to change with a certain probability) or crossed (the coordinates of two promising solutions are interchanged), resulting in new possible solutions biased toward regions

of space in which good solutions have already been found [6].

The efficiency of evolutionary algorithms depends greatly on the proper selection of crossing rules. If the influence of the coordinates is nearly independent, then crossing may produce beneficial results. If, on the other hand, the influence of the coordinates is highly correlated, as in the case of functions with deep and narrow valleys not parallel to the coordinate axes, evolutionary algorithms may not perform as efficiently [22]. Like simulated annealing, success depends on the implementation. In particular, tuning the parameters of the algorithm requires considerable insight into the nature of the particular objective function for which the minimum is sought.

Smoothing Methods

Smoothing methods are based on the idea that macroscopic features are an average of microscopic details. For example, a valley seen from an airplane appears to have a simple shape; more and more local features become visible upon closer examination at smaller and smaller scales [22]. Thus we could use some sort of smoothing process to simplify the appearance of our function until enough details are removed that we can locate a single global minimum. Then we undo the smoothing progressively, adding back detail and finding the minimum of the new function at each step, until all the smoothing is undone and the global minimizer is found. While these methods are not guaranteed to find the global minimum of all objective functions, they require relatively few function evaluations, compared with stochastic methods in particular.

Choosing a Suitable Global Optimization Method

Each class of global optimization methods has its strengths and weaknesses, and is most suitable for a certain type of objective function. Branch and bound methods work well for objective functions whose upper and lower bounds are easily computed, and for discrete objective functions. Stochastic methods befit objective functions that are inexpensive to evaluate, because hundreds of thousands of evaluations are required for even the simplest objective function. Smoothing methods are expedient for relatively smooth objective functions with global minima located in large wells.

There are no simple upper and lower bounds for the magnetotelluric objective function that

could be used for branch and bound. In addition, the magnetotelluric objective function is costly to evaluate, so stochastic methods are prohibitively expensive. While the objective function is steep in some places, it is relatively smooth, so smoothing methods seem the most suitable.

There are many smoothing methods, including neighborhood averaging methods and integral smoothing methods. While all smoothing methods are attractive for the magnetotelluric global optimization problem, we chose the Diffusion Equation Method, a smoothing and continuation method, for its elegance and simplicity.

Chapter 3

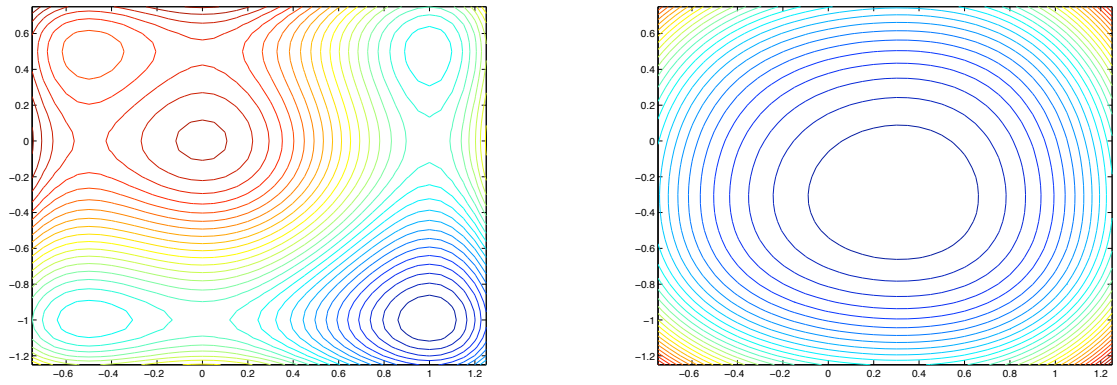
Diffusion Equation Method for Global Optimization

3.1 Background

The Diffusion Equation Method (DEM) is a global optimization method based on the concepts of smoothing and continuation. The main idea of smoothing and continuation is to transform the objective function into a sequence of progressively smoother functions, with the degree of smoothing controlled by a smoothing parameter λ. The sequence ends at a $\lambda = \lambda_{\max}$ for which only one minimum remains. Starting with this smoothest function, the minimum is found. The degree of smoothing is then reduced and the minima of the smoothed functions traced back to the global minimum of the original objective function. At each step of this *continuation*, a local optimization algorithm finds the minimum of the smoothed function, using the minimum from the previous step as starting point [19].

Ideally, if the objective function can be smoothed enough, then other local minima will be suppressed and only one minimum will remain. If the minimum of the smoothed function is found and the continuation is performed with sufficiently small steps, then by the step at which the local minima reappear, the starting point obtained from the previous step should be close enough to the global minimum that the global minimum can be found.

One question is what sort of smoothing transform should be used. Since integration is an inherently smoothing process, it is natural to use an integral transform, in which the objective



(a) (b)

Figure 3.1. Example of smoothing: $f(x,y) = (\frac{1}{2}x^4 - \frac{1}{3}x^3 - \frac{1}{2}x^2 + 2)(\frac{1}{2}y^4 + \frac{1}{3}y^3 - \frac{1}{2}y^2 + 2)$: (a) original function and (b) function smoothed using diffusion transform at $t = 0.2$.

function f is integrated against a kernel function ρ:

$$F(x,\lambda) = \frac{1}{\lambda^n} \int_{\mathbb{R}^n} f(y)\rho\left(\frac{x-y}{\lambda}\right) dy.$$

Many different classes of functions could be used as the kernel function, including the uniform density function $\rho(y) = \frac{1}{2}, \|y\|_\infty \leq 1$ and the Gaussian density function $\rho(y) = \exp(-\|y\|_2^2)$. Moré and Wu have delineated the properties of integral transforms for this purpose in [19].

The Diffusion Equation Method, proposed by Piela, Kostrowicki, and Scheraga for use in conformational analysis of molecules [28], is an integral transform method. The idea of the DEM is to transform the objective function $f(x) : \mathbb{R}^n \mapsto \mathbb{R}$ into a function $F(x,t) : \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}$ by using the diffusion operator. The diffused function is a solution to the following initial value problem:

$$\begin{aligned} \frac{\partial F(x,t)}{\partial t} &= \sum_{i=1}^{n} \frac{\partial^2 F(x,t)}{\partial x_i^2}, \\ F(x,0) &= f(x). \end{aligned}$$

The initial value problem can be solved using the Fourier transform, and we find that

$$F(x,t) = \frac{1}{(4\pi t)^{n/2}} \int_{\mathbb{R}^n} f(y) \exp\left(-\frac{\|x-y\|^2}{4t}\right) dy, \tag{3.1}$$

which resembles the convolution of f with the Gaussian density function. Thus the properties of Gaussian smoothing apply [19], but the advantage of this transformation is that it is more amenable to numerical implementation: Numerical quadrature or finite differences could be used. In one spatial dimension, for example, the value of the smoothed function at time step $n+1$ is approximated with finite differences by

$$F_j^{n+1} = F_j^n + \frac{\Delta t}{(\Delta x)^2}(F_{j+1}^n - 2F_j^n + F_{j-1}^n).$$

Another way to express the analytic diffusion transform is with the Taylor-series diffusion operator [28]

$$T(t) = \exp\left(t\frac{d^2}{dx^2}\right) := 1 + t\frac{d^2}{dx^2} + \frac{t^2}{2}\frac{d^4}{dx^4} + \frac{t^3}{3!}\frac{d^6}{dx^6} + \cdots .$$

In multiple dimensions, the diffusion operator is the product of one-dimensional operators:

$$T(t) = T_1(t)T_2(t)\ldots T_n(t),$$

where

$$T_i(t) = \exp\left(t\frac{\partial^2}{\partial x_i^2}\right).$$

This formulation has proved useful for testing the DEM on analytical test functions. Applying the differentiation operator is much simpler than kernel integration, particularly for polynomials.

3.2 Properties of Diffusion

We examine the effect of diffusion on a one-dimensional function. First we outline some properties of the diffusion operator. Then we study the effect of diffusion upon two of the most important basis sets for approximation of all functions: sinusoids and polynomials. Using this knowledge, we examine the effect of diffusion on coercive functions. We restrict our study to functions of one variable, but these results could be extended to functions in multiple dimensions, which would behave similarly to the one-dimensional case.

The diffusion operator is a linear operator, because $T(t)(f(x) + g(x)) = T(t)f(x) + T(t)g(x)$

and $T(t)(\alpha f(x)) = \alpha T(t) f(x)$.

Diffusion damps high-frequency noise. If we apply the diffusion operator to a sinusoid, we can see how oscillations are damped. Let $\alpha = 2\pi k/X$, where k is the mode number, and X is the period. Then

$$\begin{aligned} T(t)\sin(\alpha x) &= \left(1 + t\frac{d^2}{dx^2} + \frac{t^2}{2}\frac{d^4}{dx^4} + \frac{t^3}{3!}\frac{d^6}{dx^6} + \dots\right)\sin(\alpha x) \\ &= \sin(\alpha x) - \alpha^2 t \sin(\alpha x) + \alpha^4 \frac{t^2}{2}\sin(\alpha x) + \alpha^6 \frac{t^3}{3!}\sin(\alpha x) + \dots \\ &= \sin(\alpha x)\left(1 - \alpha^2 t + \frac{\alpha^4 t^4}{2} - \frac{\alpha^6 t^6}{3!} + \dots\right) \\ &= \sin(\alpha x)\exp(-\alpha^2 t) \\ &= \sin\left(\frac{2\pi k x}{X}\right)\exp\left(\frac{-4\pi^2 k^2 t}{X^2}\right). \end{aligned}$$

The same procedure can be used for the cosine function and an identical damping factor is obtained. Notice that the sine function obtains an exponential damping factor from diffusion, and that the damping grows exponentially with k^2. Thus higher frequency components damp more quickly, while lower frequency components damp at a much slower rate.

3.2.1 Effect of Diffusion on Periodic Functions

A periodic function $f(x)$ can be approximated by a Fourier series

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}\left(a_k \cos\left(\frac{2\pi k x}{X}\right) + b_k \sin\left(\frac{2\pi k x}{X}\right)\right),$$

where X is the length of the period, and a_k and b_k are the amplitudes of each component, derived by integrating f against the appropriate sinusoid.

If $|f(x)|^2$ is integrable on $[x_0, x_0 + X]$, then the Fourier series converges to f in the mean-square sense on the interval. Also, the Fourier coefficents will converge. This is shown by the following proofs, adapted from [37].

Theorem 3.1. *Let $f(x)$ be a real function of period X on $[x_0, x_1]$ (where $x_1 = x_0 + X$), and let $\{\phi_k\}$ be an orthonormal set of functions on $[x_0, x_1]$. If $|f(x)|^2$ is integrable on $[x_0, x_1]$, then the*

Fourier series converges to f in the mean-square sense on the interval.

Proof: For a fixed integer $n \geq 0$, letting

$$\Phi = \gamma_0\phi_0 + \gamma_1\phi_1 + \cdots + \gamma_n\phi_n,$$

we seek values of the constants $\gamma_0, \gamma_1, \ldots, \gamma_n$ that minimize the integral

$$J = \int_{x_0}^{x_1} |f - \Phi|^2 \, dx.$$

Observe that

$$\begin{aligned}
\int_{x_0}^{x_1} |\Phi|^2 dx &= \int_{x_0}^{x_1} \left(\sum_{j=0}^{n} \gamma_j \phi_j \right) \left(\sum_{k=0}^{n} \gamma_k \phi_k \right) dx = \sum_{k=0}^{n} |\gamma_k|^2, \\
\int_{x_0}^{x_1} f\Phi \, dx &= \int_{x_0}^{x_1} f \left(\sum_{k=0}^{n} \gamma_k \phi_k \right) dx = \sum_{k=0}^{n} c_k \gamma_k,
\end{aligned}$$

where $c_0, c_1, \ldots, c_n$ are the Fourier coefficients of f with respect to $\{\phi_k\}$.

From these observations we can see that

$$\begin{aligned}
J = \int_{x_0}^{x_1} |f - \Phi|^2 dx &= \int_{x_0}^{x_1} |f|^2 dx + \int_{x_0}^{x_1} |\Phi|^2 dx - 2\int_{x_0}^{x_1} f\Phi \, dx \\
&= \int_{x_0}^{x_1} |f|^2 dx + \sum_{k=0}^{n} |\gamma_k|^2 - 2\sum_{k=0}^{n} c_k \gamma_k.
\end{aligned}$$

Adding and subtracting $\sum_{k=0}^{n} |c_k|^2$, we obtain

$$J = \int_{x_0}^{x_1} |f|^2 dx - \sum_{k=0}^{n} |c_k|^2 + \sum_{k=0}^{n} |c_k - \gamma_k|^2.$$

From this equation it follows that J is minimized if $\gamma_k = c_k$ for $k = 0, 1, \ldots, n$.

Thus, if $|f(x)|^2$ is integrable on $[x_0, x_1]$ and if $\Phi = \gamma_0\phi_0 + \gamma_1\phi_1 + \cdots + \gamma_n\phi_n$, where $\phi_0, \phi_1, \ldots,$ form an orthonormal system on $[x_0, x_1]$, then the integral $J = \int_{x_0}^{x_1} |f - \Phi|^2 dx$ is a minimum when Φ is the nth partial sum of the Fourier series of f with respect to $\{\phi_k\}$. □

Theorem 3.2. *If $f(x)$ is a function of period X and $|f(x)|^2$ is integrable on $[x_0, x_1]$, then the Fourier coefficients $\{c_k\}$ converge to zero as $k \to \infty$.*

Proof: Recall that in Theorem 3.1 we have determined the minimum value for the integral J, representing the minimum of the difference between f and its Fourier series representation Φ. This minimum is

$$\int_{x_0}^{x_1} |f|^2 dx - \sum_{k=0}^{n} |c_k|^2 > 0.$$

Rearranging, we obtain Bessel's inequality, which is an upper bound on the sum of the coefficients:

$$\sum_{k=0}^{n} |c_k|^2 \leq \int_{x_0}^{x_1} |f|^2 dx.$$

Taking the limit as $n \to \infty$, we obtain

$$\sum_{k=0}^{\infty} |c_k|^2 \leq \int_{x_0}^{x_1} |f|^2 dx,$$

where the c_k are obtained by integrating f against $\exp((ikx)/(2\pi))$ on $[x_0, x_1]$.

In the case of a real-valued f, an equivalent expression of Bessel's inequality is

$$\frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k^2 + b_k^2) \leq \int_{x_0}^{x_1} f^2 \, dx.$$

From this it follows that the Fourier coefficients tend to 0 with $1/k$, provided that $|f|^2$ is integrable. □

Furthermore, the Riemann–Lebesgue Theorem states that the Fourier coefficients c_k of an integrable function f tend to 0 as $|k| \to \infty$. The same result applies in the real case, since $c_k = (a_k - ib_k)/2$ for $k > 0$ [37].

We have shown that the Fourier series approximation converges to an integrable function, but this says nothing about the accuracy of the Fourier series representation of diffusion. Using the Taylor series representation of the diffusion operator, we can see that the accuracy of the diffusion depends on the even derivatives of f. So if the derivatives of Φ match the derivatives of f, then the

effect of diffusion should be identical.

As it turns out, if $S[f]$ is the Fourier series expansion of f, then $S^{(k)}[f] = S[f^{(k)}]$ as long as f is a kth integral (meaning that there exists a function g such that integrating it k times yields f) [37]. Thus for a periodic function $f \in C^\infty$ the Fourier series representation exactly represents the effect of diffusion.

From the Fourier series representation, we see that the high frequency components of a periodic function are damped faster than the low frequency components, and that the zeroth component, a_0, is not damped at all. Thus as $t \to \infty$, $F(x,t) \to a_0$. This matches the physical behavior of diffusion, which ultimately leads to a constant distribution over the entire region.

3.2.2 Effect of Diffusion on Coercive Polynomials

Next we discuss the effects of diffusion on a coercive polynomial. A coercive function tends toward infinity with the absolute value of x. More precisely, "[a] continuous function f on an unbounded set $S \subseteq \mathbb{R}^n$ is said to be *coercive* if

$$\lim_{\|x\| \to \infty} f(x) = +\infty,$$

i.e., for any constant M, there is an $r > 0$ (depending on M) such that $f(x) \geq M$ for any $x \in S$ such that $\|x\| \geq r$." The implication of coercivity is that f must have a global minimum on S [8].

Theorem 3.3. *A coercive polynomial $p(x) = c_0 + c_1 x + \cdots + c_K x^K$ must be of even degree, and the coefficient of the highest degree term must be positive.*

Proof: Suppose that the highest degree is odd. Then the highest degree term could be represented as $c_{2n+1}x^{2n+1}$ for some $n \geq 0$. For x sufficiently large in absolute value, this term will dominate the lower degree terms. But as $x \to +\infty$, $x^{2n+1} \to +\infty$, while as $x \to -\infty$, $x^{2n+1} \to -\infty$, meaning that at one end of the real line $p(x) \to -\infty$, so $p(x)$ cannot be coercive. A similar argument applies for the coefficient of the highest degree term. Suppose that the highest degree term in the polynomial, the $(2n)$th term, has a negative coefficient. For sufficiently large absolute values of x, this term will dominate the lower degree terms. As $x \to \pm\infty$, $c_{2n}x^{2n} \to -\infty$. Thus at its extreme, $p(x) \to -\infty$, so $p(x)$ cannot be coercive. $\square$

The diffusion transform of a polynomial p is simply the weighted sum of its even derivatives multiplied by appropriate powers of t,

$$P(x,t) = T(t)p(x) = p(x) + t\frac{d^2p}{dx^2} + \frac{t^2}{2}\frac{d^4p}{dx^4} + \frac{t^3}{3!}\frac{d^6p}{dx^6} + \dots.$$

For a polynomial of degree K, there are $1 + \lfloor K/2 \rfloor$ terms in this expansion, and the highest power of t occurring in the expansion is $t^{\lfloor K/2 \rfloor}$. If a coercive polynomial $p(x)$ is of degree $2N$, then its diffusion transform is

$$P(x,t) = p(x) + t\frac{d^2p}{dx^2} + \frac{t^2}{2}\frac{d^4p}{dx^4} + \frac{t^3}{3!}\frac{d^6p}{dx^6} + \dots + \frac{t^{N-1}}{(N-1)!}\frac{d^{2N-2}p}{dx^{2N-2}} + \frac{t^N}{N!}\frac{d^{2N}p}{dx^{2N}}.$$

Theorem 3.4. *Let $p(x)$ be a coercive polynomial of degree $2N$ on $\mathbb{R}$, and let $P(x,t)$ be its diffusion transform. Then for any constant M and $r > 0$ (depending on M) such that $p(x) \geq M$ whenever $\|x\| \geq r$, there exists a finite $\tau > 0$ such that $P(x,t)$ is convex for $\|x\| < r$ and $t \geq \tau$.*

Proof: The second derivative of $P(x,t)$ with respect to x determines its convexity: If the second derivative is always positive, then the function is convex. Taking the second derivative of the diffusion transform with respect to x, we obtain

$$\frac{\partial^2 P}{\partial x^2} = \frac{d^2p}{dx^2} + t\frac{d^4p}{dx^4} + \frac{t^2}{2}\frac{d^6p}{dx^6} + \dots + \frac{t^{N-1}}{(N-1)!}\frac{d^{2N}p}{dx^{2N}}.$$

(Note that the last term of P drops out because higher order derivatives are zero.) The final term in the second derivative is

$$\frac{t^{N-1}}{(N-1)!}\frac{d^{2N}p}{dx^{2N}} = \frac{t^{N-1}}{(N-1)!}(2N)!\,c_{2N},$$

where $c_{2N} > 0$ is the coefficient of the x^{2N} term of p.

Because $\|x\| < r$ and the coefficients c_i are finite, each term in the second derivative is bounded for finite t. As t grows, the final term in the second derivative eventually dominates. It grows with increasing t at a faster rate than its nearest competitor. Thus there exists $\tau > 0$ such that $\frac{\partial^2 P(x,t)}{\partial x^2} > 0$ on $\{x : \|x\| < r\}$. And as t becomes larger than τ, the second derivative grows, so $\frac{\partial^2 P(x,t)}{\partial x^2} > 0$ on $\{x : \|x\| < r\}$ for $t \geq \tau$. □

3.2.3 Effect of Diffusion on Arbitrary Coercive Functions

According to the Weierstrass Approximation Theorem, an arbitrary continuous function can be approximated arbitrarily well on a closed and bounded interval by a polynomial. The following proof of the Weierstrass Approximation Theorem is adapted from [3].

Theorem 3.5. Weierstrass Approximation Theorem. *Let $I \subset \mathbb{R}$ be a closed and bounded interval and f a continuous function defined on I. Then, for every $\epsilon > 0$ there exists a polynomial p such that*

$$|f(x) - p(x)| \leq \epsilon \text{ for all } x \in I.$$

Proof: Without loss of generality, we can take $I = [0, 1]$, since any desired interval $[a, b]$ can be scaled to $[0, 1]$ by the transformation $x = (x' - a)/(b - a)$. For $N \in \mathbb{N}$, define the Bernstein polynomial B_N on $[0, 1]$,

$$B_N(x) = \sum_{k=0}^{N} f(k/N) \binom{N}{k} x^k (1-x)^{N-k}. \tag{3.2}$$

To bound $|f(x) - B_N(x)|$, we rewrite $f(x)$ with the help of the binomial formula. Multiplying f by

$$1 = (x + (1-x))^N = \sum_{k=0}^{N} \binom{N}{k} x^k (1-x)^{N-k}$$

does not change its value. Thus we have

$$\begin{aligned} f(x) - B_N(x) &= \sum_{k=0}^{N} f(x) \binom{N}{k} x^k (1-x)^{N-k} - \sum_{k=0}^{N} f(k/N) \binom{N}{k} x^k (1-x)^{N-k} \\ &= \sum_{k=0}^{N} (f(x) - f(k/N)) \binom{N}{k} x^k (1-x)^{N-k}. \end{aligned}$$

Recall that because $f(x)$ is uniformly continuous on I, for a given $\epsilon > 0$ there is a $\delta > 0$ such that

$$x, y \in [0, 1], |x - y| < \delta \Rightarrow |f(x) - f(y)| < \epsilon/2.$$

For $x \in [0,1]$, we divide the set of indices $\{k\}_0^N$ into two parts: those for which $|x - k/N| < \delta$ and those for which $|x - k/N| \geq \delta$, to define the partial sums

$$S_1 = \sum_{|x-k/N|<\delta} (f(x) - f(k/N)) \binom{N}{k} x^n (1-x)^{N-k}, \tag{3.3}$$

$$S_2 = \sum_{|x-k/N|\geq\delta} (f(x) - f(k/N)) \binom{N}{k} x^n (1-x)^{N-k} \tag{3.4}$$

so that

$$f(x) - B_N(x) = S_1 + S_2.$$

We start with a bound on $|S_1|$. By the triangle inequality, we see that

$$|S_1| \leq \sum_{|x-k/N|<\delta} |f(x) - f(k/N)| \binom{N}{k} x^k (1-x)^{N-k}.$$

It follows from our choice of δ that

$$\begin{aligned} |S_1| &\leq \frac{\epsilon}{2} \sum_{|x-k/N|<\delta} \binom{N}{k} x^k (1-x)^{N-k} \\ &\leq \frac{\epsilon}{2} \sum_{k=0}^{N} \binom{N}{k} x^k (1-x)^{N-k} \\ &= \frac{\epsilon}{2} (x + (1-x))^N \\ &= \frac{\epsilon}{2}. \end{aligned}$$

Next we bound $|S_2|$. First, we can rewrite $|x - k/N| \geq \delta$ as

$$\frac{(x-k/N)^2}{\delta^2} \geq 1.$$

Now we see that

$$|S_2| \leq \sum_{\frac{(x-k/N)^2}{\delta^2}\geq 1} |f(x) - f(k/N)| \binom{N}{k} x^k (1-x)^{N-k}.$$

Let $M = \max_{x\in[0,1]} |f(x)|$. By the triangle inequality, we have

$$|f(x) - f(k/N)| \leq |f(x)| + |f(k/N)| \leq 2M.$$

Using this inequality, it follows that

$$\begin{aligned} |S_2| &\leq 2M \sum_{\frac{(x-k/N)^2}{\delta^2}\geq 1} \binom{N}{k} x^k (1-x)^{N-k} && (3.5) \\ &\leq 2M \sum_{\frac{(x-k/N)^2}{\delta^2}\geq 1} \frac{(x-k/N)^2}{\delta^2} \binom{N}{k} x^k (1-x)^{N-k}. && (3.6) \end{aligned}$$

((3.6) is derived from the fact that $\frac{(x-k/N)^2}{\delta^2} \geq 1$.) Including the additional terms from (3.3) will only increase the sum; thus

$$\begin{aligned} |S_2| &\leq 2M \sum_{k=0}^{N} \frac{(x-k/N)^2}{\delta^2} \binom{N}{k} x^k (1-x)^{N-k} && (3.7) \\ &\leq \frac{2M}{\delta^2} \sum_{k=0}^{N} (x-k/N)^2 \binom{N}{k} x^k (1-x)^{N-k}. && (3.8) \end{aligned}$$

The sum

$$\sum_{k=0}^{N} (k-Nx)^2 \binom{N}{k} x^k (1-x)^{N-k}$$

is the variance for a binomial distribution with parameter N and probability parameter x, so it sums to $Nx(1-x)$. Thus

$$\sum_{k=0}^{N} \left(x - \frac{k}{N}\right)^2 \binom{N}{k} x^k (1-x)^{N-k} = \frac{1}{N} x(1-x)$$

and

$$\frac{2M}{\delta^2} \sum_{k=0}^{N} \left(x - \frac{k}{N}\right)^2 \binom{N}{k} x^k (1-x)^{N-k} = \frac{2M}{N\delta^2} x(1-x).$$

Since $x \in [0,1]$,

$$\frac{2M}{N\delta^2} x(1-x) \leq \frac{M}{2N\delta^2}.$$

So we find that the upper bound on $|S_2|$ is

$$|S_2| \leq \frac{M}{2N\delta^2}.$$

The sum of our two upper bounds results in an upper bound for the difference between f and its Bernstein polynomial approximation:

$$|f(x) - B_N(x)| \leq \frac{\epsilon}{2} + \frac{M}{2N\delta^2}$$

for all $x \in [0,1]$. So if we choose $N \in \mathbb{N}$ such that $\frac{M}{2N\delta^2} < \frac{\epsilon}{2}$, *i.e.*,

$$N > \frac{M}{\delta^2\epsilon},$$

then $|f(x) - B_N(x)| < \epsilon$ for all $x \in [0,1]$. $\square$

Thus we can choose a polynomial approximation of any degree greater than $\frac{M}{\delta^2\epsilon}$, and we may select the degree to be even. If the Bernstein polynomial approximation to f is coercive, then we can show that there exists a finite t and a region over which $F(x,t)$ is convex. But is the Bernstein polynomial approximation coercive?

Properties of Bernstein Polynomials

The Bernstein polynomials were formulated by S. N. Bernstein to derive a constructive proof of the Weierstrass Approximation Theorem, as outlined above. There are other polynomial bases, but they are ill-suited for this approach. For example, Taylor polynomials are applicable only to infinitely-differentiable functions, and even then may or may not converge uniformly to the function. The interpolating polynomials at equally-spaced points do not necessarily converge uniformly to the function, which rules them out [27].

At first, Bernstein polynomials seem a strange choice for constructing the Weierstrass polynomial. They are not orthogonal, and converge extremely slowly to the function. In fact, they do not exactly reproduce polynomials of degree higher than one. Only in the limit does the Bern-

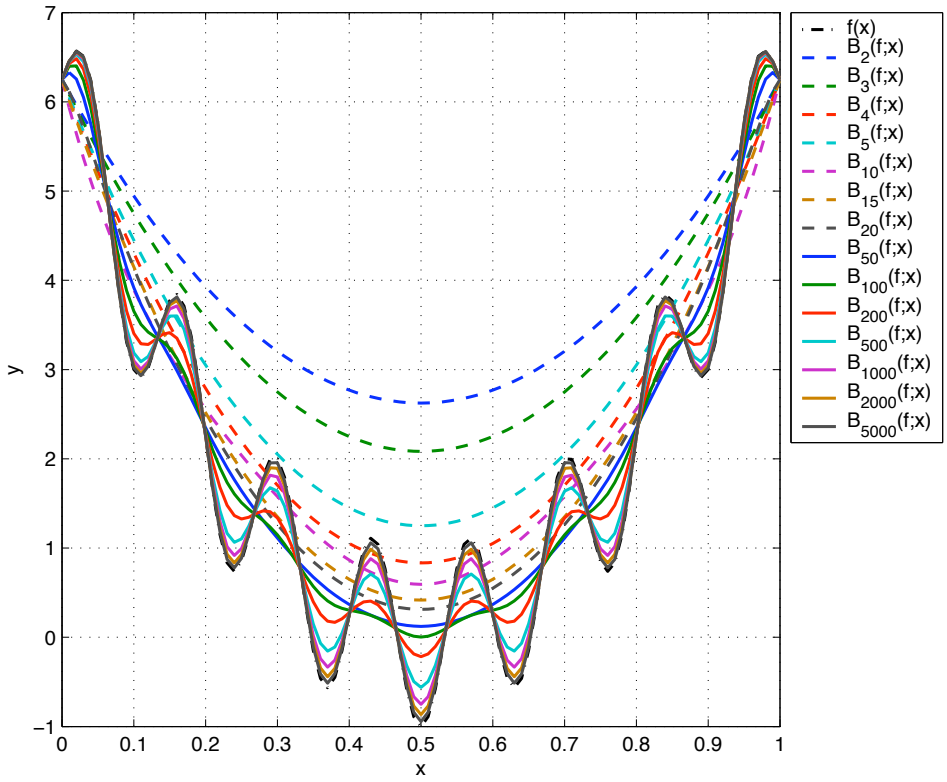


Figure 3.2. Successive Bernstein polynomial approximations to Griewank function $f(x) = 25(x-1/2)^2 - \cos(15\pi(x-1/2))$.

stein polynomial approximation of x^2 converge. Voronovskaya's Theorem [16, p. 22] states that for all twice differentiable functions, the Bernstein polynomials converge uniformly to the function like $1/n$. The convergence of the Bernstein polynomials is illustrated for the Griewank function in Figure 3.2.

We define Δ to be the forward difference operator:

$$\Delta f(x_j) = f(x_{j+1}) - f(x_j) = f(x_j + h) - f(x_j)$$

with step size $h > 0$. We further define the kth forward difference operator

$$\Delta^k f(x_j) = \sum_{r=0}^{k} (-1)^r \binom{k}{r} f(x_j + (k-r)h)$$

with step size $h > 0$. Using these above definitions, we can express the Bernstein polynomial approximation of f on $[0, 1]$ in the form

$$B_n(f; x) = \sum_{r=0}^{n} \binom{n}{r} \Delta^r f(0) x^r, \tag{3.9}$$

with step size $h = 1/n$. This formulation is equivalent to (3.2) [27].

From the Mean Value Theorem, we know that there is a relationship between finite differences and derivatives: There exists a $\xi \in (x_0, x_m)$ (where $x_m = x_0 + mh$) such that

$$\frac{\Delta^m f(x_0)}{h^m} = f^{(m)}(\xi).$$

Suppose that $f(x) = x^k$, and $n \geq k$. Then we have

$$n^r \Delta^r f(x) = 0 \text{ for } r > k$$

and

$$n^k \Delta^k f(0) = f^{(k)}(\xi) = k!$$

So if we express the Bernstein polynomial approximation as

$$B_n(x^k; x) = a_0 x^k + a_1 x^{k-1} + \cdots + a_{k-1} x + a_k,$$

then for $k < 2, a_0 = 1$, but for $k \geq 2$

$$a_0 = \binom{n}{k} \frac{k!}{n^k} = \left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right),$$

confirming that the Bernstein approximation does not reproduce polynomials of degree higher than one.

While this seems disappointing, there are several important conclusions to be gleaned from it. First, the Bernstein polynomial approximation of a polynomial is never of higher degree than the polynomial itself. Also, we can see from the formulation of a_0 that the kth derivative of the

Bernstein polynomial approximation of a coercive polynomial of degree k is positive. Thus the diffusion transform has the same effect on the Bernstein polynomial approximation of the coercive polynomial.

The Bernstein polynomial $B_n(f;x)$ can also be written in the form of a Stieltjes integral in the variable t [16],

$$B_n(f;x) = \int_0^1 f(t) \frac{\partial K_n(x,t)}{\partial t} dt,$$

where the kernel function

$$\begin{aligned} K_n(x,t) &= \sum_{r \le nt} \binom{n}{r} x^r (1-x)^{n-r}, \ \ 0 < t \le 1, \\ K_n(x,0) &= 0, \end{aligned}$$

which is constant on any interval $r/n \le t < (r+1)/n$, $r = 0, 1, \ldots, n-1$, and has the jump

$$\binom{n}{r} x^r (1-x)^{n-r}$$

at the basic point of interpolation $t = r/n$.

This formulation elucidates the smoothing properties of the Bernstein operator. Recall the illustration of the Bernstein approximations of the Griewank function in Figure 3.2. For lower order kernel functions, we see that the Bernstein approximation is smooth. It requires a higher degree approximation before the details of the function show up in the Bernstein approximation.

Lorentz discusses the theory of Bernstein polynomials on an infinite interval in [16]. First, suppose that $f(x)$ is defined on the interval $[0, b], b > 0$. To obtain the set of Bernstein polynomials $B_n(f;x;b)$ for this interval, we simply substitute $y = x/b$ into the polynomial $B_n(\phi;x;b)$ for $\phi(y) = f(b,y), 0 \le y \le 1$, and we have

$$B_n(f;x;b) = \sum_{r=0}^{n} f\left(\frac{br}{n}\right) \binom{n}{r} \left(\frac{x}{b}\right)^r \left(1 - \frac{x}{b}\right)^{n-r}.$$

For constant b this is simply a transformation to another interval. But assume that $b = b_n$ is a function of n increasing with n to $+\infty$, and that $f(x)$ is defined in the infinite interval $0 \le x < +\infty$.

In order for B_n to converge uniformly to any reasonably general $f(x)$, we must assume that the distance between two adjacent points $b_n/n \to 0$ for $n \to \infty$; in other words, $b_n = o(n)$. For example, if we approximate $f(x) = x^2$ on $[0, b_n]$, we obtain

$$B_n(x^2; x; b) = \left(1 - \frac{1}{n}\right) x^2 + \frac{b_n}{n} x.$$

In order for $B_n(x^2; x; b)$ to converge uniformly to x^2, the ratio of b_n and n must tend toward zero as n grows. Likewise, the Bernstein polynomial approximation of $f(x) = e^x$ on $[0, b_n]$ is

$$B_n(e^x; x; b) = \left[1 + \left(e^{b_n/n} - 1\right) \frac{x}{b_n}\right]^n,$$

and in order for $B_n(e^x; x; b)$ to converge uniformly to e^x, the ratio of b_n and n must vanish as n grows [2].

Two theorems by Chlodovsky, reproduced here from [16, p. 36], provide insight into the Bernstein polynomials on an unbounded interval.

Theorem 3.6. Chlodovsky's Theorem for bounded functions. *If $b_n = o(n)$ and the function $f(x)$ is bounded on $[0, +\infty)$, say $|f(x)| \leq M$, then $B_n(f; x; b) \to f(x)$ holds at any point of continuity of the function f.*

The significance of this theorem is that the interval can be expanded arbitrarily, and the Bernstein polynomial approximation still converges to $f(x)$, for bounded $|f|$.

Theorem 3.7. Chlodovsky's Theorem for unbounded functions. *Let $M(b)$ denote the maximum of $|f(x)|$ for $0 \leq x \leq b$. If $b_n = o(n)$ and*

$$M(b_n) e^{-\alpha n/b_n} \to 0$$

for each $\alpha > 0$, then $B_n(f; x; b) \to f(x)$ holds at each point of continuity of the function $f(x)$.

So if $|f(x)|$ grows at a less than exponential rate, the Bernstein polynomial approximation of f on an infinite interval converges uninformly to f.

In [2], Chlodovsky gives the following corollary to the above theorem:

Theorem 3.8. Chlodovsky's Corollary for exponentially growing functions. *If $f(x)$ is continuous on $0 \leq x < \infty$ and the inequality*

$$|f(x)| < Ce^{x^p}$$

holds for all x (where C and p are arbitrary finite constants), then $B_n(f;x;b)$ converges uniformly to $f(x)$ provided that the sequence b_n satisfies the condition

$$b_n < n^{\frac{1}{p+1+\eta}},$$

where $\eta > 0$ is arbitrarily small.

Chlodovsky also proved that the first derivative of $B_n(f;x;b)$ converges uniformly to the first derivative of f. The argument could be extended to higher order derivatives.

Chlodovsky originally conjectured that f need be only Riemann integrable, but Impens and Vernaeve [10, 11] showed that the arithmetic mean of the function evaluated at the Bernstein points must approach the continuous mean of the function as the number of points grows. The types of functions for which we seek a Bernstein polynomial approximation satisfy this condition because they are continuous.

It is easy to see that the above results on the interval $0 \leq x < \infty$ can be extended to the whole real line. Thus the Bernstein polynomial approximation of a coercive function bounded by $Ce^{|x|^p}$ exists and converges uniformly to that function on the real line.

Theorem 3.9. *Let $f(x)$ be a continuous, coercive function on $\mathbb{R}$ bounded by $Ce^{|x|^p}$ (where C and p are arbitrary, finite constants), and let $F(x,t)$ be its diffusion transform. Then for any constant M and $r > 0$ (depending on M) such that $f(x) \geq M$ whenever $\|x\| \geq r$, there exists a finite $\tau > 0$ such that $F(x,t)$ is convex for $\|x\| < r$ and $t \geq \tau$.*

Proof: Let $B_n(f;x;b)$ be the Bernstein polynomial approximation to f on $[0, b_n]$. Because f is coercive and B_n converges uniformly to f, B_n must be a coercive polynomial. By Theorem 3.4, for

any constant M and $r > 0$ (depending on M) such that $B_n(f; x; b) \geq M$ whenever $\|x\| \geq r$, there exists a finite $\tau > 0$ such that the diffusion transform of B_n is convex for $\|x\| < r$ and $t \geq \tau$.

Since the derivatives of B_n converge uniformly to the derivatives of f, in the case of $f \in C^\infty$, we can use the Taylor series diffusion operator to see that the diffusion transform of B_n converges uniformly to $F(x,t)$. Even if $f \notin C^\infty$, we can use the integral formulation to see that the diffusion transform of B_n converges uniformly to $F(x,t)$ for integrable f. Since B_n converges uniformly to f, the integral

$$\frac{1}{\sqrt{4\pi t}} \int_{\mathbb{R}} B_n(f; y; b_n) \exp\left(-\frac{(x-y)^2}{4t}\right) dy$$

converges uniformly to $F(x,t)$. Thus there exists a finite $\tau > 0$ such that $F(x,t)$ is convex for $\|x\| < r$ and $t \geq \tau$. $\square$

We have shown that diffusion damps high-frequency oscillations and causes a coercive function to become convex in a region containing the global minimum. But what does this suggest about using diffusion as the smoother for a smoothing and continuation optimization method?

3.3 Robustness of DEM

Like any global optimization method, the DEM is not foolproof. Under certain conditions, the method may not find the global minimum. There are two primary types of conditions under which the DEM may fail.

First, the method may not find a global minimum in a narrow well if it is overshadowed by a wider but shallower local minimum. Moré and Wu describe this shortfall in the Gaussian transform approach to the search for a global maximum: "In particular, the Gaussian transform eliminates tall, narrow hills; hence, if the global maximizer lies in one of these hills, the continuation approach is likely to fail" [20]. Figure 3.3 illustrates a case for which the DEM fails when diffusion is carried out until only one minimum remains.

Second, the method can fail due to implementation error. If the function is not sufficiently deformed in the initial diffusion phase, the local optimization method used inside the continuation loop may converge to a local minimizer rather than the global minimizer (see Figure 3.4). Also, if

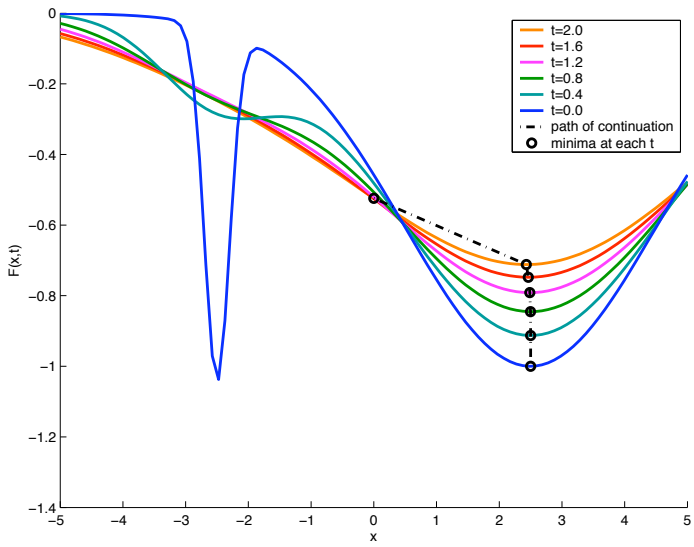


Figure 3.3. Function $f(x) = -\exp(-(x+2.5)^2/0.04) - \exp(-(x-2.5)^2/4)$ for which DEM fails: Solid curves represent $F(x,t)$ for various values of t. Dotted line traces between minima found at each step of DEM, leading to wrong extremum.

the continuation steps are too large, the starting point for the next continuation step may not be close enough to the global minimum to converge to it.

3.4 Implementation

Diffusion is traditionally implemented by performing the kernel integration described in Section 3.1. Moré and Wu have determined a method for reducing the integral to one dimension for certain classes of functions. This is particularly important in the application that they propose, because it involves millions of unknowns.

While this reduction in dimensionality works for the classes of functions that Moré and Wu optimize, kernel integration of general functions cannot be reduced in this manner. Furthermore, not all functions can be analytically integrated against this kernel, especially functions that have no analytic form.

Numerical quadrature could be used to perform the kernel integration, but for small values of t, numerical quadrature requires more function evaluations than finite differencing in order to achieve the same degree of accuracy. For a small t, the integral is heavily biased toward the

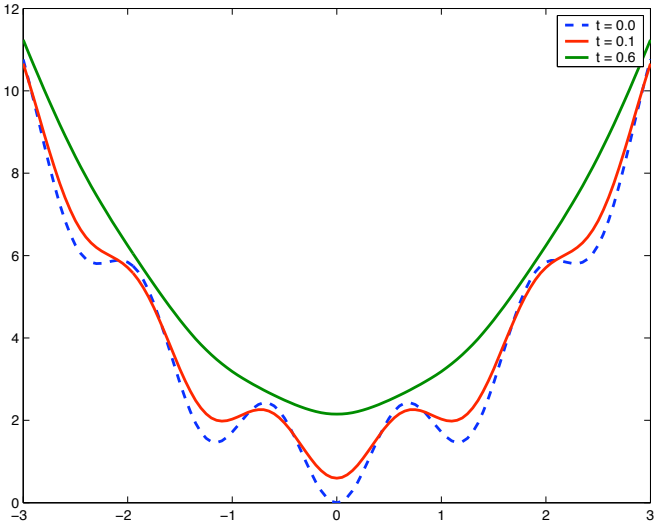


Figure 3.4. Insufficient diffusion: Griewank function $f(x) = x^2 + 1 - \cos(x)$: Original function (dashed line) and function smoothed using diffusion transform at $t = 0.1$ and $t = 0.6$ (solid red and green lines, respectively). A local optimization method may converge to a local minimum if diffusion insufficient.

immediate neighborhood of the point at which we are computing the integral, because the width of the Gaussian kernel function in equation 3.1 is proportional to $\sqrt{t}$. A quadrature rule, regardless of type, samples the function throughout the entire region. Thus some function evaluations far from the central point would have a negligible effect on the integral, yet those points are required by the quadrature. In order to sample more points near the center, a higher-order scheme must be used.

Finite differencing, on the other hand, begins sampling at points immediately adjacent to the central point, and sampling spreads away from that point at each subsequent step of diffusion. The number of points needed is directly proportional to the size of t and the degree of accuracy required.

Because the objective function arising from the magnetotelluric geoprospecting problem has no convenient analytical form, we cannot perform the analytical kernel integration. We therefore implement the diffusion transformation using finite differences because of its advantages over quadrature.

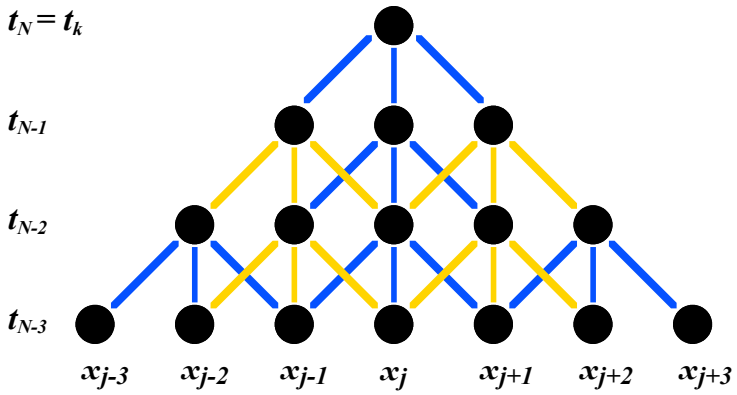


Figure 3.5. Recursive dependence between function values at different time steps.

3.4.1 Numerical Implementation

Evaluation of our objective function is expensive, so an efficient implementation is essential. We have developed a non-recursive method for the finite differencing. Our application of finite differencing differs from the traditional use, since we seek the value of the diffused function at only one point at a time, rather than a whole grid of values. Thus we build a finite difference matrix centered about that point, and use it to determine at what points the original objective function must be evaluated, as well as the weighting of those evaluations in the overall sum.

There is a recursive dependence between the function value $F_j^n \equiv F(x_j, t_n)$ and the value of its nearest neighbors at time step $n-1$. This dependence is illustrated in Figure 3.5. An analytical formula for the weights using trinomials was found for one-dimensional diffusion. The coefficient c_j for the point $x \pm j\Delta x$ in an N-step rule is computed by

$$c_j = \sum_{k=0,2,\ldots,N/2} a^{N-j-k}\, b^{k+j} \binom{N}{N-k/2} \binom{N-k/2}{N-j-k}$$

where $a = 1 - 2b$ and $b = \nu = \Delta t/(\Delta x)^2$. This formula was difficult to generalize to higher dimensions, however, so instead the structure of the finite difference matrix M was determined.

In explicit form, M is of dimension $(2N+1)^p \times (2N+1)^p$, where N is the number of steps of diffusion, and p is the number of dimensions. For $p > 1$, only a fraction of those points are actually

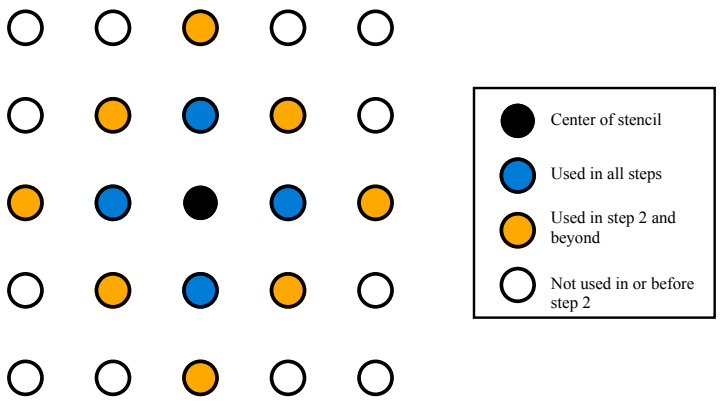


Figure 3.6. Shape of multi-dimensional finite difference stencil.

needed in the diffusion, as illustrated in Figure 3.6. The points that are needed and their respective weights are determined by the nonzeros in the product $M^N e_k$, where e_k is the kth (middle) column of the $(2N+1)^p \times (2N+1)^p$ identity matrix, with $k = \lceil (2N+1)^p/2 \rceil$. The matrix M is formed explicitly by the algorithm in Figure 3.7.

One potential drawback of the finite-differencing implementation of the DEM is the large number of function evaluations required to perform a single evaluation of the diffused function. While the number of function evaluations does grow exponentially with the number of dimensions p, as the dimensionality rises the ratio of number of evaluations to N^p decreases. The numbers of function evaluations required for one evaluation of the objective function are tabulated in Table 3.1.

It turns out that the number of function evaluations for N steps and p dimensions is the Delannoy number $D(N,p)$, described in [31] and [4, pp. 80–81]. The Delannoy numbers $D(p,q)$ also represent the total number of minimal paths with diagonal steps from the origin to (p,q). The Delannoy numbers can be derived recursively from the formula

$$D(p,q) = D(p,q-1) + D(p-1,q-1) + D(p-1,q)$$

and initial conditions $D(0,q) = D(p,0) = 1$. The Delannoy numbers are symmetric, *i.e.*, $D(p,q) =$

$np = (2N+1)^p$ *(size of matrix)*
$\nu_i = \gamma \Delta t/(\Delta x_i)^2$ $(0 < \gamma \leq 1,\ i = 1, \ldots, p)$
$a = 1 - 2\sum_i \nu_i$ *(diagonal elements of* M*)*
(Initialize matrix M *of dimension* $np \times np$ *to zero)*

```
for i = 0 : np − 1,
    M_{i,i} = a;
    k = 1;
    for j = 0 : p − 1,
        if ((i + k) < np)
            if ((i + k)/(k ∗ (2N + 1)) == i/(k ∗ (2N + 1)))
                M_{i+k,i} = ν_j;
                M_{i,i+k} = ν_j;
            end if
        else
            break;
        end if
        k = k ∗ (2N + 1);
    end for
end for
```

Figure 3.7. Algorithm for creating finite difference matrix.

N	p							
	0	1	2	3	4	5	6	7
0	1	1	1	1	1	1	1	1
1	1	3	5	7	9	11	13	15
2	1	5	13	25	41	61	85	113
3	1	7	25	63	129	231	377	575
4	1	9	41	129	321	681	1289	2241
5	1	11	61	231	681	1683	3653	7183
6	1	13	85	377	1289	3653	8989	19825
7	1	15	113	575	2241	7183	19825	48639

Table 3.1. Number of function evaluations required to perform one evaluation of diffused objective function for various numbers of steps N and dimensions p, corresponding to the Delannoy number $D(N, p)$.

$D(q, p)$. A non-recursive formulation for the Delannoy numbers is

$$D(p, q) = \sum_{d=0}^{\min(p,q)} 2^d \binom{p}{d} \binom{q}{d}.$$

3.5 Initialization of DEM

The question remains how much to diffuse the objective function. Conceptually, the objective function must be smoothed to the point that all other local minima are suppressed and only one minimum remains. But how can we know when we have reached that point?

We have shown that the diffusion process causes a coercive function to become convex in a region containing the global minimum. As t grows, that region also grows, but a non-convex function will not become convex everywhere for finite t. Figure 3.9 shows that the Rosenbrock function becomes more convex as t increases, but always exhibits non-convexity at a distance from the unique minimum.

While convexity is a sufficient condition for the existence of a unique minimum, it is not a necessary condition. Indeed, a unimodal function is not necessarily convex yet possesses a unique minimum. Nonetheless, as the objective function is smoothed it becomes more nearly convex, and therefore it becomes increasingly likely to possess a unique minimum.

We propose a heuristic inspired by convexity. If $F(x, \tau)$ satisfies the following conditions, then

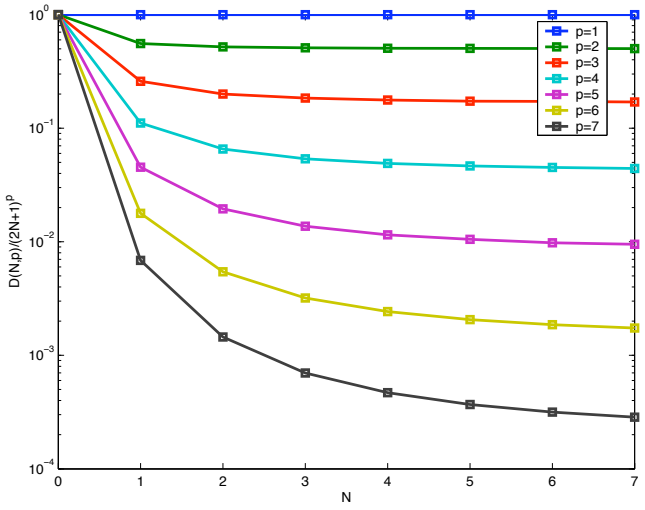
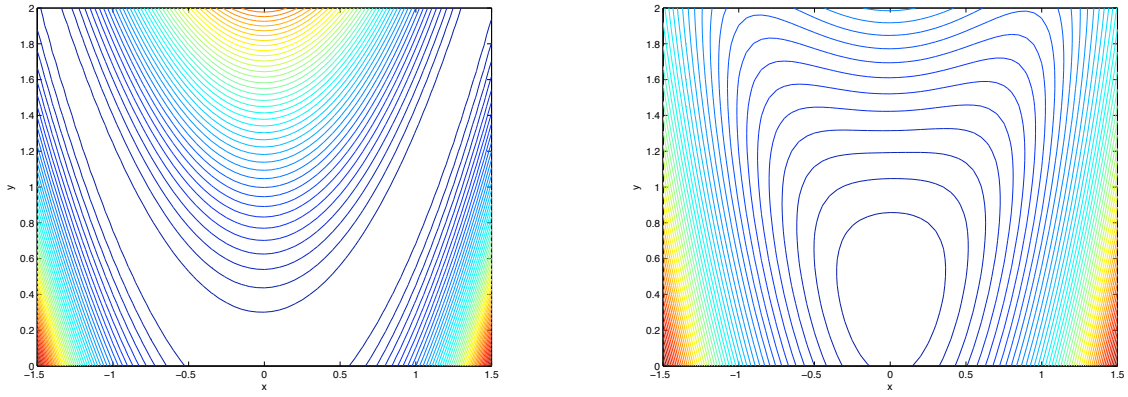


Figure 3.8. Ratio $D(N,p)/(2N+1)^p$ of number of points in p-dimensional, N-step finite difference stencil to number of points in hypercube of side length $(2N+1)$ containing the stencil.



(a) (b)

Figure 3.9. Diffusion causing convexity in Rosenbrock function $f(x,y) = 100(y-x^2)^2 + (1-x)^2$: (a) original function and (b) function smoothed using diffusion transform at $t = 0.2$.

sufficient diffusion is deemed to have occurred:

1. *Uniqueness:* A local optimization method converges to the same minimum x^*, independent of starting point.

2. *Convexity/Unimodality:* For all y, the line between $(y, F(y,\tau))$ and $(x^*, F(x^*,\tau))$ must either lie above the point $((1-\alpha)y + \alpha x^*, F((1-\alpha)y + \alpha x^*, \tau))$ for any $\alpha \in (0,1)$ (convexity), or $F((1-\alpha_1)y + \alpha_1 x^*, \tau) > F((1-\alpha_2)y + \alpha_2 x^*, \tau)$ for $\alpha_1 < \alpha_2$ (unimodality).

If the first condition is not met, the function definitely does not have a unique minimum and it must be deformed further. If the convexity condition is not met, it could be caused by a barrier between y and x^*. In this case we check for unimodality along the line between $(y, F(y,\tau))$ and $(x^*, F(x^*,\tau))$, and if the function is not unimodal then it must be deformed further.

We wrote a program implementing this heuristic. The user inputs a starting guess for t, or a default value is provided. A starting point x_0 is randomly selected and the diffused objective function is optimized using Nelder–Mead, a particularly simple optimization method (see Section 3.7.2). Then another point is selected, and optimization is again performed. Periodically, the optimization method is interrupted and the convexity of the function is tested. If the function fails the convexity test, then a unimodality test is performed in the region where the convexity test failed, to make sure that the function is consistently decreasing toward the minimum. When Nelder–Mead is finished, the point to which it converges is compared to the point to which it converged from x_0.

If the unimodality test fails, or any optimization converges to a different point, then t is deemed too small and testing restarts with a larger t. The value of t could be increased linearly or geometrically, but in our implementation, t doubles at each step.

The points at which the method begins are selected using a pseudorandom number generator. The seed is reset and the same points are selected for every value of t. We believe that this is a good strategy, because it allows us to track the progress of diffusion as it smooths the objective function. We can see whether this degree of diffusion has adequately suppressed a particular local extremum that caused one of the tests to fail in the previous step.

Using this method, it is easy to determine when a function is not unimodal. As soon as one of the minimization or line tests fails, the function is definitely not unimodal. But if the tests are passed, this does not necessarily mean that the function is unimodal. It is possible that we did not happen to select a point for which one of these tests would fail. It is impossible to examine enough points to guarantee unimodality, so instead we choose a number (a function of the dimensionality of the objective function) at which the passing of these tests would give us sufficient confidence that the objective function is unimodal.

3.6 Continuation

In the continuation steps, we seek a sequence $(x_k, F(x_k, t_k))$ that converges to the global minimum x^*. The number of steps and the difference Δt_k between steps must be determined. If too few intermediate steps are taken, the method may not converge to the global minimum. On the other hand, if too many steps are taken, computing power is wasted.

We study the effect of diffusion on the extrema of a function. We first examine the theoretical underpinnings of continuation, and then we observe the behavior of the extrema under diffusion. Finally, we suggest some heuristics for determining the optimal sequence of continuation steps.

3.6.1 Continuation Theory

As $\Delta t_k \to 0$, the sequence of points $(x(j\Delta t_k), F(x(j\Delta t_k), j\Delta t_k)), j = 0, 1, \ldots$ approaches a curve $x(t), t \geq 0$, where each $x(t)$ minimizes $F(x, t)$. Thus each $x(t)$ is a stationary point of $F(x, t)$, so

$$\frac{\partial F(x(t), t)}{\partial x} = 0.$$

Since x is a function of t, the solution of the ODE

$$\frac{\partial^2 F(x(t), t)}{\partial x^2} \frac{dx(t)}{dt} + \frac{\partial^2 F(x(t), t)}{\partial t \, \partial x} = 0$$

traces the parameterized curve [20], and with initial condition $x(0) = x^*$, this uniquely defines the path traced to the minimum by the DEM, provided that the Jacobian is nonsingular.

Moré and Wu analyze the situation where it is possible to determine a global minimizer x_k of $F(x, t_k)$ for some sequence $\{t_k\}$ converging to zero. They show that $\frac{\partial F(x(t),t)}{\partial x}$ is continuous with respect to x and t individually, and then they prove joint continuity with respect to (x, t) on $\mathbb{R}^n \times \mathbb{R}$. The proof from [20] is reproduced below.

Theorem 3.10. Theorem of joint continuity. *Assume that $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuous on $\mathbb{R}^n$ and $|f(x)| < C_1 e^{C_2\|x\|}$ (for positive C_1 and C_2). If the sequence $\{x_k\}$ converges to x^* and $\{t_k\}$ converges to zero, then*

$$\lim_{k \to +\infty} F(x_k, t_k) = f(x^*).$$

Proof: Let B_r be the ball of radius r centered at the origin, and let C_r be the complement of B_r, that is,

$$C_r = \{x \in \mathbb{R}^n : \|x\| > r\}.$$

We first show that for any $\epsilon > 0$ we can choose $r > 0$ and k_0 so that

$$\int_{C_r} |f(x_k + t_k u) - f(x^*)| \exp(-\|u\|^2) du \leq \epsilon, \qquad k \geq k_0. \tag{3.10}$$

Because we have assumed that f is bounded in absolute value by an exponential, there is a constant $\mu > 0$ such that

$$|f(x_k + t_k u) - f(x^*)| \leq \mu \exp(t_k \|u\|),$$

and since $t\|u\| \leq \frac{1}{2}\|u\|^2$ for $t \leq \frac{1}{2}$ and $\|u\| \geq 1$,

$$\int_{C_r} |f(x_k + t_k u) - f(x^*)| \exp(-\|u\|^2) du \leq \mu \int_{C_r} \exp(-\frac{1}{2}\|u\|^2) du$$

if $t_k \leq \frac{1}{2}$ and $r \geq 1$. This inequality proves (3.10) because, if r is sufficiently large, the integral $\int_{C_r} \exp(-\frac{1}{2}\|u\|^2) du$ is arbitrarily small. Now note that the continuity of f at x^* shows that for

given r and k_0 we can choose $k_1 \geq k_0$ so that

$$\int_{C_r} |f(x_k + t_k u) - f(x^*)| \exp(-\|u\|^2) du \leq 2\epsilon, \qquad k \geq k_1.$$

This inequality and (3.10) imply that

$$|F(x_k, t_k) - f(x^*)| \leq 2\epsilon, \qquad k \geq k_1,$$

which is the desired result. □

We can use the results from the above theorem to prove the following theorem.

Theorem 3.11. *Assume that $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuous on $\mathbb{R}^n$ $|f(x)| < C_1 e^{C_2\|x\|}$ (for positive C_1 and C_2). Let $\{t_k\}$ be any sequence converging to zero. If x_k is a global minimizer of $F(x, t_k)$ and $\{x_k\}$ converges to x^*, then x^* is a global minimizer of f.*

Proof: Since x_k is a global minimizer of $F(x, t_k)$,

$$F(x_k, t_k) \leq F(x, t_k), \qquad x \in \mathbb{R}^n.$$

Theorem 3.10 now implies that $f(x^*) \leq f(x)$ for any $x \in \mathbb{R}^n$. Hence, x^* is a global minimizer of f. □

Moré and Wu have shown that if $\{x_k\}$ is a sequence of global minimizers of $F(x, t_k)$, the sequence converges to the global minimizer of $f(x)$. But sometimes the $\{x_k\}$ we obtain are not global minimizers of $F(x, t_k)$. As we have already indicated, there are cases for which the global minimum cannot be found, or where the continuation steps are too large.

3.6.2 Behavior of Extrema under Diffusion

Diffusion causes local extrema to move. Over time, neighboring extrema drift toward one another and combine. We see an example of such behavior in Figure 3.10, in which the extrema of the diffused sixth-degree Legendre polynomial are tracked over time.

We can treat the graph in Figure 3.10 as a plot of displacement over time. We can determine the

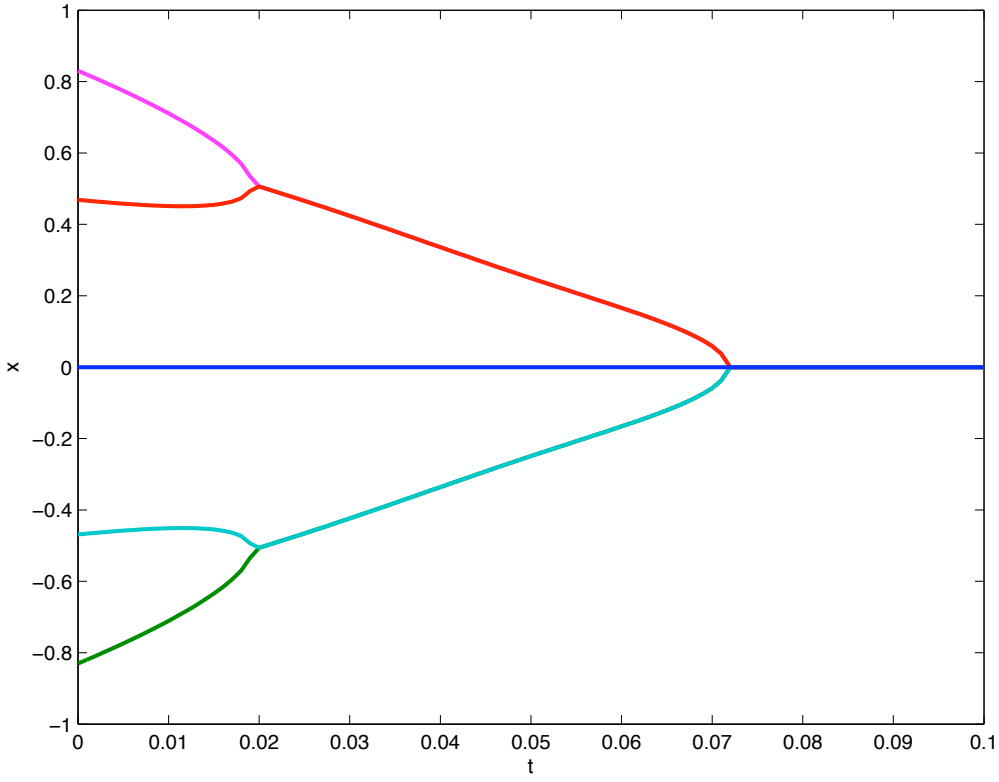


Figure 3.10. Agglomeration of extrema of sixth-degree Legendre polynomial under diffusion. Curves trace five extrema of the polynomial as t increases.

"velocity" of a given extremum by calculating the slope of the curve representing its displacement, and the "acceleration" of a given extremum by calculating the curvature of the displacement curve. For most values of t, the extrema move at a constant velocity, as evidenced by the straightness of the curve on the graph. At certain times, such as $t \approx 0.02$ and $t \approx 0.07$ in the case of the sixth-degree Legendre polynomial, pairs of extrema join and anihilate. At times just before the pairs join, we can see that the curvature of the curves increases, marking an acceleration of extrema as they approach one another. From these observations we can glean that a sudden change in displacement could mark a point at which bifurcation of extrema will occur when diffusion is reversed.

3.6.3 Heuristics for Continuation

It seems important to proceed with small steps at times close to the bifurcation of extrema, but realistically, we cannot determine in advance the times at which we must be cautious. Because a sudden change in displacement could indicate a bifurcation, we could set a threshold value limiting the distance our minimum can move at each continuation step, with the rationale that a fast-moving minimum is a sign that the topography of the objective function is changing drastically. If the distance the minimum moves exceeds the threshold value, we could go back to the previous step and subdivide it into two continuation steps, repeating the subdivision if necessary until the distance moved between steps does not exceed the threshold value. But this method assumes that our continuation steps are already small enough to detect the acceleration of the extrema pairs. If the continuation steps are large, the acceleration of the extrema pairs may be lost in the averaging over time.

Another method for making certain that the size of the continuation step is small enough would be to run the optimization algorithm twice: once with larger steps and once with smaller steps. If both runs find the same optimum, then the larger step size is acceptable. This works, of course, only if the smaller step size is sufficiently small that it finds the optimum.

3.7 Local Methods in Inner Loop

In each continuation step, a minimum is found using a local optimization method. In our implementation of the DEM, we have taken two different approaches to optimization by using two fundamentally different local optimization methods: BFGS and Nelder–Mead.

3.7.1 BFGS

The BFGS method, named for its four inventors Broyden, Fletcher, Goldfarb, and Shanno, is a quasi-Newton optimization algorithm that uses a quadratic model of the objective function. At each step of the algorithm, the approximate Hessian is updated using recently obtained gradient information.

As a quasi-Newton method, a search direction is strategically chosen based on current approximate Hessian and gradient information, and the minimum along a line emanating from the current starting point is found. From the difference in optima and gradients from the current and previous steps, the approximate Hessian is updated. For many practical problems, BFGS exhibits superlinear convergence, and the approximate Hessian updating process is known to have self-correcting properties [26].

For the magnetotelluric geoprospecting application, this method is unfortunately quite costly. Gradient information must be obtained through finite differencing, requiring $p + 1$ evaluations of the p-dimensional diffused objective function. In addition, each evaluation of the diffused objective function requires $D(N, p)$ evaluations of the original objective function for N steps of numerical diffusion. Thus, the cost of updating the Hessian alone exceeds $pD(N, p)$.

3.7.2 Nelder–Mead

The Nelder–Mead algorithm is a direct-search algorithm for unconstrained minimization. The Nelder–Mead algorithm maintains a simplex of approximations to the minimum, which is updated at each step. The vertices are sorted by ascending objective function values. At each step of the algorithm, the goal is to replace the worst vertex with a new vertex along a line that passes through the worst vertex and the centroid of the remaining points.

With the exception of shrink steps, the algorithm requires only two function evaluations per iteration. Shrink steps are sufficiently rare that the average number of evaluations per iteration is essentially two. In the case of evaluating the p-dimensional diffused objective function, with N steps of numerical diffusion, this method requires only $2D(N, p)$ function evaluations per step.

Unfortunately, the Nelder–Mead algorithm is known to stagnate at nonstationary points of certain smooth functions of low dimensionality [18]. Kelley has proposed a solution to this problem by using a sufficient decrease condition and applying an oriented restart to the simplex when necessary [12].

While one step of Nelder–Mead is less expensive than one step of BFGS, many more Nelder–Mead steps are typically required to find a minimum. Still, in our experience, Nelder–Mead finds the minimum of most of the sort of functions for which it is suitable to use the DEM with fewer overall function evaluations than BFGS.

3.8 Performance of DEM

The diffusion equation method for global optimization can be quite costly, particularly when the discrete transform is performed. We might therefore begin to wonder if it is worth all the expense. In this section, we present results of using the DEM on test objective functions, and compare the cost to the cost of brute force optimization methods.

3.8.1 Performance of DEM on Test Functions

The DEM was tested on three 2-D functions:

- Quartic test function: $f(x, y) = (\frac{1}{2}x^4 - \frac{1}{3}x^3 - \frac{1}{2}x^2 + 2)(\frac{1}{2}y^4 + \frac{1}{3}y^3 - \frac{1}{2}y^2 + 2)$, with global minimum $x^* = (1, -1)$.
- Rosenbrock function: $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$, with global minimum $x^* = (1, 1)$.
- Goldstein–Price function: $f(x, y) = (1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2))(30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2))$, with global minimum $x^* = (0, -1)$.

Five continuation steps were performed: $t = 0.20, 0.15, 0.10, 0.05, 0$. The Nelder–Mead optimization method was used within the continuation steps. Before the Nelder–Mead method terminates, the size of the simplex and the difference between the largest and smallest function values on the simplex must be smaller than their respective tolerances. With the rationale that the exact location and function value of the minimum at intermediate continuation steps is not as important as its exact location in the final step, we varied the tolerance at each step, tightening it as t approached zero. The simplex size tolerance varied linearly from 10^{-3} to 10^{-5}, and the function value tolerance varied linearly from 10^{-2} to 10^{-3}. We also ran the method with tolerances kept constant at the smallest values (10^{-5} and 10^{-3}, respectively) to compare the results.

Continuing with the rationale that the exact location and function value are unimportant until the final step, we performed the numerical diffusion transform with varying coarseness of finite differencing. A larger Δt means that the diffused objective function value is less accurate. We compared the outcome and expense of $\Delta t = 0.05$ and $\Delta t = 0.025$.

Because of the simple nature of the test functions, we were also able to perform analytical diffusion upon them. We tested its performance with varying and constant tolerances too. Finally, we compared the performance of the DEM with MAPS, a model-assisted pattern search algorithm developed by Siefert et al. [30]. The user provides MAPS with a region in which to search for the minimum, a function evaluation allowance, and a model size, which should be less than the function evaluation allowance. Using a number of function evaluations equal to the model size, MAPS forms a model of the objective function, and from the model locates areas in which the global minimum is likely to be found. Within those areas, MAPS performs a pattern search until it has exhausted its function evaluation allowance. The outcome is described in Table 3.2.

In every case, the DEM successfully found the global minimum of the test functions. MAPS did not find the minima of the Rosenbrock and Goldstein–Price functions. This was expected in the case of the Rosenbrock function in particular, because the narrow, curved valley in which the minimum lies is especially pathological for the pattern search algorithm. Still, when MAPS is able to find the minimum, it does so with much less expense than even the analytical form of DEM.

		DEM			
		Numerical		Analytical	MAPS
	Tolerance	$\Delta t = 0.05$	$\Delta t = 0.025$		(model size)
Quartic Test Function	Vary	5043	15956	292	200 (180)
	Constant	7910	24617	403	
Rosenbrock Function	Vary	5566	17294	356	> 1000
	Constant	8453	26193	453	
Goldstein–Price Function	Vary	5386	17267	306	> 1000
	Constant	8477	26571	422	

Table 3.2. Comparison of performance of DEM and MAPS on two-dimensional test functions. Nelder–Mead used as inner local optimization routine, and function value and simplex size tolerances allowed to vary or kept constant. Numerical diffusion performed with $\Delta t = 0.05$ or 0.025; both numerical and analytical diffusion performed with five continuation steps $t = 0.20, 0.15, 0.10, 0.05, 0.0$.

As expected, tightening the tolerance as the DEM progresses results in a decrease in expense, and using a larger time step reduces the expense of the method by roughly a factor of three. Even so, requiring several thousand function evaluations to find the minimum of a simple test function hardly seems a bargain. How does the expense of the DEM compare to simple brute force methods?

3.8.2 Comparison of DEM and Brute Force Methods

A naïve method of locating the global minimum of a function might be to create a grid of points and evaluate the objective function at each point. The point with the smallest function value would be taken to approximate the global minimum, provided that the global minimum is located within the grid, and the spacing between points is small enough to detect the global minimum with adequate accuracy.

For the purpose of this comparison, we will restrict our concern to the location of the global minimum. We will define the accuracy of our optimization method, Δx, as the distance between neighboring grid points. This is similar to the stopping tolerance of Nelder–Mead, in which the largest distance between simplex points is used to determine whether the method has converged adequately.

On a one-dimensional interval of length one, if we wish to attain an accuracy of Δx, then we need at least $\lceil 1/\Delta x \rceil + 1$ points in the grid. On a two-dimensional interval, to attain an accuracy of Δx and Δy in the x and y directions, respectively, we would need at least $(\lceil 1/\Delta x \rceil + 1)(\lceil 1/\Delta y \rceil + 1)$

points in the grid. Let us suppose for the sake of simplicity that we desire the same accuracy in all coordinate directions. For a p-dimensional objective function, we need at least $(\lceil 1/\Delta x \rceil + 1)^p$ points. The cost of this method rises dramatically as the dimensionality of the objective function increases.

If we were to find the global minimum of the objective function with $\Delta x \approx 10^{-5}$ on a unit square, it would require roughly 10^{10} function evaluations. Reducing the accuracy by a factor of ten reduces the number of function evaluations by a factor of 100. Enlarging the region in which the minimum is sought only adds to the expense of the brute force method.

This method cannot attain the accuracy of the DEM for a reasonable number of function evaluations. First, the DEM is an unconstrained optimization method, so the location of the minimum is unrestricted. For the brute force method to work, we would need to know a priori that the minimum is located within the region. Second, the most expensive run of the DEM on the test functions required less than 27,000 function evaluations, or approximately 0.00027% of the evaluations required to find the minimum on a unit square with the brute force method.

As the dimensionality of the objective function increases, the cost of the DEM undoubtedly increases as well. The cost of numerical diffusion grows like the Delannoy number $D(n, p)$, where n is the number of steps of diffusion. (See Section 3.4.1 for a detailed discussion of the cost of numerical diffusion.) While $D(n, p) = O(n^p)$, the constant decreases as p increases. Furthermore, as the dimensionality of the objective function increases, the cost of the internal local optimization routine also grows. Still, it seems that the rising cost of the DEM would never exceed the cost of the brute force method.

Another approach might be to sample points randomly throughout the region, and select the point for which the function value is smallest as the global minimum. The probabilistic accuracy of this method is proportional to $1/\sqrt{N}$, where N is the number of points. The square-root relation is independent of the dimensionality of the objective function.

If we seek the global minimum of a two-dimensional objective function, this method performs no better than the uniform grid. It is at higher dimensions that this approach becomes competitive. For sufficiently high dimensionality p, this method may become more efficient than the DEM,

although the value of p would depend on the particular objective function and its behavior under diffusion.

Chapter 4

Implementation of Solution to Geoprospecting Problem

4.1 Overview of Algorithm

In searching for the solution of the magnetotelluric geoprospecting problem, the basic algorithm is to find the parameters that optimize the function Ψ from Equation (2.3), given the true measurements of the input α and resulting right-hand side b. The minimum of Ψ is sought using an optimization method. Nearly all optimization methods are inherently serial, because the choice of the next point depends on the results for previous points. The optimization step turns out to be only a small fraction of the overall calculation, however. The bulk of computation time is consumed by evaluation of the function Ψ, the norm of the difference between the calculated coefficients α and the true coefficients α_{true}. We present our approach for minimizing the time consumed in computing Ψ, and give computational results from our implementation.

4.2 Computation of Objective Function in Two Dimensions

In order to calculate the coefficients α, the matrix A must first be constructed. Each entry of the matrix is computed through quadrature:

$$\begin{aligned} a_{ij} &= -\frac{m^2+n^2}{16}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\sigma(x,y)(\cos(\omega x-\gamma y)+\cos(\omega x+\gamma y) \\ &\quad -\cos(\beta x+\gamma y)-\cos(\beta x-\gamma y)-\cos(\omega x-\delta y)-\cos(\omega x+\delta y) \\ &\quad +\cos(\beta x-\delta y)+\cos(\beta x+\delta y))dx\,dy, \end{aligned}$$

where $\omega = m-p, \beta = m+p, \gamma = n-q, \delta = n+q$; m, n, p, and q are the Fourier mode numbers; and (i, j) are determined by a function of m, n, p, q, and their respective maxima, M, N, P, and Q. The quadrature routine we use is DCUHRE, a Fortran routine from TOMS (#698). It is a robust and efficient quadrature routine, and performs well despite the difficulty of the integral.

4.3 Computation of Objective Function in Three Dimensions

In three dimensions, the entries of A are given by

$$\begin{aligned} a_{ij} &= -\frac{m^2+n^2+p^2}{32}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\sigma(x,y,z)\ (\cos(\omega x-\gamma y-\eta z)+\cos(\omega x-\gamma y+\eta z) \quad (4.1) \\ &- \cos(\omega x-\gamma y-\lambda z)-\cos(\omega x-\gamma y+\lambda z)+\cos(\omega x+\gamma y-\eta z)+\cos(\omega x+\gamma y+\eta z) \\ &- \cos(\omega x+\gamma y-\lambda z)-\cos(\omega x+\gamma y+\lambda z)-\cos(\beta x+\gamma y-\eta z)-\cos(\beta x+\gamma y+\eta z) \\ &+ \cos(\beta x+\gamma y-\lambda z)+\cos(\beta x+\gamma y+\lambda z)-\cos(\beta x-\gamma y-\eta z)-\cos(\beta x-\gamma y+\eta z) \\ &+ \cos(\beta x-\gamma y-\lambda z)+\cos(\beta x-\gamma y+\lambda z)-\cos(\omega x-\delta y-\eta z)-\cos(\omega x-\delta y+\eta z) \\ &+ \cos(\omega x-\delta y-\lambda z)+\cos(\omega x+\delta y+\lambda z)-\cos(\omega x+\delta y-\eta z)-\cos(\omega x+\delta y+\eta z) \\ &+ \cos(\omega x+\delta y-\lambda z)+\cos(\omega x+\delta y+\lambda z)+\cos(\beta x-\delta y-\eta z)+\cos(\beta x-\delta y+\eta z) \\ &- \cos(\beta x-\delta y-\lambda z)-\cos(\beta x-\delta y+\lambda z)+\cos(\beta x+\delta y-\eta z)+\cos(\beta x+\delta y+\eta z) \\ &- \cos(\beta x+\delta y-\lambda z)-\cos(\beta x+\delta y+\lambda z))\,dx\,dy\,dz, \end{aligned}$$

where $\omega = m - q, \beta = m + q, \gamma = n - r, \delta = n + r, \eta = p - s$, and $\lambda = p + s$; m, n, p, q, r, and s are the Fourier mode numbers; and (i, j) are determined by a function of m, n, p, q, r, s, and their respective maxima, M, N, P, Q, R, and S. We used Monte Carlo integration for the three-dimensional quadrature, as we explain next.

4.3.1 Monte Carlo Integration

As the dimensionality of a multiple integral increases, the cost of most quadrature routines grows rapidly. Conceptually, if we require N function evaluations for a given accuracy in one dimension, then for the same accuracy in two dimensions we need $O(N^2)$ function evaluations, and in k dimensions, $O(N^k)$ function evaluations.

Exceptions to this rule are *Monte Carlo methods,* in which the error estimate always goes to zero with $1/\sqrt{N}$, meaning that a hundred-fold increase in N results in an additional decimal digit of accuracy, independent of dimensionality. Compared to other quadrature methods, Monte Carlo methods perform poorly for one- and two-dimensional integrands, but outperform other methods for integrands of three or more variables.

The main idea is to sample the integrand at N points distributed throughout the region of integration, and then multiply the mean of the samples by the volume of the region. We approximate the integral $I(f) = \int_\Omega f(x)dx$ by

$$I(f) \approx S_N(f) = \frac{V}{N} \sum_{i=1}^{N} f(x_i),$$

where V is the volume of the region of integration. The resulting sum is an estimate to within a certain probability of the true value of the definite integral.

The *strong law of large numbers* makes it nearly certain that $S_N \to I$ as $N \to \infty$, but provides no information about the error $I - S_N$ for a given sample size [13]. We can obtain a probabilistic estimate of the error, however, by calculating the expected deviation of S from I.

The *standard deviation* $\sigma(S_N(f))$ characterizes the quadratic mean of the error $I - S_N$, and

$$\sigma(S_N(f)) = \frac{\sigma(f)}{\sqrt{N}},$$

where $\sigma(f)$ is the standard deviation of $f(x)$. Unfortunately, in general we do not know $\sigma(f)$, so instead we estimate the standard deviation of $S_N(f)$ as

$$\sigma(S_N(f)) = \left(\frac{1}{N(N-1)} \sum_{i=1}^{N} \frac{(f(x))^2}{\rho(x)} - S_N^2 \right)^{1/2},$$

where $\rho(x)$ is the density function associated with the particular Monte Carlo method ($\rho(x) = 1$ for the basic Monte Carlo method).

From the analytic formula for the standard deviation, we can see that the error in S_N shrinks with $1/\sqrt{N}$. Another way to derive this behavior is by using the *central limit theorem:*

$$\lim_{N\to\infty} P\left(a \le \frac{S_N(f) - I(f)}{\sigma(f) N^{-1/2}} < b \right) = \frac{1}{\sqrt{2\pi}} \int_a^b \exp(-t^2/2)\, dt. \tag{4.2}$$

The central limit theorem states that the distribution of the *standardized* random variable

$$\frac{S_N(f) - I(f)}{\sigma(f) N^{-1/2}}$$

converges to the normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. From (4.2), we can derive for any $\alpha \ge 0$

$$\lim_{N\to\infty} P\left(|S_N(f) - I(f)| \le \alpha \sigma(f) N^{-1/2} \right) = \frac{1}{\sqrt{2\pi}} \int_{-\alpha}^{\alpha} \exp(-t^2/2)\, dt. \tag{4.3}$$

By setting

$$\epsilon_N := \alpha \sigma(f) N^{-1/2} \quad \text{and} \quad \delta := \frac{1}{\sqrt{2\pi}} \int_{-\alpha}^{\alpha} \exp(-t^2/2)\, dt,$$

we can express (4.3) as

$$\lim_{N\to\infty} P(|S_N(f) - I(f)| \le \epsilon_N) = \delta.$$

Thus, the rate of convergence $\epsilon_N = O(N^{-1/2})$ as $N \to \infty$ holds for fixed likelihood δ of $|S_N(f) - I(f)| \le \epsilon_N$.

According to [13, p. 205], Monte Carlo methods do not in general integrate smooth functions more accurately or efficiently than they integrate non-smooth functions. So we can expect that

a Monte Carlo method tested on a smooth function will perform equally well for our non-smooth objective function.

VEGAS

The basic Monte Carlo method covers the region of integration with randomly distributed points. Faster convergence might be obained if rather than being distributed evenly throughout the region of integration, the points were strategically placed in regions that have the largest impact on the average value of the function. We could develop a density function $\rho(x)$ that weights the distribution of points toward regions having a large impact on the average value of the function. This changes our integral approximation to

$$I \approx S = \frac{V}{N} \sum_{i=1}^{N} \frac{f(x_i)}{\rho(x_i)},$$

where $\int_\Omega \rho(x)dx = 1$. In the case of the basic Monte Carlo method, $\rho(x) = 1$.

Ideally, the standard deviation of the integral is minimized when

$$\rho(x) = \frac{|f(x)|}{\int_\Omega |f(x)|dx}.$$

Lepage's VEGAS algorithm is an iterative, adaptive Monte Carlo algorithm that approximates this ideal density function by dividing the integration volume into hypercubes using a rectangular grid [15, 14]. In the initial steps, random points are evenly distributed through the equally-spaced hypercubes, and thus are uniformly distributed. From iteration to iteration, the hypercubes are concentrated in regions where $|f(x)|$ is largest, by adaptively adjusting the increment sizes on each axis. Because they are evenly distributed through the hypercubes, the sample points are concentrated in regions having a large impact on the average value of the function. Figure 4.1 illustrates the concentration of hypercubes.

For a given iteration, we see that

$$I \approx S_k = \frac{V}{N} \sum_{i=1}^{N} \frac{f(x_i)}{\rho(x_i)}$$

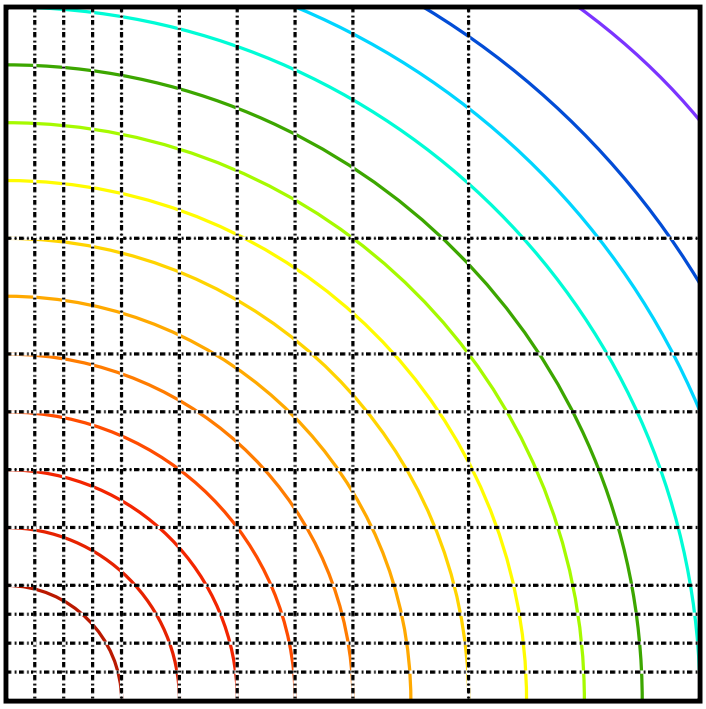


Figure 4.1. Concentration of hypercubes in VEGAS. Sample function largest in lower left corner, so hypercubes concentrated in that region.

with standard deviation

$$\sigma_k = \left(\frac{1}{N(N-1)} \sum_{i=1}^{N} \frac{(f(x))^2}{\rho(x)} - S_k^2 \right)^{1/2}.$$

And the cumulative estimate of the integral at step k is

$$\bar{S} = \bar{\sigma}^2 \sum_{j=1}^{k} \frac{S_j}{\sigma_j^2},$$

where

$$\bar{\sigma}^2 = \left(\sum_{j=1}^{k} \frac{1}{\sigma_j^2} \right)^{-1}.$$

The standard deviation σ_k of S_k at each iteration decreases until the optimal grid is obtained. While the standard deviation of an individual iterate no longer decreases after that point, the standard deviation of the composite estimate $\bar{S}$ continues to decrease like $1/\sqrt{mN}$, where m is the number of iterations.

We selected the VEGAS algorithm to perform our quadrature because the piecewise constant conductivity function creates two regions where the maximum absolute value of the integrand differs markedly. The two regions have different degrees of impact on the integral, and VEGAS should concentrate the hypercubes on the region with the largest impact, leading to faster convergence.

4.4 Parallelization of Computation of Objective Function

Each entry of the matrix is independent of the others, so they can be computed in parallel. We used our allocations on two NCSA Linux clusters and the CSE Turing cluster to develop MPI-based parallel algorithms for efficient function evaluation.

4.4.1 Parallelization in Two Dimensions

The size of the matrix can range from 9×9 to 100×100. This range represents the smallest size with significant physical meaning and a practical upper bound for computation. Our program evenly distributes this work across multiple processors using a manager-worker approach. After the matrix has been calculated, the coefficients α are determined by solving the corresponding linear system,

which takes a negligible amount of time because the matrix, although dense, is relatively small. The manager process runs the optimization routine, coordinates the division of the matrix into evenly sized sections and the distribution of the workload to the processors, compiles the matrix, and performs the function evaluation. After the worker processes have computed their portions of the matrix, they send the results back to the manager process, which then compiles all the data into one matrix, solves the linear system, computes the norm, and then presents the function evaluation to the optimization routine. Based on this result, the optimization routine then chooses a new point at which to invoke the function evaluation routine, and the procedure continues until the optimization routine terminates, presumably at a minimum of the objective function.

4.4.2 Parallelization in Three Dimensions

In the three-dimensional case, the smallest matrix with physical meaning is of dimension 27×27. The cost of function evaluation increases drastically for three dimensions, so we must streamline our algorithm to maximize efficiency. We first note that the matrix values are nearly symmetric, because a_{ij} is a constant multiple of a_{ji} (see Equation 4.2). We can take advantage of this symmetry and reduce the number of integrals performed by nearly a factor of two. Second, we changed the method of sharing the work between processors.

The manager process still runs the optimization routine, coordinates the division of the matrix into evenly sized sections and the distribution of the workload to the processors, compiles the matrix, and performs the function evaluation. But instead of distributing individual matrix entries to the worker processes, we instead take advantage of the parallelism inherent in Monte Carlo methods by having each worker perform Monte Carlo quadrature for each matrix entry. Because each worker performs an equal share of the total number of function evaluations required, this approach eliminates quadrature-associated load imbalance issues.

After the worker processes have computed their portions of the quadrature, they send the results back to the manager process, which then compiles all the data into one matrix, solves the linear system, computes the norm, and then presents the function evaluation to the optimization routine. Based on this result, the optimization routine then chooses a new point at which to invoke the

function evaluation routine, and the procedure continues until the optimization routine terminates, presumably at a minimum of the objective function.

4.5 Strategies for DEM

There are three levels of work over which we can exploit parallelism. The manager process runs the DEM routine, and is responsible for obtaining diffused function values that are used by the local optimization routine inside the DEM. The diffused function values, in turn, are the weighted sums of strategically-chosen objective function values. These objective function values are independent of one another and can be evaluated in parallel. Likewise, the matrix entries needed in the computation of the objective function are independent of one another, and can also be evaluated in parallel. Figure 4.2 illustrates the three levels of parallelism that can be exploited.

4.5.1 Determining Optimal Distribution of Work

For the Diffusion Equation Method, we use a three-tier approach. The manager process divides the function evaluations required by the finite differencing scheme evenly among groups of processors. Led by a driver process, each group evaluates the function at a different point. The driver process then returns the value of the objective function to the manager, and the manager process collects the results and performs the finite differencing to obtain the value of the diffuse function.

The number of function evaluations required depends on the dimensionality of the objective function, the size of the time step, and the value of the diffusion parameter t. (For more details on determining the number of function evaluations, see Section 3.4.1.)

In a preliminary implementation, the groups and their driver processes were statically defined, based on the dimensionality of the objective function. Consequently many processors were idle in the last step of the DEM because only one function evaluation is required at time zero.

A newer, more sophisticated implementation uses the three-tier approach, but the manager process assigns the roles of driver and worker to the other processors dynamically at each evaluation of the transformed function. This approach is more flexible and allows different numbers of steps to be taken for different time values, although some processors could still be idle in the final step

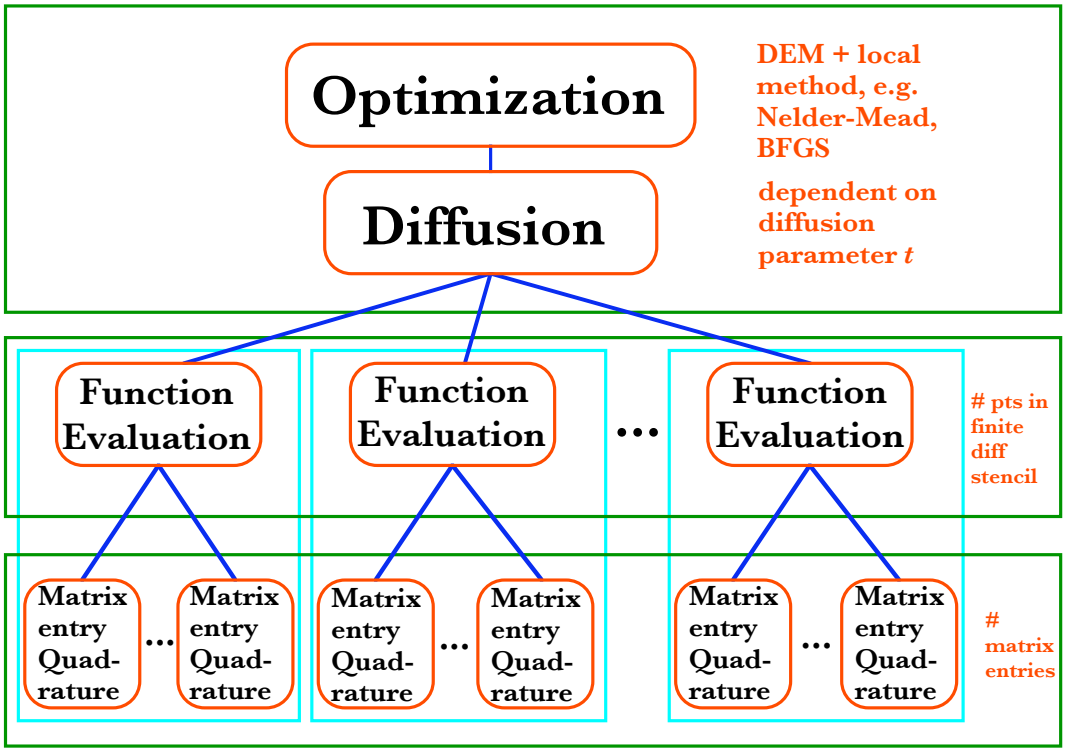


Figure 4.2. Three-tier approach to computing the diffused objective function. Green rectangles indicate three levels, and blue lines indicate sequential dependence.

N	Optimal		Heuristic		Ratio $t_{\text{optimal}}/t_{\text{heuristic}}$	
	# Drivers	Time	# Drivers	Time	Theoretical	Empirical
16	5	135	8	164	0.823	0.845
24	2	91	12	123	0.740	0.770
32	7	68	16	82	0.829	0.815
48	5	45	24	82	0.549	0.563
64	13	34	25	41	0.829	0.842
96	7	24	25	27	0.960	0.923
128	25	17	25	17	1.000	1.012
256	3	9	25	9	1.000	1.009

Table 4.1. Theoretical comparison of optimal and heuristic work distribution in the evaluation of a 25-point stencil, for N processors.

of the DEM, in the case of large sets of processors.

Optimal Distribution of Work for Two-Dimensional Objective Function

We developed two methods of determining the number of drivers to use for a given stencil size f and number of processors N. The first is a simple heuristic based on the rationale that each driver process should have at least one worker process. The second is a more complicated calculation of the optimal distribution of work, taking into account the atoms of work (*i.e.*, the number of matrix entries to compute) required for each function evaluation.

For the simple heuristic, if $f < \lfloor N/2 \rfloor$, then we set the number of drivers $D = f$. Otherwise, D is set to $\lfloor N/2 \rfloor$, so that each driver has at least one worker. The method of calculating the optimal distribution of work takes into account the number of atoms of work A, in addition to f and N. The algorithm is described in Figure 4.3.

We compared these two methods in performance evaluation runs. In the limit where the number of processors N greatly exceeds the stencil size f, both algorithms choose $D = f$. For small and intermediate values of N, the second method outperforms the simple heuristic.

We ran a set of performance evaluations for $f = 25$ and $A = 81$ number of matrix entries, varying the number of processors. We also calculated the relative timings for the optimal and heuristic configurations, and compared the ratio of the calculated relative timings to the ratio of actual timings.

f = stencil size (number of function evaluations to perform)
N = number of processors
A = number of atoms of work per function evaluation
D = optimal number of drivers (to be determined)

for $d = 1 : \min\{f, N\}$,
    $r^+ = \lceil f/d \rceil$;
    $r^- = \lfloor f/d \rfloor$;
    $g^+ = \lceil N/d \rceil$;
    $g^- = \lfloor N/d \rfloor$;
    $w^+ = \lceil A/g^+ \rceil$;
    $w^- = \lceil A/g^- \rceil$;
    $t^+ = r^+ w^+$;
    $t^- = r^- w^-$;
    if $(\mathrm{mod}\{f, d\} > \mathrm{mod}\{N, d\})$
        $t^{+-} = r^+ w^-$;
    else
        $t^{+-} = 0$;
    end if
    $t_d = \max\{t^+, t^-, t^{+-}\}$;
end for
$D = \arg\min_d\{t_d\}$;

Figure 4.3. Algorithm for determining optimal number of driver processes.

We ran three benchmarking jobs for each number of processors and configuration strategy. In the runs, one hundred evaluations of the diffused function were performed, each requiring 25 evaluations of the original objective function. Figure 4.4 compares the performance of the optimal and simple heuristic work distribution.

The straight line of its performance curve in Figure 4.4 suggests superb scalability in the case of the optimal work distribution. The net performance of the heuristic, however, suggests that it is not a particularly good strategy. The performance of the heuristic for $N = 48$ is particularly miserable, because the heuristic assigns 24 processors as drivers. The load is unbalanced, resulting in 46 processors idling as two processors perform the 25th function evaluation. In this case, the heuristic chose the worst possible configuration.

It seems that it is more important to distribute the function evaluations evenly than to distribute the worker processes evenly. One driver being short by one worker has less impact than one driver being overloaded with function evaluations. Perhaps a better heuristic would be to choose the number of drivers close to a divisor of the stencil size.

Optimal Distribution of Work for Three-Dimensional Objective Function

The cost of computing the three-dimensional objective function greatly exceeds the cost for the two-dimensional case, so we eliminate idle time by evenly distributing the work for all the objective function evaluations across all the processors. The manager process coordinates the work and collects the integrals from the processors. It then computes the objective function values and their weighted average, which is returned as the diffused objective function value.

We ran a set of performance evaluations for stencil size $f = 5$, varying the number of processors. On the CSE Turing G5 Xserve cluster, we ran three benchmarking jobs for each number of processors, and in each run, four evaluations of the diffused function were performed, each requiring five evaluations of the original objective function. Figure 4.5 shows the performance of the method on varying numbers of processors. Again the performance curve is remarkably straight and its slope indicates superlinear speedup for this method.

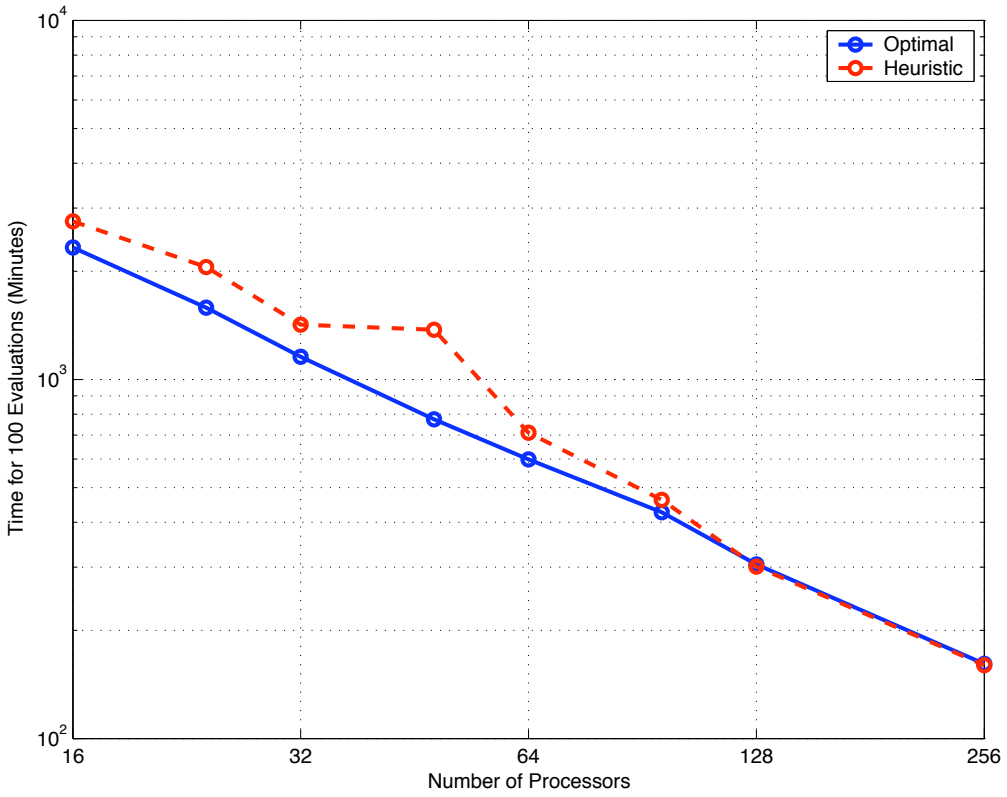


Figure 4.4. Scalability of DEM function evaluations, for $f = 25$ stencil size and 2-D objective function.

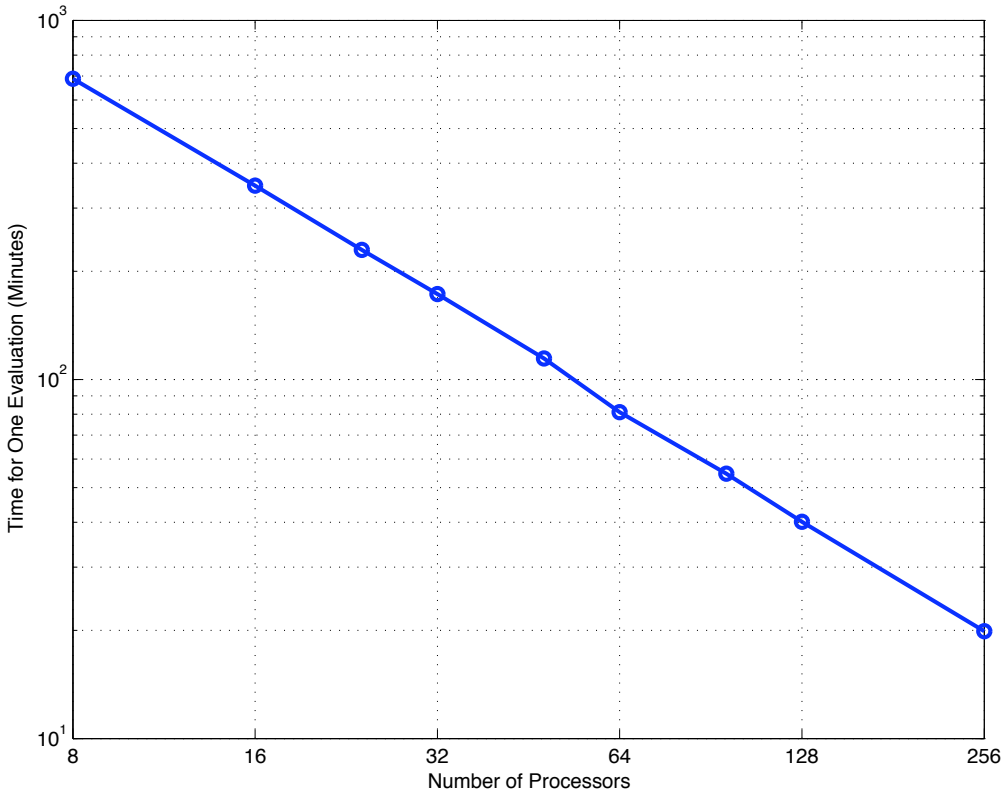


Figure 4.5. Scalability of DEM function evaluations, for $f = 5$ stencil size and 3-D objective function.

4.6 Parallel Performance Modeling

We can use parallel performance modeling to assess the potential of our implementation. There are several measures of parallel performance, including parallel execution time, speedup, and efficiency. We explore these measures of parallel performance for our objective function in two and three spatial dimensions.

For a given problem of constant size, we define T_1 as the program's execution time using a single processor, and T_N as the program's execution time using N processors in parallel. The *speedup* S_N is the ratio T_1/T_N, and the *efficiency* E_N is defined as the ratio $T_1/(NT_N)$. Ideally, $S_N = N$ and $E_N = 1$, but in most cases the speedup and efficiency do not achieve those ideal values.

The parallel execution time T_N is the sum of three components: T_{comp}, the time spent in parallel computation; T_{comm}, the time spent sending and receiving messages; and T_{idle}, the time spent idle. The computation time is application-dependent and may vary for different numbers of processors. The communication time, modeled by the cost of sending a message, is the sum of two components: t_s, the startup time, and $t_w L$, the transfer time per word times the length of the message. On most real machines, t_s is several orders of magnitude larger than t_w, so the second term can be neglected, except for very large messages.

4.6.1 Efficiency of DEM Computation of 2-D Objective Function

We used our results from the scalability analysis of the optimal configuration to compute empirical efficiency ratios. Due to wall time limits on the supercomputer, we were unable to compute T_1 for the fixed problem size we used for the scalability analysis, so we instead compute the "pseudoefficiency," defined as

$$E_{n,p} = \frac{nT_n}{pT_p},$$

for all combinations of n and p, as tabulated in Table 4.2. The pseudoefficiency between two numbers of processors n and p gives an idea of the relative efficiency of each configuration. If $E_{n,p} < 1$, then the configuration for n processors is more efficient than the configuration for p processors. For the fixed problem size, the efficiency remains relatively constant for all configurations, with a

n	p							
	16	24	32	48	64	96	128	256
16	1.0000	0.9806	1.0072	1.0036	0.9734	0.9109	0.9552	0.9023
24	1.0198	1.0000	1.0272	1.0235	0.9927	0.9289	0.9741	0.9202
32	0.9928	0.9736	1.0000	0.9965	0.9664	0.9044	0.9484	0.8959
48	0.9964	0.9770	1.0036	1.0000	0.9699	0.9076	0.9518	0.8990
64	1.0273	1.0074	1.0347	1.0311	1.0000	0.9358	0.9813	0.9270
96	1.0979	1.0765	1.1058	1.1018	1.0687	1.0000	1.0487	0.9906
128	1.0469	1.0265	1.0544	1.0507	1.0190	0.9536	1.0000	0.9446
256	1.1083	1.0867	1.1163	1.1123	1.0788	1.0095	1.0586	1.0000

Table 4.2. Pseudoefficiency ratio $E_{n,p} = (nT_n)/(pT_p)$ for 2-D objective function.

difference in pseudoefficiency of less than 12% in the most extreme case.

4.6.2 Model of DEM Computation of 2-D Objective Function

We use a three-tier approach to computing the diffused objective function. A manager process oversees driver processes, which supervise worker processes. For more details about the computation, see Sections 4.4.1 and 4.5.

The three-tier computation can be modeled as the sum of three components:

$$T_N = T_N^{\text{comp}} + T_N^{\text{comm}} + T_N^{\text{init}},$$

where T_N^{comp} and T_N^{comm} are the time spent in computation and communication, respectively, and T_N^{init} is the time spent by the manager process in serial computation and initialization.

The time spent in computation is equal to $P(N, f)$, the maximum number of matrix entries computed by one processor (in Figure 4.3, this is called t_d), times t_{quad}, the average time to compute one matrix entry. During the communication phases, $3N + d - 1$ messages are passed, where d is the number of drivers as determined by the algorithm in Figure 4.3 or by some other means, such as the simple heuristic. The time spent in communication is $(3N + d - 1)t_s$, neglecting the transfer cost, which is small compared to the startup cost. The time spent by the manager process in serial computation and initialization should be about the same for all N.

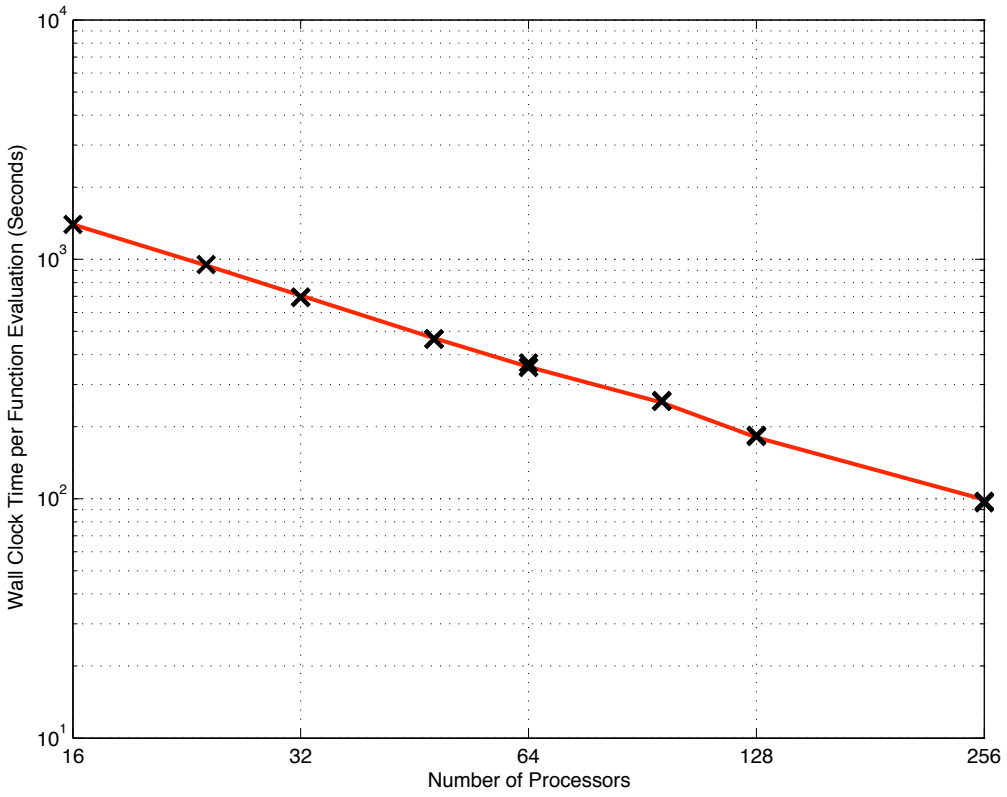


Figure 4.6. Fit of performance model and empirical data for DEM for 2-D objective function. Crosses represent timings on NCSA's tungsten supercomputer.

Thus we can rewrite our computation time as

$$T_N = P(N, f)t_{\text{quad}} + (3N + d - 1)t_s + t_{\text{init}}.$$

We performed a least squares fit on our results from the scalability analysis and determined that $t_{\text{quad}} = 10.3311$ seconds, $t_s = 3.81077 \times 10^{-3}$ seconds, and $t_{\text{init}} = 3.19500$ seconds for the optimal work distribution of the 2-D integrand on NCSA's tungsten supercomputer. Figure 4.6 shows the excellent fit of the model and the performance of the algorithm on NCSA's tungsten.

n	p								
	8	16	24	32	48	64	96	128	256
8	1.0000	0.9921	0.9984	0.9943	1.0019	1.0609	1.0484	1.0703	1.0804
16	1.0079	1.0000	1.0064	1.0022	1.0099	1.0693	1.0567	1.0788	1.0890
24	1.0016	0.9937	1.0000	0.9958	1.0035	1.0625	1.0500	1.0720	1.0821
32	1.0057	0.9978	1.0042	1.0000	1.0077	1.0669	1.0544	1.0765	1.0866
48	0.9981	0.9902	0.9965	0.9924	1.0000	1.0588	1.0463	1.0682	1.0783
64	0.9426	0.9352	0.9412	0.9373	0.9444	1.0000	0.9882	1.0089	1.0184
96	0.9539	0.9464	0.9524	0.9484	0.9557	1.0119	1.0000	1.0209	1.0306
128	0.9343	0.9270	0.9329	0.9290	0.9361	0.9912	0.9795	1.0000	1.0094
256	0.9256	0.9183	0.9242	0.9203	0.9274	0.9819	0.9703	0.9907	1.0000

Table 4.3. Pseudoefficiency ratio $E_{n,p} = (nT_n)/(pT_p)$ for 3-D objective function.

4.6.3 Efficiency of DEM Computation of 3-D Objective Function

We computed empirical efficiency ratios using the results from the scalability analysis of the DEM using the three-dimensional objective function. As in the case of the two-dimensional objective function, we were unable to compute T_1 for the fixed problem size due to wall time limits, so we instead compute the pseudoefficiency

$$E_{n,p} = \frac{nT_n}{pT_p}$$

for all combinations of n and p, as tabulated in Table 4.3. Interestingly, with this method we obtained superlinear speedup as the number of processors increased. For example, the pseudoefficiency $E_{n,8}$ exhibits a downward trend as n grows, meaning that the 256-processor configuration is the most efficient.

Superlinear speedup is usually attributed to advantageous caching. As the amount of work decreases, a larger fraction of the data can be held in faster memory, and less memory swapping is necessary. We tested to see if this caused the superlinear speedup in this case, by increasing the amount of work with the number of processors, thus keeping constant the amount of work to be performed per processor. If the superlinear speedup is a result of the caching, then holding the amount of work per processor constant should result in a constant run time for all numbers of processors. It turns out that holding the amount of work constant does result in a constant run time across all numbers of processors. For more details, see Section 4.6.4.

4.6.4 Model of DEM Computation of 3-D Objective Function

The computation of the diffused objective function is performed with a manager-worker approach. For more details about the computation, see Sections 4.4.2 and 4.5.

This two-level computation can be modeled as the sum of three components:

$$T_N = T_N^{\text{comp}} + T_N^{\text{comm}} + T_N^{\text{init}},$$

where T_N^{comp} and T_N^{comm} are the time spent in computation and communication, respectively, and T_N^{init} is the time spent by the manager process in serial computation and initialization.

The time spent in computation is inversely proportional to the number of processors N, and directly proportional to the stencil size and the number of matrix entries to be computed. The communication requirements are modest: one broadcast and one gather. The message to be broadcast is simply the parameters, so cost per word can be neglected for the broadcast, but for the gather, the size of the message is large to begin with, and grows with the number of processors. The number of matrix entries to be computed, A, is 1890, and the number of iterations of the VEGAS algorithm, v, is five. So we model the time spent in communication as $2\lg(N)t_s + AvN\lg(N)t_w = 2\lg(N)t_s + 1890 \times 5 \times N\lg(N)t_w$, where t_s is the startup time, and t_w is the transfer cost per double. Both the broadcast and gather operations use a tree structure to complete the communication, so the cost should be logarithmic with respect to the number of processors. The time spent by the manager process in serial computation and initialization should be about the same for all N.

Thus we can express the computation time as

$$T_N = \frac{A}{N}t_{\text{quad}} + 2\lg(N)t_s + AvN\lg(N)t_w + t_{\text{init}}.$$

We performed a non-negative least squares fit on our results from the scalability analysis and determined that $t_{\text{quad}} = 139.83$ seconds, and t_s, t_w, and t_{init} all equal to zero on CSE's Turing supercomputer, indicating that the corresponding terms in the performance model are relatively insignificant. Figure 4.7 shows the fit of the model and the performance of the algorithm on CSE's

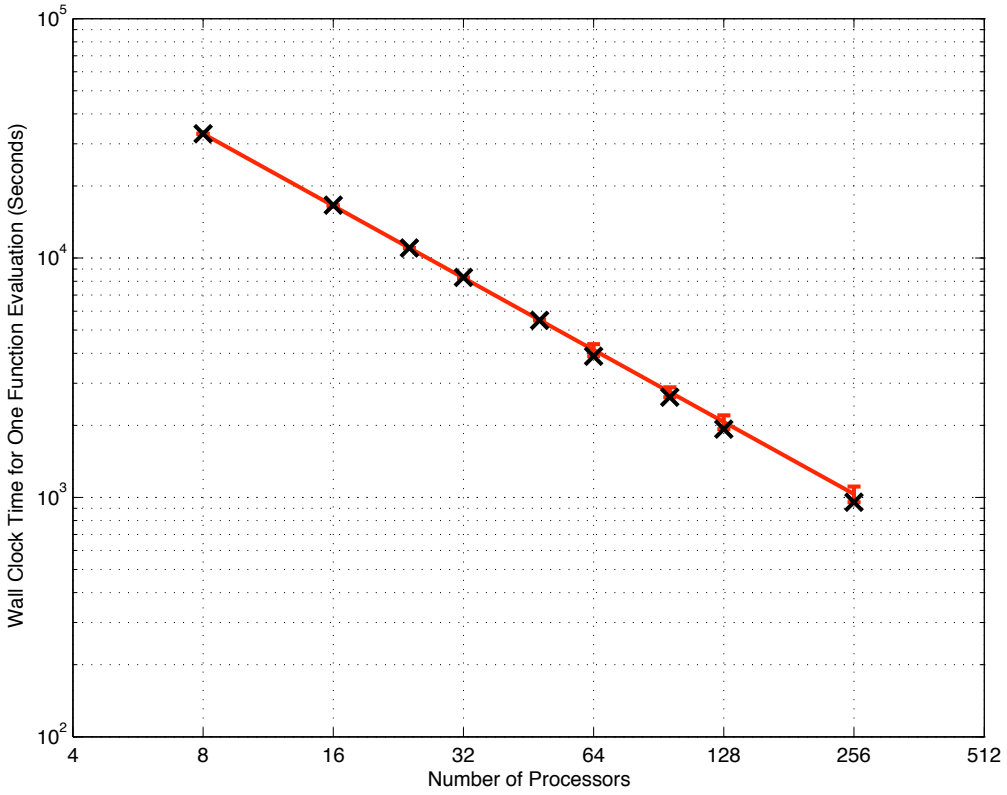


Figure 4.7. Fit of performance model and empirical data for DEM for 3-D objective function. Crosses represent timings on CSE's Turing supercomputer.

Turing. Due to the superlinear speedup of the algorithm, the model becomes increasingly inaccurate for large N, as indicated by the growing error bars and increasing distance from the line to the crosses.

We also performed a non-negative least squares fit on our results from the scalability analysis in which the amount of work per processor was kept constant. In this case, we can write the time for N processors as

$$T_N = At_{\text{quad}} + 2\lg(N)t_s + AvN\lg(N)t_w + t_{\text{init}}.$$

Because t_{quad} and t_{init} are multiplied by constant coefficients, we are unable to fit them separately, but we end up with $t_{\text{quad}} + t_{\text{init}}/A = 174.65$ seconds, $t_s = 1.20$ seconds, and $t_w = 0.00$

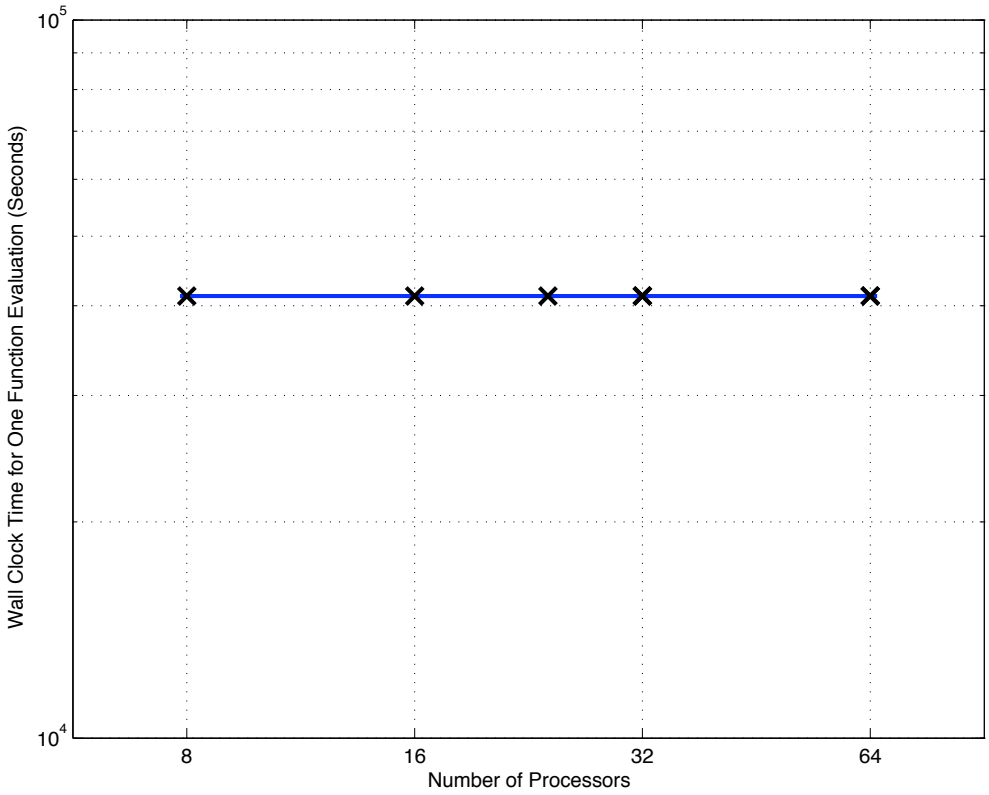


Figure 4.8. Fit of performance model and empirical timings of DEM with amount of work per processor kept constant for 3-D objective function. Crosses represent timings on CSE's Turing supercomputer.

seconds. Figures 4.8 and 4.9 show the fit of the model and the performance of the algorithm on CSE's Turing supercomputer. In Figure 4.8, we observe that the performance curve is a horizontal line, meaning that the superlinear speedup for the fixed problem size is due to caching. Figure 4.9 shows the performance curve in more detail, including the error bars from the fit and the variations in timings.

Assuming that t_{init}/A is negligible, we can compute the relative difference between t_{quad} computed from the runs in which the total amount of work remained constant and t_{quad} computed from the runs in which the amount of work per processor remained constant. The relative difference is approximately 22%, suggesting that our model is reasonably realistic.

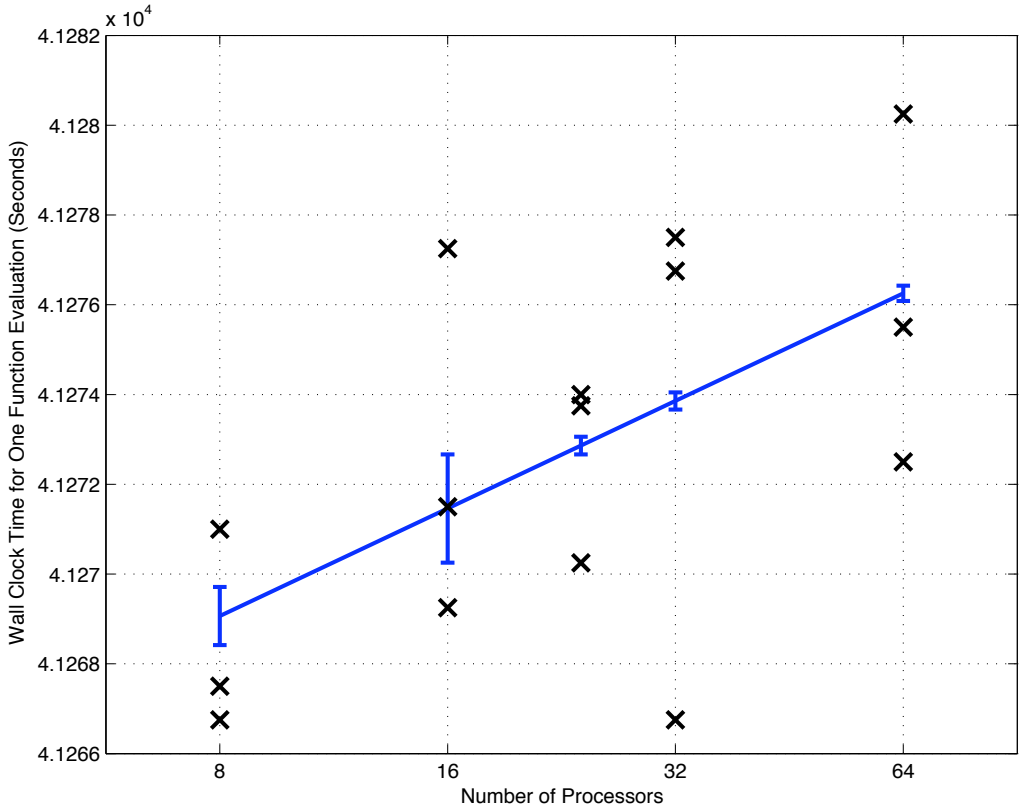


Figure 4.9. Fit of performance model and empirical timings of DEM with amount of work per processor kept constant for 3-D objective function. Crosses represent timings on CSE's Turing supercomputer.

$t_{\max}$	# Iterations DEM	# Evaluations	Convergence
0.02	6	631	(0.998971, -0.004323)
	4	181	(0.999932, -0.000133352)
	3	333	(1.175560, 0.405093)
0.015	3	336	(0.561734, 6.513328)
0	1	89	(1.327086, 0.528704)

Table 4.4. Performance of DEM on geoprospecting objective function. Number of evaluations of diffused objective function and minimizer found for various settings. Low number of evaluations for $t_{\max} = 0.02$ and four iterations is because Nelder–Mead used instead of BFGS.

4.7 Performance on Magnetotelluric Test Problem

We have shown that the computation of the original and diffused objective functions can be performed in parallel and exhibit a high degree of scalability. While it is good to know that the objective function is easily computed, it is also important to test the efficacy of the global optimization method. In this section, we illustrate its the success of this method with a case study of the two-dimensional magnetotelluric objective function.

We performed the analysis with varying the function in only the first two variables, a and e^2, and setting all the rest, x_0, y_0, and θ, to zero. We set the true solution to $z^* = (a, e^2) = (1.0, 0.0)$. If we use BFGS, a local optimization method, starting from $z_0 = (1.4, 0.4)$, we converge to a local minimizer, $z = (1.327, 0.529)$. But when we use six steps of diffusion with $t_{\max} = 0.02$, with the same local optimization method in the inner loop, we converge to the correct solution. Figure 4.10 illustrates the convergence of the method to the true solution.

If we do not take enough continuation steps, however, the global minimum may not be found. Taking four iterations of the DEM with the same $t_{\max} = 0.02$ was sufficient to converge to the global minimum, but reducing it to three iterations was not. Likewise, if the function is not sufficiently diffused, the method will not converge to the global minimum. If we take $t_{\max} = 0.015$, the function is insufficiently diffused and the global minimizer is not found. These results are summarized in Table 4.4.

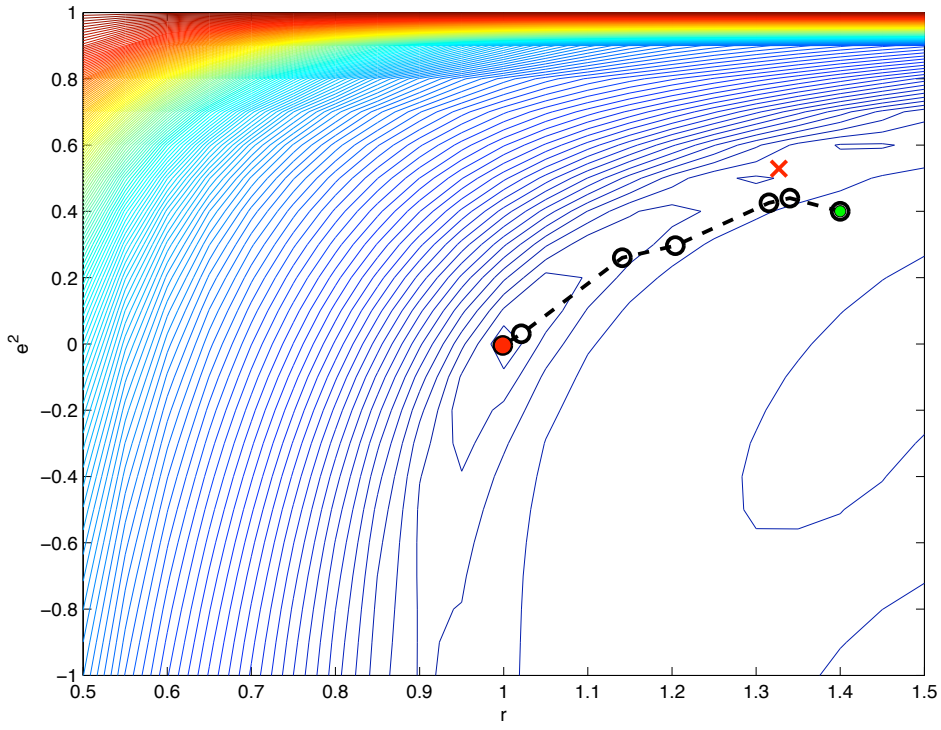


Figure 4.10. Solution of magnetotelluric objective function using DEM. Starting from $(1.4, 0.4)$, BFGS converges to local minimizer (red cross). DEM (with BFGS as local optimization routine) traces path along dashed line, with circles representing result of each continuation step, converging to the global minimizer, $(1, 0)$.

References

[1] Richard P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[2] I. Chlodovsky. Sur le développement des fonctions définies dans un intervalle infini en séries de polynomes de M. S. Bernstein. *Compositio Mathematica*, 4:380–393, 1937.

[3] Ole Christensen and Khadija L. Christensen. *Approximation Theory: from Taylor Polynomials to Wavelets*. Birkhäuser, Boston, 2004.

[4] Louis Comtet. *Advanced Combinatorics*. Reidel, Boston, 1974.

[5] Burton S. Garbow, Kenneth E. Hillstrom, and Jorge J. Moré. LMDIF from MINPACK. `http://www.netlib.org/minpack/`.

[6] Perry Gray, William Hart, et al. A survey of global optimization methods. `http://www.cs.sandia.gov/opt/survey/main.html`, 1997.

[7] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[8] Michael T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, New York, 2nd edition, 2002.

[9] Reiner Horst and Hoang Tuy. *Global Optimization: Deterministic Approaches*. Springer-Verlag, Berlin, 2nd edition, 1993.

[10] C. Impens. Asymptotics of Bernstein polynomials. G. Alexits Memorial Conference, Budapest, 1999. (Slides obtained from author).

[11] C. Impens and H. Vernaeve. Asymptotics of differentiated Bernstein polynomials. *Constructive Approximation*, 17:47–57, 2001.

[12] C. T. Kelley. Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10(1):43–55, 1999.

[13] Arnold R. Krommer and Christoph W. Ueberhuber. *Computational Integration*. Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[14] G. P. Lepage. VEGAS: An adaptive multi-dimensional integration program. Cornell preprint CLNS 80-447, 1980.

[15] G. Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27:192–203, 1978.

[16] G. G. Lorentz. *Bernstein Polynomials*. Chelsea, New York, 2nd edition, 1986.

[17] A. Marcuello-Pascual, P. Kaikkonen, and J. Pous. 2-D inversion of MT data with a variable model geometry. *Geophys. J. Int.*, 110:297–304, 1992.

[18] K. I. M. McKinnon. Convergence of the Nelder–Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158, 1998.

[19] Jorge Moré and Zhijun Wu. Smoothing techniques for macromolecular global optimization. Technical report, Argonne National Laboratory, 1995.

[20] Jorge J. Moré and Zhijun Wu. Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7(3):814–836, 1997.

[21] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comp. J.*, 7:308–313, 1965.

[22] Arnold Neumaier. Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, 39(3):407–460, 1997.

[23] Gregory A. Newman. Crosswell electromagnetic inversion using integral and differential equations. *Geophysics*, 60(3):899–911, 1995.

[24] Gregory A. Newman and David L. Alumbaugh. Three-dimensional magnetotelluric inversion using non-linear conjugate gradients. *Geophysics Journal International*, 140:410–424, 2000.

[25] Gregory A. Newman and G. Michael Hoversten. Solution strategies for two- and three-dimensional electromagnetic inverse problems. *Inverse Problems*, 16:1357–1375, 2000.

[26] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[27] George M. Phillips. *Interpolation and Approximation by Polynomials*. Springer-Verlag, New York, 2003.

[28] Lucjan Piela, Jaroslaw Kostrowicki, and Harold A. Scheraga. The multiple-minima problem in the conformational analysis of molecules. Deformation of the potential energy hypersurface by the diffusion equation method. *Journal of Physical Chemistry*, 93:3339–3346, 1989.

[29] Oleg Portniaguine and Michael S. Zhdanov. Focusing geophysical inversion images. *Geophysics*, 64(3):874–887, 1999.

[30] Christopher Siefert, Virginia Torczon, and Michael W. Trosset. Model-assisted pattern search. `http://www.cs.wm.edu/~va/software/maps/`.

[31] N. J. A. Sloane. The on-line encyclopedia of integer sequences. `http://www.research.att.com/~njas/sequences/`.

[32] Torquil Smith et al. Sharp boundary inversion of 2D magnetotelluric data. *Geophysical Prospecting*, 47:469–486, 1999.

[33] A. N. Tikhonov and V. Arsenin. *Solutions of Ill-Posed Problems*. John Wiley and Sons, Washington, D.C., 1977.

[34] A. N. Tikhonov and V. B. Glasko. Application of the regularization method to geophysical interpretation problems. *Izvestiya, Academy of Science, USSR. Physics of the Solid Earth*, 11(1):25–32, 1975.

[35] Aimo Törn and Antanas Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1989.

[36] M. S. Zhdanov and N. G. Golubev. Use of the finite functions method for the solution of the 2D inverse problem. *J. Geomag. Geoelectr.*, 35:707–721, 1983.

[37] A. Zygmund. *Trigonometric Series*. Cambridge University Press, Cambridge, 1988.

Author's Biography

Rebecca Hartman–Baker was born in Lexington, Kentucky in 1975. She received a B.S. in Physics from the University of Kentucky in 1998, and joined the graduate program in Computer Science at the University of Illinois at Urbana–Champaign in 1998. After completing her degree, she joined Oak Ridge National Laboratory in Oak Ridge, Tennessee as a postdoctoral research associate. In addition to numerical analysis and supercomputing, Rebecca enjoys karate, music, and sewing.