

Abstract

BODILY, CHRIS H. Numerical Differentiation Using Statistical Design. (Under the direction of John F. Monahan.)

Derivatives are frequently required by numerical procedures across many disciplines. Numerical differentiation can be useful for approximating derivatives. This dissertation will introduce computational differentiation (the process by which derivatives are obtained with a computer), focusing on statistical response surface (RSM) designs for approximating derivatives. The RSM designs are compared with two competing numerical methods: a rival saturated statistical design approach, and a method employing finite differencing. A covariance model incorporating function curvature and computer round-off error is proposed for estimating the derivative approximation variances. These variances and the computational workload each method requires become the basis for comparing the derivative approximations. A diagnostic test for variable scaling errors is also described.

Numerical Differentiation Using Statistical Design

by

Chris H Bodily

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

STATISTICS

in the

GRADUATE SCHOOL

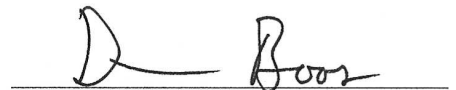
at

NC STATE UNIVERSITY

2002



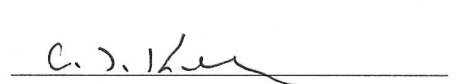
Dr. J. Monahan, Chair



Dr. D. Boos



Dr. F. Giesbrecht



Dr. T. Kelley

Numerical Differentiation Using Statistical Design

Copyright 2002

by

Chris H Bodily

To my family and all who have a hand in my success:

*Oh say, what is truth?
'Tis the fairest gem
That the riches of worlds can produce,
And priceless the value of truth will be when
The proud monarch's costliest diadem
Is counted but dross and refuse.*

*Yes, say, what is truth?
'Tis the brightest prize
To which mortals or Gods can aspire.
Go search in the depths where it glittering lies,
Or ascend in pursuit to the loftiest skies:
'Tis an aim for the noblest desire.*

*The sceptre may fall from the despot's grasp
When with winds of stern justice he copes.
But the pillar of truth will endure to the last,
And its firm-rooted bulwarks outstand the rude blast
And the wreck of the fell tyrant's hopes.*

*Then say, what is truth?
'Tis the last and the first,
For the limits of time it steps o'er.
Tho the heavens depart and the earth's fountains burst,
Truth, the sum of existence, will weather the worst,
Eternal, unchanged, evermore.*

John Jaques (1827-1900)

Biography

Chris H Bodily was born in Ogden, Utah, to Gary and Carroll Bodily in June, 1970. He grew up in Syracuse, Utah, and graduated from Clearfield High School in June, 1988. After one year of undergraduate study at Brigham Young University, he left the United States in November, 1989, for nearly two years to serve a full-time volunteer mission for the Church of Jesus Christ of Latter-day Saints in Taiwan, R.O.C. Returning in July, 1991, he resumed his studies at Brigham Young University. He received his B.S. in Statistics in April, 1994, and his M.S. in Statistics in August, 1996, from BYU. In February, 1996, he took an internship with the Quality, Management, and Technology Center at E.I. Du Pont de Nemours and Company in Wilmington, Delaware, and afterwards entered the doctoral program in the Department of Statistics at North Carolina State University in August, 1996. While a Ph.D. student, he also worked as a graduate research assistant for Glaxo Wellcome Inc., and in July, 2000, took employment with Becton Dickinson Technologies, in Research Triangle Park, NC.

While a student at NC State he and his wife welcomed two children into their family and are currently expecting their third. Chris successfully defended his dissertation on June, 21, 2002, and is currently a Senior Statistician at Becton Dickinson Technologies.

Acknowledgements

I want to thank my advisor, John Monahan, for his tremendous patience and support. I enjoyed our many conversations while working through this research. I especially admire his intuition and enthusiasm. I also wish to thank my committee and my study groups without whom I would have never graduated.

Certainly, I owe a debt of gratitude to my wife and family for enduring years of seemingly endless schooling with me. Their support continues to be essential to my success. Finally, I have a deep belief in a Father in Heaven whose guidance and encouragement I feel constantly in my life. I am grateful for His love and attention.

Contents

List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Notation	3
2 The Literature and Current Use of Numerical Differentiation	4
3 An Example	10
4 Differencing	14
4.1 The First Derivative in One Dimension and the Forward Difference	14
4.2 Three-point formulas and the Central Difference	18
4.3 The Second Derivative in One Dimension	20
4.4 Derivatives in Two Dimensions	21
4.5 How small should h be?	22
5 Competing methods	25
5.1 The Spendly Method	25
5.2 Del12F	28
5.3 A Statistical Perspective	30
5.3.1 Response Functions and Differentiation	31
5.3.2 Estimating the Approximate Response Function	32
5.3.3 RSM Design Considerations	33
5.3.4 Constructing the RSM Designs	35
5.3.5 Statistical Issues	37
6 A Model for Evaluating Derivative Approximations	39
6.1 Comparing Methods	40
6.2 An Overview of Random Functions	43
6.3 A Useful Covariance Function	44
6.4 Generating Random Functions	45

6.5	Derivative Approximations as Linear Combinations	51
6.6	The Covariance Between a Random Function and its Derivatives	53
6.6.1	A Sidenote on Design Scale	55
6.7	Computing Derivative Approximation Variances	56
6.8	Derivative Comparison Results – Which Approximation is Better?	62
6.8.1	Recommendations	62
6.9	Diagnostics	69
6.9.1	Forecast Variances	69
6.9.2	Diagnostic Simulations	72
7	Conclusions and Future Work	82
7.1	Future Work	83
	Bibliography	85
A	Covariance Function Derivatives	88
A.1	Covariances between the function and its first derivative	88
A.2	Covariances between the function and its second derivative	89
A.3	Covariances between the function and its mixed derivative	89
B	RSM Design Tables	90

List of Figures

3.1	Del12F function evaluation locations	11
5.1	Spendly simplex design grid and function evaluation locations	26
5.2	Spendly Hessian matrix function evaluation locations and coefficients	27
5.3	Del12F mixed derivative approximation function evaluation locations and coefficients	29
5.4	Inner and Outer Response Surface Designs in two dimensions	37
6.1	Plot of Three Sample Random Functions.	41
6.2	Plot of 5 Random Functions with curvature parameter $a = 0.7$	48
6.3	Plot of 5 Random Functions with curvature parameter $a = 1.0$	49
6.4	Plot of 5 Random Functions with curvature parameter $a = 1.3$	49
6.5	Scatter Plot of R-Square by Shape Parameter a for a One-Dimensional Random Function	50
6.6	Plot of Root Average Derivative Approximation Variance by Problem Dimension for First Derivatives in Double Precision.	63
6.7	Plot of Root Average Derivative Approximation Variance by Problem Dimension for First Derivatives in Single Precision.	64
6.8	Plot of Root Average Derivative Approximation Variance by Problem Dimension for Second Derivatives in Double Precision.	65
6.9	Plot of Root Average Derivative Approximation Variance by Problem Dimension for Second Derivatives in Single Precision.	66
6.10	Plot of Root Average Derivative Approximation Variance by Problem Dimension for Mixed Derivatives in Double Precision.	67
6.11	Plot of Root Average Derivative Approximation Variance by Problem Dimension for Mixed Derivatives in Single Precision.	68
6.12	Plot of Diagnostic Test Level for Scaling Errors Using Single Precision – No Scaling Errors.	74
6.13	Plot of Diagnostic Test Power for Scaling Errors Using Single Precision – One Dimension Scaled Badly.	75
6.14	Plot of Diagnostic Test Power for Scaling Errors Using Single Precision – Highest Dimension Scaled Badly.	76

6.15 Plot of Diagnostic Test Power for Scaling Errors Using Single Precision – Two Dimensions Slightly Off-Scale.	77
6.16 Plot of Diagnostic Test Level for Scaling Errors Using Double Precision – No Scaling Errors.	78
6.17 Plot of Diagnostic Test Power for Scaling Errors Using Double Precision – One Dimension Scaled Badly.	79
6.18 Plot of Diagnostic Test Power for Scaling Errors Using Double Precision – Highest Dimension Scaled Badly.	80
6.19 Plot of Diagnostic Test Power for Scaling Errors Using Double Precision – Two Dimensions Slightly Off-Scale.	81

List of Tables

5.1	Number of Experiments Required for Response Surface Designs	34
6.1	Table of Method Variances and Optimum Step Sizes (h)	46
6.2	Table of Simulated Scaling Patterns	72

Chapter 1

Introduction

Derivatives are frequently required by numerical procedures across many disciplines. Many iterative nonlinear equation solvers, optimization algorithms, and integration algorithms for differential equations require derivative information about the functions upon which they work. Closed form expressions for these derivatives may be unavailable for a number of reasons. Perhaps they are too expensive, too difficult, or even impossible to calculate. In these cases, numerical differentiation can be useful for approximating the needed derivatives.

A good example is nonlinear regression, where the partial derivatives of the nonlinear response function are needed in the Gauss-Newton or Levenberg-Marquardt algorithms. If the derivatives are not provided they can be approximated using numerical differentiation. Thus a solution to the nonlinear problem is found without first working through the analytic derivative calculations—a tangential problem which can be unwieldy at times. On the other hand, one may question settling for approximate results when the analytic derivative

calculations are exact. Another potential benefit of numerical differentiation comes from the fact that analytic derivatives are only useful if they are calculated correctly. Many with experience in working nonlinear problems can attest that a common cause of non-convergence to a solution is misspecification (via miscalculation) of the derivatives. In this situation, a quality numerical differentiation algorithm can either approximate or permit checking the accuracy of the analytic derivatives. In either case, the derivative approximation can provide reliable results, requiring only that the nonlinear model is correctly specified.

The topic of this thesis involves computational differentiation or the process by which derivatives are obtained with a computer. More specifically we are interested in using statistical response surface designs to approximate the derivatives as an alternative (or perhaps an enhancement) to finite differencing. In general, a numerical derivative is the fruit of a particular computational algorithm that is used to approximate a derivative. Several methods of computational differentiation exist and each has advantages and drawbacks. The sections that follow provide a brief overview of the literature and the different approaches to computational differentiation. A convenient notation is presented to aid in distinguishing derivative approximations arising from the various methods, and a brief introduction to polynomial interpolation precedes a discussion of finite differencing. Two specific approaches are introduced, which will serve as benchmarks for our proposed method employing statistical response surface designs. The final sections discuss this new approach with groundwork explaining how regression models may be useful for computing derivatives and provide information on the designs used.

1.1 Notation

Several methods for approximating the derivatives of a function f will be described in the following sections. Exact derivative calculations will be indicated by f' and f'' for the first and second derivatives of f , respectively. If the derivative is calculated with respect to a particular variable, the variable or variable number will be included as a subscript. A second (or repeated) subscript indicates a mixed (or second) derivative. For example, the exact second derivative of f with respect to x_i is labelled as f''_{ii} . When indicating an approximation to the derivative, such as a forward difference, a superscript label will replace the apostrophe(s) and the order of the derivative will be indicated by the number of subscripts. For example, the first derivative of f with respect to x as approximated by a forward difference is labelled as f_x^{fwd} and the mixed derivative of f with respect to x_i and x_j as approximated by a central difference is labelled as f_{ij}^{cen} .

Chapter 2

The Literature and Current Use of Numerical Differentiation

From calculus, the derivative of the function f at x_0 is defined as

$$f'(x_0) = \lim_{h \rightarrow 0} \frac{f(x_0 + h) - f(x_0)}{h}. \quad (2.1)$$

Examining this definition it is clear why derivatives are referred to as slopes, or “the rise of a function divided by the run.” The above equation motivates a simple approximation for the first derivative, called a forward difference:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h},$$

for some small positive h .

It is interesting that this derivative can be estimated—without knowing the form of f —if only function evaluations and abscissae of the function are available. That is, one can take the difference in two function evaluations, $f(x_0 + h) - f(x_0)$ (the rise), and scale

the difference by $(x_0 + h) - x_0$ (the run) to arrive at an estimate for the derivative of f at x_0 . In thinking about the quality of this estimate it is important to consider how far the function “runs” between the two evaluations, remembering that the definition above involved a limit where the “run” distance approaches zero.

The forward difference illustrated above is one of a group of derivative approximations, called finite differences. In fact, finite difference approximations exist for many derivatives commonly of interest, including second order and mixed partial derivatives. These approximations, calculated simply as linear combinations of function evaluations, are very convenient. Furthermore, they regard the function being differentiated as a mere black box. However, while they are fairly easy to understand and compute, it is important to note that these formulas are approximations only. They are subject to approximation truncation and machine roundoff errors, and we have the additional problem of selecting a proper “run” distance. Therefore, their accuracy is very important to consider whenever they are used.

As mentioned earlier, derivatives are useful in optimization algorithms. Offering an alternative to finite differencing, Conn et al. (1997) discuss progress in unconstrained optimization *without* derivatives. This reference is included because, like finite differencing, their methods use only function evaluations without knowledge of the form of the function itself. Additionally, they consider situations where computing f may be very expensive and the values of the derivatives of f may not be available either because f results from some physical, chemical, or econometric measurements, or from a very large, complex computer simulation for which the source code is effectively unavailable. That is, f may not be the

result of a callable program. Conn et al. provide an introduction to derivative-free methods of optimization, methods which do not attempt to directly compute approximations to the unavailable derivative information. These algorithms seek a “trust region” around a point of interest in which they create a quadratic model of the objective function f that is cheaper to evaluate. However, the model is not constructed from first and second derivatives. Instead, the model is built so that it interpolates function values at past points. Two specific approaches using Lagrange fundamental polynomials and Newton fundamental polynomials are introduced and examined. The Lagrange polynomials are potentially expensive since they must be reconstructed for every new interpolation. In contrast, the Newton polynomials can be updated. Additionally, Conn et al. stress the importance of keeping the geometry of the problem in mind at all times to insure a proper model is being used.

Tolsma and Barton (1998) also acknowledge that the cost associated with evaluation of numerical derivatives may be significant. They review several ways in which numerical derivatives can be computed including hand-coding of analytical derivatives, finite differences, reverse polish notation (RPN) evaluation, symbolic differentiation, and automatic differentiation. They list three prerequisites for a good differentiation method as given by Iri (1991), namely that it should be fast with a low complexity, it should be free from truncation error, and it should be able to be performed automatically. The previously mentioned methods are evaluated with regard to these prerequisites.

Symbolic derivative calculations employ a data tree structure to describe f . The algorithm then traverses the tree applying derivative rules to the tree branches arriving at a

new function that is (hopefully) the analytic derivative of f . The advantage of this method is that the results are exact. However, depending on the form of f the results are often much more complex; so complex that Tolsma and Barton estimate the cost of computing a gradient as roughly n times the cost of evaluating f , where n is the number of variables in f . This cost may be prohibitively expensive in some situations.

The RPN approach is an alternative to the tree notation that it is more compact but still preserves most of the information about f that is present in the tree notation. The elements of f are stored in a function stack (a first-in, last-out data structure) and processed one element at a time. Making use of a second stack for derivatives, the RPN method processes the elements of f from the function stack until a single element is left. If n partial derivatives of a vector-valued function are needed, then $n + 1$ stacks are required. Like symbolic derivatives, those produced by the RPN method are exact, requiring a similar amount of computer processing time and resources.

Finally, automatic differentiation methods represent f as a composite of so-called elementary functions which are enumerated in a tree. However, unlike the symbolic derivative tree, redundant branches in the tree form of f are removed. Elementary partial derivatives are computed using standard rules of differentiation like the product and quotient rules. Having computed elementary derivatives, the final result must be assembled through a process called accumulation. Each of these derivative calculation methods is evaluated with respect to the three criteria given by Iri. According to Tolsma and Barton only automatic differentiation meets all of the criteria. They refer to Griewank (1989) for more details.

Wexler (1988) offers an alternative computational derivative method also using data trees. In this method, values of the partial derivative(s) are calculated while retaining some information about the form of the function without any symbolic manipulation. All desired higher order derivatives may be calculated exactly providing they exist.

For specific implementations of the above-mentioned algorithms, a quick survey of data analysis software will turn up several examples. A few examples include Maple, Mathematica, MatLab, and MathCad. These mathematics programs serve many purposes, only one of which is symbolic manipulation (including differentiation) of functions.

Perhaps the most prevalent use of computational differentiation arises in finding function extrema (optimization) and in solving differential equations. The NAG and IMSL numerical software libraries make use of subroutines to accomplish both of these tasks. Regarding optimization in particular, these libraries supply subroutines to calculate both forward and central differences. Furthermore, in their optimization subroutines, numerical differentiation routines are called in an effort to verify that user-supplied analytic derivatives (in the event they are provided) are correctly calculated and successfully coded.

Statistical software packages also make wide use of computational differentiation. SAS (SAS 2001) uses an extensive symbolic differentiation system that generates exact derivative code. This system was used in the new NLMIXED procedure for nonlinear mixed models. However, in the older MIXED procedure for linear mixed models, the first and second derivatives of all the covariance structures were hand-coded directly. In general, the CMP procedure is available in SAS to symbolically differentiate functions that can be coded directly in the SAS language. Otherwise, SAS employs results from Dennis and

Schnabel (1983) and Abramowitz and Stegun (1972) to numerically differentiate functions using forward or central differences based on calls to the function or to its gradient. SAS also uses central difference equations to calculate second order partial derivatives.

Chapter 3

An Example

We are interested in using statistical modelling to approximate the derivative of f . The Del12F algorithm (Monahan 2001) uses finite differencing to approximate first and second order as well as mixed derivatives. Gathering function evaluations at certain locations around and including the point at which the derivative approximation is desired, Del12F collects data from what can also be viewed as an experimental design layout. Figure 3.1 illustrates this design with $\mathbf{x} = (x_i, x_j)^\top$ defined as a vector of two elements, y_k indicating evaluations at the various locations, h_i and h_j indicating step sizes in each direction, and $\mathbf{e}_i$ and $\mathbf{e}_j$ representing elementary vectors.

The differencing formulas employed by Del12F use linear combinations of the function evaluations to yield its derivative approximations. Viewed another way, Del12F uses differencing to analyze a dataset and obtain derivative approximations. This same dataset can be analyzed using least squares by first fitting a quadratic model and then using the model coefficients to approximate the derivatives. A property of the least squares solution

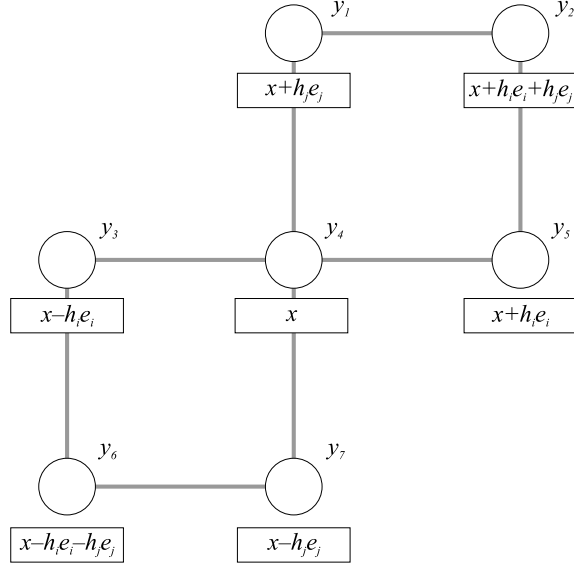


Figure 3.1: Del12F function evaluation locations

is that the coefficients are also linear functions of the data. That is, the formulas used to calculate the model coefficients in the linear model may also suggest alternatives to the differencing formulas.

To illustrate the comparison we consider the derivative of f with respect to x_i , $\partial f / \partial x_i$. To approximate this derivative, Del12F requests function evaluations at y_3 and y_5 and performs the following calculation:

$$f_i^{\text{D12F}} = \frac{y_5 - y_3}{2h}.$$

The least squares approximation for this derivative is $\hat{\beta}_i$, the estimated coefficient for x_i in the model. Since $\hat{\beta}_i$ is a linear function of the y_k we can compare its form to the differencing formula given above. The least squares solution gives:

$$f_i^{\text{D12F-ls}} = \hat{\beta}_i = \frac{1}{3} \left[\frac{-y_1 + y_2 - 2y_3 + 2y_5 - y_6 + y_7}{2h} \right].$$

Now that two estimators of the derivative are available a natural question arises of which is better. A common statistical approach is to make distributional assumptions on $\mathbf{y}$, the vector containing the function evaluations. These assumptions create a setting for defining what is meant by better and allows a comparison of the estimators. We define a design matrix $\mathbf{X}$ as

$$\mathbf{X} = \begin{bmatrix} 1 & x_{i1} & x_{j1} & x_{i1}^2 & x_{j1}^2 & x_{i1}x_{j1} \\ 1 & x_{i2} & x_{j2} & x_{i2}^2 & x_{j2}^2 & x_{i2}x_{j2} \\ & & & \vdots & & \\ 1 & x_{i7} & x_{j7} & x_{i7}^2 & x_{j7}^2 & x_{i7}x_{j7} \end{bmatrix},$$

where x_{ik} corresponds to the value of x_i when the k^{th} y -observation was taken. Finally, we assume that $\mathbf{y}$ follows a normal distribution, $\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I})$. Now we can compare the estimators based on bias and variability.

The Del12F estimator is unbiased since

$$E\left\{f_i^{\text{D12F}}\right\} = E\left\{\frac{y_5 - y_3}{2h}\right\} = \beta_i.$$

Also, the variance of the Del12F estimator is

$$Var\left\{f_i^{\text{D12F}}\right\} = Var\left\{\frac{y_5 - y_3}{2h}\right\} = \frac{\sigma^2}{2h^2}.$$

Similar calculations for the least squares estimator $\hat{\beta}_i$ show that

$$E\{\hat{\beta}_i\} = E\left\{f_i^{\text{D12F-ls}}\right\} = E\left\{\frac{1}{3}\left[\frac{-y_1 + y_2 - 2y_3 + 2y_5 - y_6 + y_7}{2h}\right]\right\} = \beta_i, \text{ and}$$

$$Var\{\hat{\beta}_i\} = Var\left\{f_i^{\text{D12F-ls}}\right\} = Var\left\{\frac{1}{3}\left[\frac{-y_1 + y_2 - 2y_3 + 2y_5 - y_6 + y_7}{2h}\right]\right\} = \frac{\sigma^2}{3h^2}.$$

Given that both estimators are unbiased we can simply compare their variances to see which is better. The Gauss-Markov Theorem states that under the distributional assumptions

we have made, the least squares solution must have minimum variance among all linear unbiased estimators. That is certainly the case here as

$$Var \left\{ f_i^{\text{D12F-ls}} \right\} = \frac{2}{3} Var \left\{ f_i^{\text{D12F}} \right\}.$$

It is important to note that the distributional assumptions may not always hold and therefore the least squares solution may not always be better. Still, this example serves to show how statistical tools and concepts may be used to evaluate the various derivative approximations under different assumptions.

Chapter 4

Differencing

Since we will compare the derivatives approximated by statistical modelling to those approximated by differencing, it is important to understand how the differencing formulas arise. This section provides an introduction to finite differencing. The first subsections discuss differencing results in one dimension and provide detail about the formula derivations. Results for two dimensions follow, ending with a discussion on selecting h , the differencing step size.

4.1 The First Derivative in One Dimension and the Forward Difference

A more formal numerical approximation for $f'(x_0)$ than that described in Section (2) above can be developed, as detailed by Burden and Faires (1993), using interpolation polynomials and the Cauchy remainder theorem. We do not intend to derive the interpolating polynomials or the remainder result here, but give a few details to establish notation.

An example follows below illustrating how each result is applied to the forward difference.

Briefly, given $x_0, x_1, \dots, x_n$ and a function f we can construct a polynomial p of degree at most n such that $f(x_k) = p(x_k)$ for each $k = 0, 1, \dots, n$. The polynomial is written as

$$p_n(f; x) = \sum_{k=0}^n f(x_k) l_k(x)$$

where

$$l_k(x) = \prod_{i \neq k} (x - x_i) / \prod_{i \neq k} (x_k - x_i).$$

Davis (1963) gives a concise explanation of the $l_k(x)$ in $p_n(f; x)$ as polynomials of degree $(n - 1)$ whose values at the x_i resemble the Kronecker delta function. That is, $l_k(x_k) = 1$ and $l_k(x_j) = 0$ if $k \neq j$.

We next state the remainder theorem also given in Davis (1963), and use it to determine the accuracy of the interpolating polynomial.

Theorem 1 - *The Cauchy Remainder for Polynomial Interpolation: Let $f(x) \in C[a, b]$ and suppose that $f^{(n+1)}(x)$ exists at each point of (a, b) . If $a \leq x_0 < x_1 < \dots < x_n \leq b$, then*

$$f(x) - p_n(f; x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_n)}{(n + 1)!} f^{(n+1)}(\xi)$$

where $\min(x, x_0, x_1, \dots, x_n) < \xi < \max(x, x_0, x_1, \dots, x_n)$. The point ξ depends upon $x, x_0, x_1, \dots, x_n$, and f .

Combining the polynomial and remainder results we obtain a representation of f . In particular suppose that $x_0, x_1, \dots, x_n$ are $(n + 1)$ distinct numbers in an interval $I = (a, b)$ and $f \in C^{n+1}(I)$. Using p_n and Cauchy's remainder we can write

$$f(x) = \sum_{k=0}^n f(x_k) l_k(x) + \frac{(x - x_0) \cdots (x - x_n)}{(n + 1)!} f^{(n+1)}(\xi(x)) \quad (4.1)$$

for some $\xi(x)$ in I . Next, differentiating this expression we obtain

$$\begin{aligned} f'(x) &= \sum_{k=0}^n f(x_k)l'_k(x) + D_x \left[\frac{(x-x_0) \cdots (x-x_n)}{(n+1)!} \right] f^{(n+1)}(\xi(x)) \\ &\quad + \frac{(x-x_0) \cdots (x-x_n)}{(n+1)!} D_x [f^{(n+1)}(\xi(x))], \end{aligned} \quad (4.2)$$

where D_x is the differential operator with respect to x (i.e. $D_x f(x)$ means $f'(x)$). A problem occurs when approximating $f'(x)$ for arbitrary x values since we then lack information about $D_x[f^{(n+1)}(\xi(x))]$. However, when x is one of the numbers x_k , some favorable simplification occurs and equation (4.2) reduces to

$$f'(x_k) = \sum_{j=0}^n f(x_j)l'_j(x_k) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{\substack{j=0 \\ j \neq k}}^n (x_k - x_j). \quad (4.3)$$

We can now apply these results to the forward difference. To arrive at a two-point, forward difference approximation, we simply apply equation (4.3) to two datapoints, x_0 and $x_1 = x_0 + h$, for $h \neq 0$. In this case,

$$l_0(x) = \frac{(x-x_1)}{(x_0-x_1)} \quad \text{and} \quad l_1(x) = \frac{(x-x_0)}{(x_1-x_0)}.$$

Then,

$$l'_0(x) = \frac{1}{(x_0-x_1)} \quad \text{and} \quad l'_1(x) = \frac{1}{(x_1-x_0)}.$$

Now substituting into equation (4.3) we have

$$\begin{aligned}
 f'(x_0) &= f(x_0)l'_0(x_0) + f(x_1)l'_1(x_0) + \frac{(x_0 - x_1)}{2}f''(\xi(x)) \\
 &= f(x_0)\frac{1}{(x_0 - x_1)} + f(x_1)\frac{1}{(x_1 - x_0)} + \frac{(x_0 - x_1)}{2}f''(\xi(x)) \\
 &= \frac{f(x_0)}{-h} + \frac{f(x_0 + h)}{h} - \frac{h}{2}f''(\xi(x)) \\
 &= \frac{f(x_0 + h) - f(x_0)}{h} - \underbrace{\frac{h}{2}f''(\xi(x))}_{\text{remainder} = O(h)} \quad \text{so,} \\
 f^{\text{fwd}} &= \frac{f(x_0 + h) - f(x_0)}{h}. \tag{4.4}
 \end{aligned}$$

For small h , equation (4.4), a two-point forward difference, can be used to approximate $f'(x_0)$. This approximation holds with an error that is $O(h)$ if $f''(x)$ is bounded for $x \in I$. So for a given h , the forward difference is a simple calculation involving $f(x_0)$ and $f(x_0 + h)$. Furthermore, as seen in equation (2.1), the fact that a derivative is really a limit suggests that h should be small. The remainder in the line preceding equation (4.4) relates the choice of h to the quality (accuracy) of this forward difference approximation. However, rounding error and machine accuracy will enter the scene when h gets too small.

The forward difference is so labelled when $h > 0$. Otherwise, this formula is known as a backward difference and yields another approximation for $f'(x_0)$. Together, these two differencing formulas provide a method for calculating the derivative of f over the entire interval I , including the endpoints. As we have seen, the forward and backward differences both require two function evaluations. Thus, if the function f has already been evaluated at a specific point x , then approximating the derivative of f at x requires only one more function evaluation. If a third evaluation can be afforded, one could calculate both the forward and backward difference. Then, the two differences could be averaged as

“subsamples” to get a more precise estimate (a central difference) of the derivative at x (we will see how much more precise in the next section). This would produce the following approximation based on three sample points

$$\begin{aligned}
 f^{\text{avg}}(x_0) &= \frac{1}{2}(f^{\text{fwd}}(x_0) + f^{\text{bkwd}}(x_0)) \\
 &= \frac{f(x_0 + h) - f(x_0 - h)}{2h} \\
 &= f^{\text{cen}}(x_0).
 \end{aligned} \tag{4.5}$$

4.2 Three-point formulas and the Central Difference

As illustrated by equation (4.5), other approximations to $f'(x)$ are available that require more than two function calls. In fact, using three equally spaced points

$$x_0, \quad x_1 = x_0 + h, \quad \text{and} \quad x_2 = x_0 + 2h, \quad (\text{for some } h \neq 0)$$

with equation (4.3), three new approximation formulas emerge that exhibit higher accuracy than the forward and backward differences of the previous section. Evaluating equation

(4.3) at each of the three points yields the following expressions

$$\begin{aligned}
 f^{3\text{pt}}(x_0) &= \frac{1}{h} \left[-\frac{3}{2}f(x_0) + 2f(x_1) - \frac{1}{2}f(x_2) \right] + \frac{h^2}{3}f^{(3)}(\xi_0) \\
 &= \frac{1}{h} \left[-\frac{3}{2}f(x_0) + 2f(x_0 + h) - \frac{1}{2}f(x_0 + 2h) \right] + \frac{h^2}{3}f^{(3)}(\xi_0) \\
 &= \frac{1}{2h} [-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)] + \frac{h^2}{3}f^{(3)}(\xi_0)
 \end{aligned} \tag{4.6}$$

$$\begin{aligned}
 f^{3\text{pt}}(x_1) &= f'(x_0 + h) = \frac{1}{h} \left[-\frac{1}{2}f(x_0) + \frac{1}{2}f(x_2) \right] - \frac{h^2}{6}f^{(3)}(\xi_1) \\
 &= \frac{1}{2h} [-f(x_0) + f(x_0 + 2h)] - \frac{h^2}{6}f^{(3)}(\xi_1)
 \end{aligned} \tag{4.7}$$

$$\begin{aligned}
 f^{3\text{pt}}(x_2) &= f'(x_0 + 2h) = \frac{1}{h} \left[\frac{1}{2}f(x_0) - 2f(x_1) + \frac{3}{2}f(x_2) \right] + \frac{h^2}{3}f^{(3)}(\xi_2) \\
 &= \frac{1}{2h} [f(x_0) - 4f(x_0 + h) + 3f(x_0 + 2h)] + \frac{h^2}{3}f^{(3)}(\xi_2)
 \end{aligned} \tag{4.8}$$

Now, redefining x_0 to be $x_0 + h$ in equation (4.7) (i.e. $f(x_0) = f(x_0 + h - h)$ is redefined as $f(x_0 - h)$), and x_0 in equation (4.8) to be $x_0 + 2h$ “centers” these formulas to be approximations for $f'(x_0)$. The simplified expressions yield three different formulas for approximating $f'(x_0)$ based on three points:

$$f'(x_0) = \frac{1}{2h} [-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)] + \frac{h^2}{3}f^{(3)}(\xi_0) \tag{4.9}$$

$$f'(x_0) = \frac{1}{2h} [-f(x_0 - h) + f(x_0 + h)] - \frac{h^2}{6}f^{(3)}(\xi_1) \tag{4.10}$$

$$f'(x_0) = \frac{1}{2h} [f(x_0 - 2h) - 4f(x_0 - h) + 3f(x_0)] + \frac{h^2}{3}f^{(3)}(\xi_2) \tag{4.11}$$

Finally, since equation (4.11) can be obtained from equation (4.9) by substituting $-h$ for h , only the first two equations will be compared for their accuracy (we really only have two derivative approximations).

By observing the remainder terms in the first and second equations above it is clear that the error in equation (4.10) is about half that of equation (4.9). This fact will be useful when comparing two numerical derivative schemes described later. Equation (4.10) is the central difference formula that was mentioned at the end of the previous section and we can now see that its accuracy is much better than the forward difference formula (e.g. $O(h^2)$ vs. $O(h)$). Again the improvement comes at the cost of more function evaluations. If the function f is evaluated at a specific point x , then approximating the derivative of f at x using the central difference requires additional function evaluations at both $(x - h)$ and $(x + h)$. Equation (4.9) also has better accuracy than the forward difference and is useful for approximating derivatives at the endpoint of an interval.

4.3 The Second Derivative in One Dimension

Approximations can also be found for higher order derivatives of a function. If we proceed to differentiate equation (4.2) a second time, we encounter simplification problems with the derivatives in the remainder term and are unable to evaluate the approximation accuracy using our established method. However, Burden and Faires (1993) show that a second order derivative approximation can be easily constructed by expanding the function f in a third order Taylor polynomial about a point x_0 and evaluating this expansion at $(x_0 - h)$ and $(x_0 + h)$ to obtain:

$$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_1)h^4$$

and

$$f(x_0 - h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(\xi_{-1})h^4,$$

where $x_0 - h < \xi_{-1} < x_0 < \xi_1 < x_0 + h$.

Adding these two equations gives

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + f''(x_0)h^2 + \frac{1}{24}[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})]h^4.$$

Finally, solving for $f''(x_0)$ gives

$$f''(x_0) = \frac{1}{h^2}[f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{24}[f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})].$$

Now if $f^{(4)}$ is continuous on $[x_0 - h, x_0 + h]$, we let $f^{(4)}(\xi) = \{f^{(4)}(\xi_1) + f^{(4)}(\xi_{-1})\}/2$, for some ξ , where $x_0 - h < \xi < x_0 + h$. Then $f^{(4)}(\xi)$ is a number between $f^{(4)}(\xi_{-1})$ and $f^{(4)}(\xi_1)$, and the Intermediate Value Theorem permits the equation for $f''(x_0)$ above to be rewritten as

$$\begin{aligned} f''(x_0) &= \frac{1}{h^2}[f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{24}[2f^{(4)}(\xi)] \\ &= \frac{1}{h^2}[f(x_0 - h) - 2f(x_0) + f(x_0 + h)] - \frac{h^2}{12}f^{(4)}(\xi). \end{aligned}$$

A final examination of the remainder term reveals that this approximation is $O(h^2)$ which is comparable in accuracy with the other three-point formulas described in the previous section (though this one is about two times better than the central difference approximation).

Also, this derivative approximation appears to a statistician as a quadratic contrast on the function evaluations at the points $(x_0 - h)$, x_0 , and $(x_0 + h)$.

4.4 Derivatives in Two Dimensions

Moving beyond one dimension involves a shift to vector/matrix notation. Define $\mathbf{h}$ as the vector of step sizes used in the difference calculations, h_i is the i^{th} element in the

$\mathbf{h}$ vector, and $\mathbf{e}_i$ is the i^{th} elementary vector. The various differences from the previous sections are used to “assemble” the gradient vector and Hessian matrix for two and higher dimensional problems. Higher dimensional problems can be handled by taking the variables two at a time and using the two dimensional difference formulas. The gradient of f , $\nabla f(\mathbf{x})$, has for its elements the partial derivatives of f with respect to the elements of $\mathbf{x}$. The partial derivative of f with respect to x_i , $\partial f / \partial x_i$, can be estimated using either forward, backward, or central differences:

$$f_i^{\text{fwd}} = \frac{f(\mathbf{x} + h_i \mathbf{e}_i) - f(\mathbf{x})}{h_i}, \quad (4.12)$$

$$f_i^{\text{bwd}} = \frac{f(\mathbf{x}) - f(\mathbf{x} - h_i \mathbf{e}_i)}{h_i}, \quad (4.13)$$

$$f_i^{\text{cen}} = \frac{f(\mathbf{x} + h_i \mathbf{e}_i) - f(\mathbf{x} - h_i \mathbf{e}_i)}{2h_i}. \quad (4.14)$$

Mixed derivatives, the off-diagonal elements in the Hessian matrix, require more function evaluations. Resembling an interaction contrast, the following formula uses the point at which the mixed derivative is requested as a lower left corner gathering function evaluations above and to the right:

$$f_{ij}^{\text{fwd}} = \frac{(f(\mathbf{x} + h_i \mathbf{e}_i + h_j \mathbf{e}_j) - f(\mathbf{x} + h_i \mathbf{e}_i)) - (f(\mathbf{x} + h_j \mathbf{e}_j) - f(\mathbf{x}))}{h_i h_j} \quad (4.15)$$

This approach is asymmetric and Monahan (2001) employs a symmetrized form, presented later in Section (5.2), which uses function calls on both sides of the point of interest.

4.5 How small should h be?

As mentioned before, taking a derivative is really taking a limit as h tends to zero. However, due to the weaknesses of computer arithmetic, taking h too small risks

encountering wild results. Monahan uses the following example.

Suppose evaluating a function f at x involves a roundoff error ϵ_x . That is, $fl(f(x)) = f(x) + \epsilon_x$ where $fl(\cdot)$ is the floating-point representation operator. Then calculating a forward difference with roundoff error yields

$$\begin{aligned} fl(f^{\text{fwd}}(x)) &= \frac{f(x+h) + \epsilon_{x+h} - f(x) - \epsilon_x}{h} \\ &= \frac{f(x+h) - f(x)}{h} + \frac{\epsilon_{x+h} - \epsilon_x}{h}. \end{aligned}$$

If h is taken too small the roundoff error above will grow, tending to infinity. The (roundoff error)/(step size) ratio will then dominate the derivative approximation rendering it useless. So, it would seem logical to select an h just small enough to avoid roundoff error. Monahan explains that while this seems reasonable, that threshold is not known. Therefore, instead of facing the risks associated with selecting h too small, experience shows that living with some bias in the derivative approximation is preferable. A good rule of thumb is letting $h = \eta x$ where $\eta = \sqrt{\epsilon}$ and ϵ is the machine epsilon. Then the relative step size in the derivative approximation is

$$\frac{(x+h) - x}{x} = \frac{(x+\eta x) - x}{x} = \eta.$$

Dennis and Schnabel (1983) explain that when considering a forward difference,

$$\frac{f(x+h) - f(x)}{h},$$

as an approximation to $f'(x)$, we must consider two errors. The forward difference is $O(h)$ in exact arithmetic, which suggests that smaller values of h will yield a better approximation. However, when considering that the numerator of the forward difference must be

evaluated using computer arithmetic, another error $\delta = f(x + h) - f(x)$, occurs. The error in the numerator results from errors in the function values themselves and in potentially catastrophic cancellation errors due to subtraction. That is, if h is too small, $x + h$ may be close enough to x that $f(x + h)$ is nearly equal to $f(x)$. As a result, the good digits in δ are lost leaving only round-off error. Finally, evaluating δ/h further inflates the error. Dennis and Schnabel point out that, in this case, the forward difference may have very few or zero good digits. Therefore, when selecting h , both $O(h)$ and δ/h must be considered.

As rules of thumb, Dennis and Schnabel recommend choosing $h = \epsilon^{1/2}x$ for forward differences, and $h = \epsilon^{1/3}x$ for central, second order, and mixed derivatives.

Chapter 5

Competing methods

5.1 The Spendly Method

One approximation method for calculating the gradient and Hessian is developed by Spendly, Hext, and Himsworth (1962) (hereafter “Spendly”). This method collects function evaluations over a simplex region of points to fit a quadratic response surface model. The coefficients from the quadratic model are used to assemble the gradient vector and Hessian matrix at the point $\mathbf{x}$.

The simplex region is formed by a corner point $\mathbf{x}$, simplex vertices $\mathbf{x} + h_i \mathbf{e}_i$, midpoints between the corner point and the vertices $\mathbf{x} + (h_i/2) \mathbf{e}_i$, and the midpoint between the vertices $\mathbf{x} + (h_i/2) \mathbf{e}_i + (h_j/2) \mathbf{e}_j$.

The number of function evaluations required for this approach is $d(d+3)/2 + 1$ where d is the dimension of the problem. Since the number of evaluations is equal to the number of coefficients in the quadratic response surface model, this approach is saturated and provides no information to verify the model fit. That is, all the data collected is used

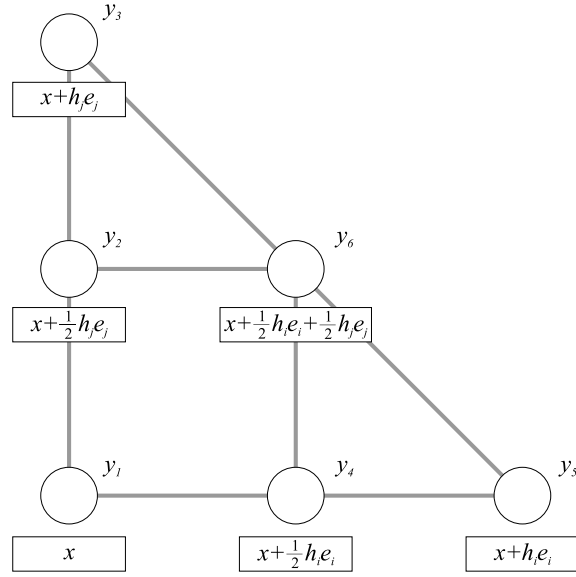


Figure 5.1: Spendly simplex design grid and function evaluation locations

to estimate the model coefficients leaving none remaining to evaluate the model itself. This design is also not symmetric and cannot employ central difference style approximations, potentially resulting in some bias or reduced accuracy.

For a two dimensional problem where $\mathbf{x} = (x_1, x_2)$ and $h_1 = h_2 = h/2$, the solutions from this scheme are as follows. Figure 5.1 illustrates the simplex design grid and labels the function evaluation locations as y_i , $i = 1, 2, \dots, 6$. The derivative approximation calculations are:

$$f_1^{\text{Spnd}} = \frac{1}{2} \left[\frac{-3y_1 + 4y_4 - y_5}{h/2} \right] \quad (5.1)$$

$$f_2^{\text{Spnd}} = \frac{1}{2} \left[\frac{-3y_1 + 4y_2 - y_3}{h/2} \right] \quad (5.2)$$

$$f_{12}^{\text{Spnd}} = \frac{y_1 - y_2 - y_4 + y_6}{(h/2)^2} \quad (5.3)$$

$$f_{11}^{\text{Spnd}} = 2 \frac{y_1 - 2y_4 + y_5}{h^2} \quad (5.4)$$

$$f_{22}^{\text{Spnd}} = 2 \frac{y_1 - 2y_2 + y_3}{h^2} \quad (5.5)$$

We point out the similarity between equations (5.1) and (5.2), and equation (4.9) from Section (4.2). Also recall that the errors using this approximation are about double those made using the central difference approximation.

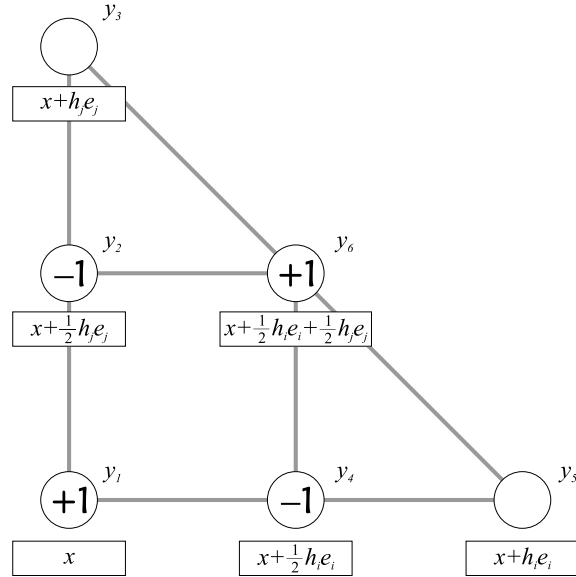


Figure 5.2: Spendlly Hessian matrix function evaluation locations and coefficients

Figure 5.2 graphically illustrates the function evaluations and their coefficients used by Spendly to approximate the mixed derivative.

5.2 Del12F

From the results in Section (4.2) we know that, though requiring more work, the central difference formulas outperform forward differences. As mentioned earlier, the Del12F scheme employs the central difference approximation for the gradient and a symmetrized formula for calculating the elements of the Hessian matrix. The symmetrized formula for the Hessian matrix calculations is as follows:

$$f_{ij}^{\text{D12F}} = \frac{1}{2h_i h_j} \left[[f(\mathbf{x} + h_i \mathbf{e}_i + h_j \mathbf{e}_j) - f(\mathbf{x} + h_i \mathbf{e}_i)] - [f(\mathbf{x} + h_j \mathbf{e}_j) - f(\mathbf{x})] + [f(\mathbf{x} - h_i \mathbf{e}_i - h_j \mathbf{e}_j) - f(\mathbf{x} - h_i \mathbf{e}_i)] - [f(\mathbf{x} - h_j \mathbf{e}_j) - f(\mathbf{x})] \right].$$

The graphical representation of this mixed-derivative formula is shown in Figure 5.3.

Working in d dimensions, this method requires $d(d+1) + 1$ function evaluations but greatly reduces the error (the remainder is $O(h^2)$). As discussed in Chapter 3, it is also interesting to consider a modelling approach to Del12F. For illustration, consider the same two-dimensional problem and view the Del12F layout as the statistical design described in Chapter 3. The least squares estimates for a quadratic response surface model yield the following formulas:

$$f_1^{\text{D12F}} = \frac{1}{3} \left[\frac{-y_1 + y_2 - 2y_3 + 2y_5 - y_6 + y_7}{2h_1} \right] \quad (5.6)$$

$$f_2^{\text{D12F}} = \frac{1}{3} \left[\frac{2y_1 + y_2 + y_3 - y_5 - y_6 - 2y_7}{2h_2} \right] \quad (5.7)$$

$$f_{12}^{\text{D12F}} = \frac{1}{2} \left[\frac{-y_1 + y_2 - y_3 + 2y_4 - y_5 + y_6 - y_7}{h_1 h_2} \right] \quad (5.8)$$

$$f_{11}^{\text{D12F}} = \frac{y_3 - 2y_4 + y_5}{h_1^2} \quad (5.9)$$

$$f_{22}^{\text{D12F}} = \frac{y_1 - 2y_4 + y_7}{h_2^2} \quad (5.10)$$

The formulas for the mixed and second order derivative approximations are exactly those used in the Del12F algorithm (remember to double the quadratic coefficients, see eq. (5.12) below). That is, Del12F uses the least-squares solutions for these approxima-

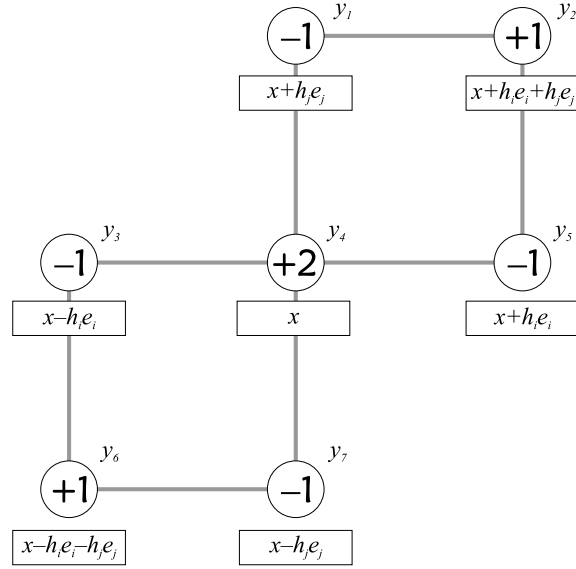


Figure 5.3: Del12F mixed derivative approximation function evaluation locations and coefficients

tions. The only difference between Del12F and a least squares approach is in the first-order derivative approximations, as pointed out in Chapter 3.

5.3 A Statistical Perspective

Viewing the derivative approximation problem from a statistical perspective we seek to benefit from the tools already available in the discipline. Chapter 3 demonstrated the usefulness of statistical design in producing an alternative mixed derivative approximation to that used by the Del12F algorithm. Furthermore, given two approximations to the same derivative, Chapter 3 also demonstrated the usefulness of statistical theory in comparing these approximations.

The goals of this research are to build response surface designs that allow estimation of a quadratic model—resulting in derivative approximations—requiring fewer evaluations than Del12F. We also seek to provide a method for comparing derivative estimates arising from both least squares and differencing methods that considers various distributional assumptions on the function evaluations. Finally, computer code will be created to calculate least squares derivative approximations based on the response surface designs.

In relation to the competing methods of Spendly and Del12F, the response surface designs have some key properties. These designs will improve on the Del12F approach in reducing the number of function evaluations required to estimate the gradient and Hessian matrix at a given point. However, their sample size is still larger than the Spendly method; therefore, they are not saturated and may provide feedback on the model goodness of fit and the appropriateness of the scaling used.

The sections that follow introduce the regression model approximation to a function and provide details on approximating derivatives with this model. Since the model must be fit to data, a brief section describing response surface designs is also included.

5.3.1 Response Functions and Differentiation

Given $f(x_1, x_2, \dots, x_d)$, where f is a scalar-valued function of a d -dimensional vector $\mathbf{x}$, we wish to calculate the gradient and Hessian matrix of f at $\mathbf{x}$. Additionally this is to be accomplished by first approximating f with a simple quadratic polynomial, and then differentiating the approximation. Regarding f , Khuri and Cornell (1987) explain that if f is smooth and continuous, we can employ a Taylor Series expansion and approximate f as a polynomial. For a specific example, suppose f is *exactly* a second-degree polynomial in the region of the levels of two variables x_1 and x_2 . Then,

$$\begin{aligned} f(x_1, x_2) &= \beta_0 + \mathbf{x}^T \mathbf{b} + \frac{1}{2} \mathbf{x}^T \mathbf{B} \mathbf{x} \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2, \end{aligned} \tag{5.11}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}, \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 2\beta_{11} & \beta_{12} \\ \beta_{12} & 2\beta_{22} \end{bmatrix}.$$

Equation (5.12) can be viewed as a regression function where β_1 , β_2 , β_{11} , β_{22} , and β_{12} are simply regression coefficients with intercept β_0 , and the variables x_1 , x_2 , x_1^2 , x_2^2 , and $x_1 x_2$ are explanatory variables.

Taking the first and second derivatives of f with respect to $\mathbf{x}$ illustrates how the described regression approach can be useful for numerical differentiation:

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{b} + \mathbf{B}\mathbf{x} \Big|_{\mathbf{x}=\text{point of interest}} \quad (5.12)$$

$$\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}'} = \mathbf{B}. \quad (5.13)$$

As shown above, the vector $\mathbf{b}$ and matrix $\mathbf{B}$ are each composed of coefficients that can be estimated from a regression model. For a specific point $\mathbf{x} = (x_1, x_2)^\top$, the regression coefficients in $\mathbf{b}$, β_1 and β_2 , are the values of the first-order partial derivatives, $\partial f / \partial x_1$ and $\partial f / \partial x_2$, of f with respect to $\mathbf{x}$ evaluated at the origin. Likewise, the coefficients in $\mathbf{B}$, $2\beta_{11}$, $2\beta_{22}$, and β_{12} , are defined as the values of the second-order partial derivatives, $\partial^2 f / \partial x_1^2$, $\partial^2 f / \partial x_2^2$, and $\partial^2 f / \partial x_1 \partial x_2$.

5.3.2 Estimating the Approximate Response Function

Having established that the gradient and Hessian of f can be estimated by fitting a regression model, we now consider the details of collecting the data. Experimental design techniques deal with this very issue and will be used to decide the values of $\mathbf{x}$ at which the function f will be evaluated. In a typical design setting, the structural form of f is unknown and often complicated. Clearly, an approximate model is useful only if it mirrors the unknown function. The seemingly difficult task of modelling the unknown function may be simplified if we create the approximation over a focused region of interest (e.g. in a neighborhood of $\mathbf{x}$). In particular, quadratic models are frequently useful when the region of interest is sufficiently small. Even complex, non-quadratic functions can be well fit by a quadratic model if the region of interest is appropriately restricted.

Response surface methodology (or RSM) will be used to estimate the f function using a quadratic polynomial. Lawson and Erjavec (2000) describe how RSM is used to design and collect experimental data to allow fitting a general quadratic equation for prediction. They propose that any situation can be described by a mathematical model if we know enough about it. Lawson and Erjavec also mention that, as described in Section (5.3.1), the quadratic equation is somewhat equivalent to expanding the true model in a Taylor series about the center of the region and then dropping all terms greater than order two. Most importantly, they point out that if the region of interest is of modest size, the quadratic (polynomial) fit is quite good.

5.3.3 RSM Design Considerations

Table 5.1 lists the RSM design sample sizes indexed by the problem dimension. The sample sizes in the small- n column for dimensions 2 through 10 are taken from the literature (see Hartley (1959), Westlake (1965), and Draper and Lin (1990)). These sample sizes were regressed on d (the number of dimensions) and extrapolated predictions were used as a simple guideline in selecting sample sizes for dimensions 11 through 20. In general, the RSM sample sizes grow exponentially with the dimension of the problem, but by fractioning the factorial design, this growth is reduced to quadratic. The table also includes a column for Del12F, showing the number of required function calls. The final column lists the number of coefficients in a quadratic model and therefore the number of function evaluations required by the Spendly approach. This number represents the absolute minimum number of experiments or function calls required to estimate the quadratic model.

The application of designed experiments to the problem of numerical differentiation is done with the hope of reducing the required number of function evaluations. For a problem of a given size, the basis of comparison for the number of required runs will be the number required by Del12F.

The family of small- n central composite designs (or CCD) consists of a fractional factorial design augmented with axial points to allow for estimation of curvature. For small problems (in dimension), the number of runs or evaluations they require is not too much more than the number of coefficients in the quadratic model to be fit, for larger problems the

Table 5.1: Number of Experiments Required for Response Surface Designs

Number of Dimensions	Small- n CCD	Aug. Small- n CCD	Del12F	Spendly Method
2	7	11	7	6
3	11	17	13	10
4	17	25	21	15
5	22	32	31	21
6	29	41	43	28
7	37	51	57	36
8	47	63	73	45
9	57	75	91	55
10	67	87	111	66
11	90	112	133	78
12	105	129	157	91
13	119	145	183	105
14	135	163	211	120
15	153	183	241	136
16	171	203	273	153
17	190	224	307	171
18	209	245	343	190
19	231	269	381	210
20	253	293	421	231

Note: The number of coefficients in the quadratic model corresponds to the number of evaluations required by the Spendly method.

sample size can be considerably larger than the minimum (though still less than Del12F). In using these designs one must typically forfeit the usual model diagnostics such as the lack of fit test and residual plots. Thus, in exchange for a reduced number of runs, we must be willing to assume that we are approximating the function in a region small enough for the quadratic model to fit well.

5.3.4 Constructing the RSM Designs

Designs for dimensions 2 through 10 are available in the literature and in some statistical analysis software. The designs are basically some type of fractional factorial “cube” with appended axial points. Following similar principles for the larger designs (dimensions 11-20), a resolution five fractional factorial was generated using the FACTEX procedure in SAS with a sample size greater than what we desired for our actual design. We then used the SAS OPTEX procedure to select a D-Optimal subset from this fraction that could still estimate all the main effects and two-factor interactions in the model. Our starting value for the D-optimal subset size was the extrapolated value described in the previous section. We then tested smaller sample sizes by hand to see if a smaller design could be found.

It is interesting to note that the final sample sizes selected for the RSM (Small- n , not Aug. Small- n) method closely follow the number of function evaluations required by the Spendly method, which exhibits (exact) quadratic growth. For dimensions 2 through 10, the samples sizes are nearly equal, indicating that the small- n sample sizes grow quadratically as well. For dimensions 11 through 20, a regression of Spendly sample size on dimension

yields

$$y_{\text{Spendly}} = -127 + 17.5d + 0.5(d - 16)^2,$$

where the same fit for the small- n designs yields

$$y_{\text{Small-}n} = -126.2 + 18.6d + 0.5(d - 16)^2.$$

The similarity of these two regression models again shows that the small- n sample sizes grow quadratically with the dimension of the problem. However, since the sample sizes are determined by hand, we can only describe a general trend by examining these models, but are not necessarily able to predict the sample sizes.

To aid in assessing the goodness of fit for a small- n central composite design, we augment the fractioned or D-optimal cube (corner points) with face points to form an *inner design*. This design is then further augmented with the star points at ± 1.25 to form an *outer design*. We can use the *inner design* to predict values for actual function evaluations at the axial points. These estimates can be compared to actual evaluations at the axial points to detect possible lack of model fit.

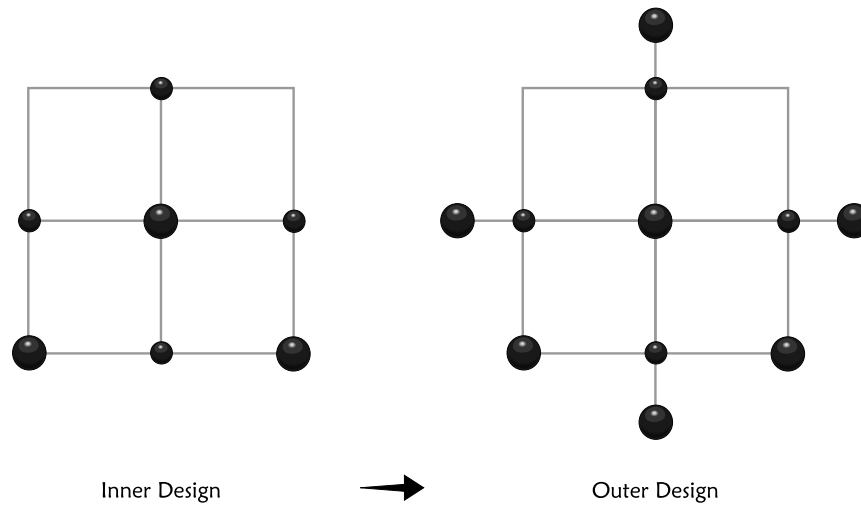


Figure 5.4: Inner and Outer Response Surface Designs in two dimensions

5.3.5 Statistical Issues

In order to justify our approach, a few statistical issues relating to the linear model assumptions must be considered.

Replication allows estimation of the unknown pure error variance which is used in significance tests and prediction. Given that we are working in a deterministic situation, replication is pointless and yields no information.

Replication is also useful in testing the model goodness of fit when higher-order terms are estimable, but not included in the model. In this situation, the residual variability can be partitioned into pure error and model lack of fit error (variability associated with the higher-order terms not included in the model) which allows testing the reduced model goodness of fit. Our designs will only estimate a quadratic model without any higher-order (i.e. cubic) terms. Without replication and higher order terms we are unable to perform

lack of fit tests. To deal with this issue we will estimate the model coefficients using the *inner design* and have function evaluations available at the axial points. We will use the inner design to forecast values for the axial points and then compare these estimates to the actual function evaluations. If there is a large enough difference between the forecasts and the actual values, we will suspect that the model is not scaled properly and exhibits lack of fit.

Finally, linear models are usually fit under an assumption that the errors are uncorrelated. That is, the response from any one run has no effect on the response or error for any other run. Data arising from a computed function may not meet this assumption since computers do not have a random error structure. While more complicated linear models may deal with correlated errors, it may be that finite differencing outperforms linear modelling in these situations. In any case, we will consider the least squares solution as it is still unbiased, and somewhat robust to non-independent errors in the data.

Chapter 6

A Model for Evaluating Derivative Approximations

Having reviewed the methods for approximating derivatives, we are now interested in comparing their performance. As described earlier, the Spendly method is very efficient, requiring the least number of function evaluations possible to approximate the gradient and Hessian. However, the point at which the derivative is approximated is located on the edge of the design, instead of the center. Alternatively, one could use the Del12F approach which is symmetric about the point of interest, but requires a higher number of function evaluations. Finally, the response surface method requires a sample size smaller than Del12F and provides feedback on variable scaling, but sacrifices some precision.

Choosing among the three approximation methods seems difficult at this point, and it would be helpful to somehow quantify the trade-offs between precision, sample size, and variable scaling feedback.

The model we introduce in this chapter provides a framework for comparing the derivative approximations and helps quantify the benefits of each method.

6.1 Comparing Methods

To compare the three methods, the usual approach involves choosing a test bed of sample functions to differentiate. Then comparisons might be made by checking the derivative approximations against the analytical derivative of the sample functions at several points. However, it is not clear how functions should be selected for inclusion in the test bed. Given the infinite number of functions from which to choose, selecting a representative sample that guarantees a fair and thorough comparison is a difficult and subjective task.

Ideally, the test bed would consist of functions that meet the following criteria. First, we want a population of functions to sample from that have a prescribed level of curvature, where the second derivative does not change too quickly as to make estimation of it fruitless. For example, introducing response surface models in Section (5.3.1), we discussed a basic requirement for the method that functions be locally quadratic around the point of interest. This one criterion clearly eliminates many functions from consideration, but there are still infinitely many more from which to choose. Second, we can consider the test functions in light of the particular application we are interested in. For a statistical application like maximum likelihood, we might only consider test functions that are similar to the likelihoods of interest. The same can be said for any nonlinear model of interest.

While these guidelines may help suggest functions for consideration, conclusions drawn from comparing the derivative methods would be conditional on the particular test

bed. Then it may be possible for one to argue that a particular function should or should not be included. Or, we may inadvertently select functions that somehow favor one of the derivative approximation methods over another.

Since our interest is in general conclusions about the derivative methods over as wide a class of test functions as possible, we want to avoid the issues involved with selecting specific test functions. Instead, we prefer to define a population of functions that meet a prescribed level of curvature, which are similar to those we deal with in practice. One possible approach uses stochastic process models for a random function. Specifically, the model to compute evaluations is the sum of two random components:

$$\mathbf{y} = fl(f(\mathbf{x})) = f(\mathbf{x}) + \epsilon_x,$$

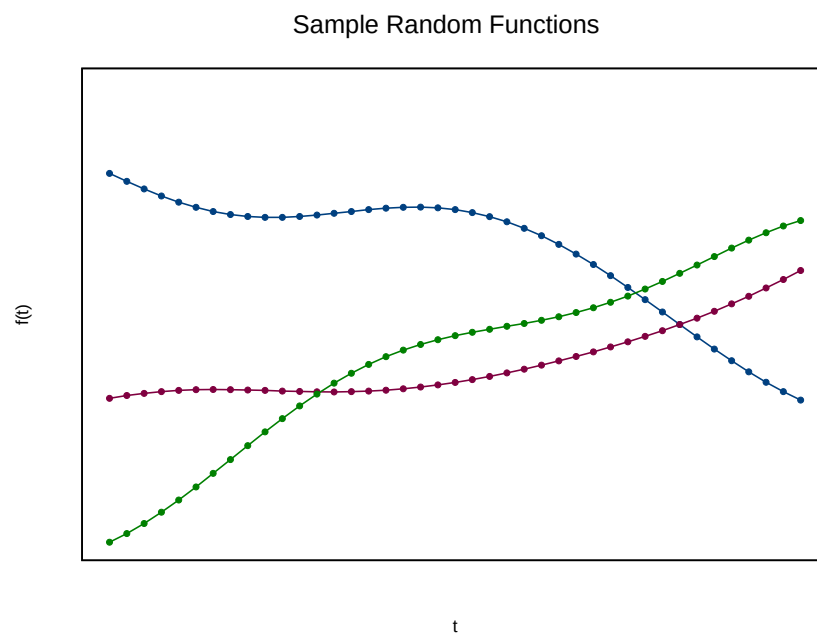


Figure 6.1: Plot of Three Sample Random Functions.

where $f(\mathbf{x})$ is the value of a random function evaluated at $\mathbf{x}$, and ϵ_x is the round-off error at $\mathbf{x}$. We use covariance matrices to represent the variability associated with each piece of the model, and the covariance matrix of the $f(\mathbf{x}_i)$ dictates the curvature of the function. If f has low curvature and $\mathbf{x}_i$ is near $\mathbf{x}_j$, then $f(\mathbf{x}_i)$ should be highly correlated with $f(\mathbf{x}_j)$. The variability due to the random function will be modelled as a covariance matrix $\mathbf{\Sigma}$ for the function evaluations. To this covariance matrix, we add a diagonal matrix $\mathbf{D}$ representing round-off error. This matrix is defined as

$$\mathbf{D} = \begin{bmatrix} \sigma_r^2 \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}, \quad (6.1)$$

where σ_r^2 represents the variability due to round-off errors in floating point arithmetic. Though we represent round-off errors using a random variance component in our model, we point out that round-off errors in computer arithmetic are not random. The scaled identity matrix in the upper-left corner of $\mathbf{D}$ is $(n \times n)$, where n is the number of coefficients in the derivative approximation linear combination. The form of $\mathbf{D}$ highlights that while each function evaluation is affected by round-off error, these round-off errors are uncorrelated.

Using this model, we can generate sample functions such as those in Figure 6.1 that look just like real-life functions we need to differentiate. More importantly, we can use the model to evaluate and compare the derivative approximations we introduced earlier without selecting a specific function. Using a metric such as mean squared error (MSE), we can estimate the precision of each derivative approximation, and use that precision estimate to compute optimal step sizes (h). Also, since round-off error is incorporated into the model, we will be able to see the effect round-off error has on the derivative approximation precision and the corresponding step size.

6.2 An Overview of Random Functions

The previous section describes some issues concerning test functions that we hope to avoid by using randomly generated functions with a prescribed level of curvature. The advantage to this approach is that we can test the derivative methods on functions that belong to a large (infinite) class without ever needing to work with a specific function at all. One great advantage to this approach is that conclusions we draw apply generally across the entire class of functions. Now the problem of selecting the function test bed is reduced to establishing a criterion indicating if a function belongs to the class or not.

Working with random functions is not without difficulty, and the following overview describes some of the important details and challenges. We first require a way to express the random function so as to be able to control its curvature. This is accomplished by selecting a set of function abscissae and assigning a covariance model for the random function evaluations at these points. A properly selected covariance function provides a method for controlling curvature, and must be twice differentiable. This differentiability requirement is necessary for a stochastic process result that we introduce later, and comes as we are estimating second derivatives. The random function value at each abscissa is a random variable that can follow any probability distribution, and we use the normal distribution for convenience. We also incorporate an additional random component into the covariance function that represents the round-off error in floating arithmetic. Finally, we evaluate the derivative approximations by considering various functions of the random function evaluations. As we work with evaluations at many abscissae and in multiple dimensions to approximate a derivative, this is clearly a fairly complex multivariate problem.

6.3 A Useful Covariance Function

The preceding overview points out the necessity of a proper covariance function for creating random functions with appropriate curvature. For a random function f with $\mathbb{E}f(\mathbf{x}) = 0$, we introduce one such covariance function,

$$\text{Cov}\{f(\mathbf{x}), f(\mathbf{x} + \mathbf{h})\} = \sigma^2 \exp\left\{-(\mathbf{a}^\top \mathbf{h})^2\right\}, \quad \mathbf{a}, \mathbf{h} \in \mathbb{R}^d, \quad (6.2)$$

where $\text{Cov}\{f(\mathbf{x}), f(\mathbf{x} + \mathbf{h})\}$ is stationary (i.e. depends on $\mathbf{h}$, not $\mathbf{x}$). This spatial covariogram (Cressie 1991) performs well for modelling functions like those we are interested in. Furthermore, it is computationally easy to work with and possesses all of the necessary derivatives. Finally, as already mentioned, the recommended step sizes calculated to minimize the derivative approximation variance using this covariance function are similar to those suggested by Dennis and Schnabel (1983).

A useful feature of this covariance function lies in choosing values for the elements of $\mathbf{a}$. Defining $\mathbf{h}$ as the distance between two points, the elements of $\mathbf{a}$ control the function curvature in the different dimensions. Statistically speaking, using equation (6.2), the covariance between two points is a function of the distance between them, diminishing as the distance increases. In other words, function abscissae that are close in proximity to each other are more likely to have similar function evaluations than those further apart. We later exploit this property both in generating instances of the random function, and in evaluating the diagnostics for detecting variable scaling problems.

Table 6.1 shows some results for a few specific derivative approximations. More detail follows regarding the variance and step size calculations, but for now we note that the optimal step sizes for the particular derivative approximations are similar to those recommended by Dennis and Schnabel (1983).

6.4 Generating Random Functions

In generating a realization of the random function we actually only work with a vector of function evaluations, not the random function itself. To illustrate this approach, we will describe the steps to generate an instance of the random function in one dimension.

We begin by selecting an interval in $\Re$, $[-1, 1]$ say, and choose n abscissae $\{t_i, i = 1, \dots, n\}$ in this interval. Place the selected points in a vector $\mathbf{t}$, and using equation (6.2) calculate the covariance matrix Σ for the vector of function evaluations $f(\mathbf{t})$ as:

$$\text{Cov}\{f(t_i), f(t_j)\} = \Sigma_{ij} = \begin{cases} \exp\{-a^2(t_i - t_j)^2\} & (i \neq j) \\ 1 & (i = j) \end{cases},$$

where a (a scalar here) is the function curvature parameter. An interesting range of values for a is $[\frac{1}{2}, 10]$, with smaller values of a resulting in functions with less curvature.

For plotting only, evaluations of the random function itself are based on any one of many statistical distributions, and as mentioned earlier, we will use the standard normal distribution. We next construct a vector of initial random function evaluations, $\mathbf{w} = (w_1, w_2, \dots, w_n)'$, sampled from the selected distribution (in this example, w_i iid $N(0,1)$).

The next step in creating the random function evaluations is transforming $\mathbf{w}$ into $\mathbf{y} = f(\mathbf{t}) = \Sigma^{1/2}\mathbf{w}$, where $\mathbf{y} \sim N_n(\mathbf{0}, \Sigma)$, and $\Sigma^{1/2}$ is the square root of the covariance

Table 6.1: Table of Method Variances and Optimum Step Sizes (h)

Method	Approximate Variance	Optimum h
Forward difference	$\frac{2\sigma_r^2}{h^2} + 3\sigma^2 a^4 h^2 + O(h^4)$	$0.90 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{2}}$
Central difference	$\frac{\sigma_r^2}{2h^2} + \frac{10\sigma^2 a^6 h^4}{3} + O(h^6)$	$0.65 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Differencing 2 nd der	$\frac{6\sigma_r^2}{h^4} + \frac{35\sigma^2 a^8 h^4}{3} + O(h^6)$	$0.92 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{4}}$
Spendly _{1,1}	$\frac{26\sigma_r^2}{h^2} + \frac{5\sigma^2 a^6 h^4}{6} + O(h^6)$	$1.58 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Spendly _{1,11}	$\frac{96\sigma_r^2}{h^4} + 30\sigma^2 a^6 h^2 + O(h^4)$	$1.36 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Del12F LS 1 st der	$\frac{\sigma_r^2}{3h^2} + 6\sigma^2 a^6 h^4 + O(h^6)$	$0.55 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Small- $n_{2,1}$	$\frac{5\sigma_r^2}{12h^2} + \frac{25\sigma^2 a^6 h^4}{6} + O(h^6)$	$0.61 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Small- $n_{2,2}$	$\frac{\sigma_r^2}{2h^2} + \frac{10\sigma^2 a^6 h^4}{3} + O(h^6)$	$0.65 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Small- $n_{3,1}$	$\frac{\sigma_r^2}{2h^2} + \frac{10\sigma^2 a^6 h^4}{3} + O(h^6)$	$0.65 \frac{1}{a} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$
Small- $n_{3,11}$	complicated	$0.90 \left(\frac{1}{a} \right)^{\frac{5}{6}} \left(\frac{\sigma_r}{\sigma} \right)^{\frac{1}{3}}$

 σ_r^2 = round-off error variance a = random function curvature parameter σ^2 = random function scale parameter

matrix Σ . Initially, a Cholesky root was attempted to calculate $\Sigma^{1/2}$. However, due to the smoothness of the covariance function, Σ was nearly singular and the Cholesky factorization routine had problems with numeric precision. Alternatively, since this covariance matrix is symmetric and positive definite, we calculated a spectral decomposition as $\Sigma = \mathbf{C}\mathbf{\Lambda}\mathbf{C}'$ (see Rencher 1995, p. 41). Here, $\mathbf{C}$ contains the normalized eigenvectors and $\mathbf{\Lambda}$ is diagonal containing the corresponding eigenvalues. To deal with the round error we retained only eigenvalues greater than 1e-15, setting any others to zero on the $\mathbf{\Lambda}$ diagonal. Next, we calculated $\mathbf{\Lambda}^{1/2}$ by taking the square root of the diagonal elements of $\mathbf{\Lambda}$, and then formed $\Sigma^{1/2} = \mathbf{C}\mathbf{\Lambda}^{1/2}\mathbf{C}'$. Finally, a plot of (y_i, t_i) shows what the function looks like (recall Figure 6.1).

Using the method just described, we can generate functions while varying the shape parameter a to examine its effect on the resulting function realizations. Then, once an appropriate value of a is determined, we can fix a and proceed to compare the variance approximations using the random function. An appropriate value for a produces appropriately curved functions behaving similarly to those differentiated in the particular field of application (e.g. likelihood functions, nonlinear response functions, etc.).

To aid in deciding on a value for a , 15 random functions were generated for values of a between 0.2 and 1.6. Figures 6.2 through 6.4 show sample functions for $a = 0.7$, 1.0, and 1.3, respectively. Next, a quadratic regression model was fit to each function and the R^2 values from these fits were plotted against the value of a . Based on this plot, we thought it reasonable to require that functions have an average R^2 value of at least 0.90 so as to obtain functions as noisy (curved) as possible, without throwing into question whether

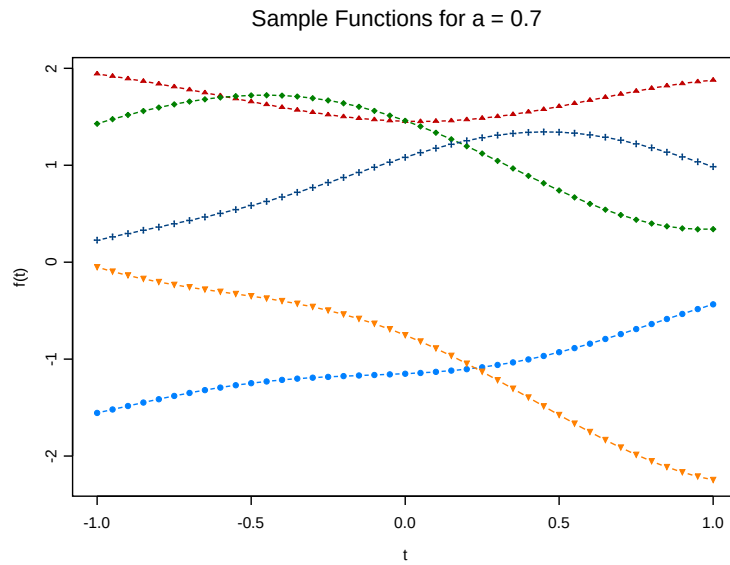


Figure 6.2: Plot of 5 Random Functions with curvature parameter $a = 0.7$.

seeking optima of the function makes sense. Values of a larger than 1.0 exhibit individual functions with R^2 values much less than 0.90, indicating they have higher curvature than is suitable for our purposes. We therefore decided to use $a = 1$ in generating functions for testing the derivative approximations.

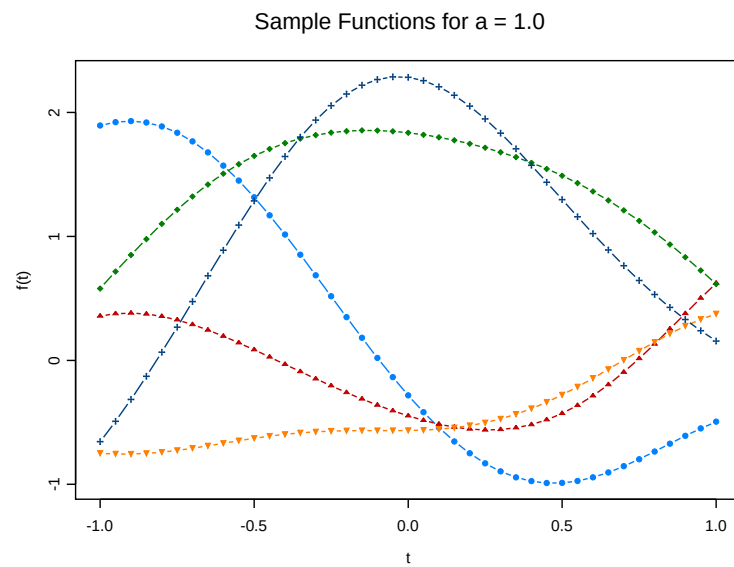


Figure 6.3: Plot of 5 Random Functions with curvature parameter $a = 1.0$.

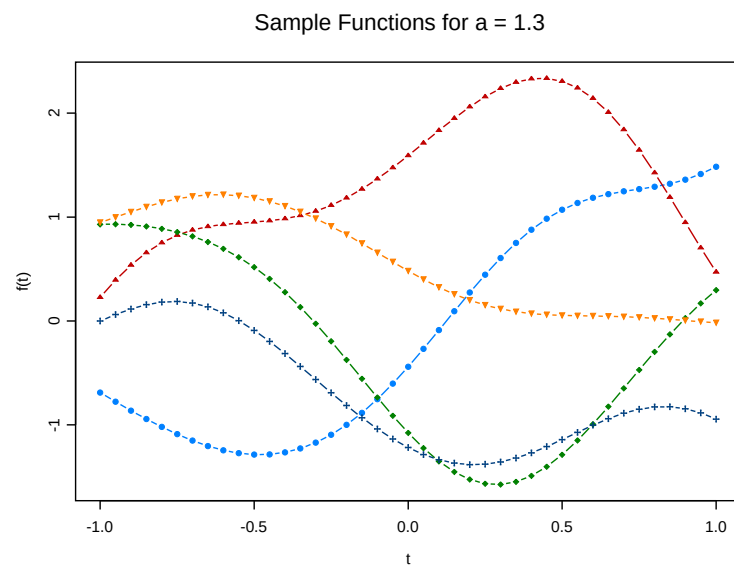


Figure 6.4: Plot of 5 Random Functions with curvature parameter $a = 1.3$.

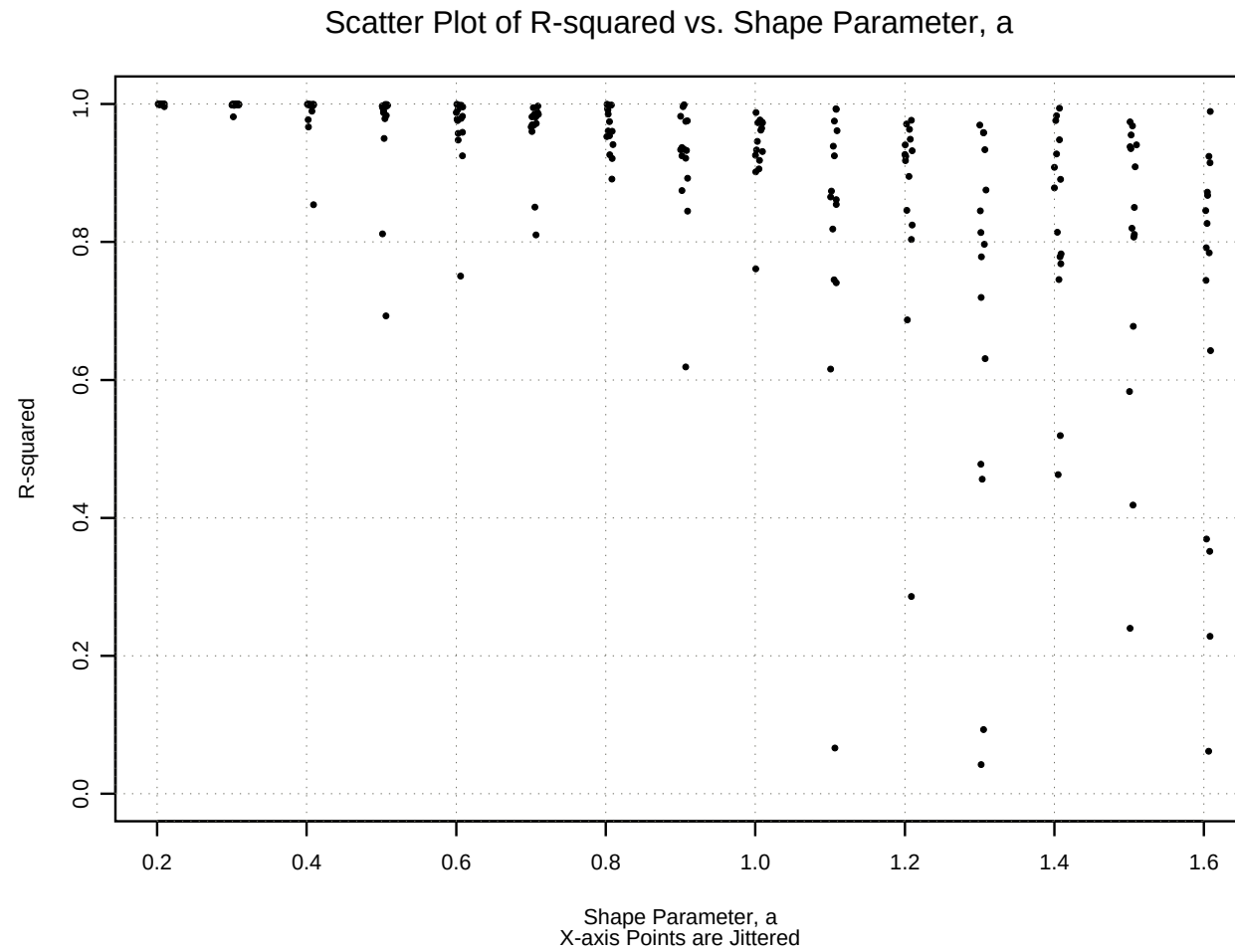


Figure 6.5: Scatter Plot of R-Square by Shape Parameter a for a One-Dimensional Random Function

6.5 Derivative Approximations as Linear Combinations

At this point, it is helpful to notice that all three derivative approximations we consider can be written in a general form which makes them amenable to modelling. The generalization comes by simply expressing each approximation as a linear combination of function evaluations. For example, the central difference approximation,

$$f^{\text{cen}}(x) = \frac{f(x_0 + h) - f(x_0 - h)}{2h},$$

can be written generally as a linear combination of function evaluations $\boldsymbol{\ell}^\top \mathbf{y}$, where

$$\boldsymbol{\ell} = \begin{bmatrix} \frac{1}{2h} \\ -\frac{1}{2h} \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} f(x_0 + h) \\ f(x_0 - h) \end{bmatrix}.$$

Both the RSM and Spendly methods are based on least squares, so the approximations they produce are linear combinations of function evaluations.

We are ultimately interested in the variance of the difference of a derivative approximation and the actual derivative at a particular point as

$$\mathbb{E} \{f^{\text{cen}}(t) - f'(t)\}^2 = \text{Var} \{f^{\text{cen}}(t) - f'(t)\},$$

since $\mathbb{E}f^{\text{cen}}(t) = \mathbb{E}f'(t) = 0$. In comparing the derivative approximations, smaller variances are better.

Now, for a particular approximation, the difference in the variance function above can also be written as a linear combination in vector notation using

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \\ -1 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ f'(\mathbf{t}) \end{bmatrix}, \quad (6.3)$$

where the z_i are the coefficients in the derivative approximation linear combination, the y_i are function evaluations, and $f'(\mathbf{t})$ is the exact derivative evaluated at a point of interest, $\mathbf{t}$. For Del12F, the z_i in a particular derivative approximation are the coefficients in the approximations derived in Sections (4.2), (4.3), and (4.4). For the Spendly and RSM methods, the z_i for a particular derivative can be calculated using least squares as $(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$, or they can be obtained from a numerical procedure like the Householder or Givens transformation. Once $\mathbf{z}$ is obtained, the variance of interest can be computed as

$$\begin{aligned} \text{Var}(\mathbf{z}^\top \mathbf{y}) &= \mathbf{z}^\top \text{Cov}(\mathbf{y}) \mathbf{z} \\ &= \mathbf{z}^\top (\mathbf{\Sigma} + \mathbf{D}) \mathbf{z}, \end{aligned} \quad (6.4)$$

where $\mathbf{\Sigma}$ is the covariance matrix of the function evaluations and $\mathbf{D}$ is the diagonal round-off error matrix described earlier. Note that the variance corresponding to the last component in $\mathbf{y}$ (i.e. $f'(\mathbf{t})$) will be 0. Since only the covariances between the function evaluations and the derivative at a particular point are needed, we define a general covariance function for $\mathbf{y}$ (where $\mathbf{y}$ is defined as in eq. (6.3)) using the results from the following section.

6.6 The Covariance Between a Random Function and its Derivatives

The following useful stochastic process result (Loève (1963), pp. 136-7) provides a way in which we can calculate the variances and covariances necessary to compare the derivative approximation methods for our random function:

Useful Result (Loeve (1963), p. 137):

For two points s and t , if $\Gamma_f(s, t) = \text{Cov}(f(s), f(t))$ is the covariance of a random function f , then f' exists and

$$\text{Cov}(f(s), f'(t)) = \frac{\partial}{\partial t} \Gamma_f(s, t),$$

$$\Gamma_{f'}(s, t) = \frac{\partial^2}{\partial s \partial t} \Gamma_f(s, t),$$

and in general,

$$\text{Cov}(f^{(n)}(s), f^{(m)}(t)) = \frac{\partial^{n+m}}{\partial s^n \partial t^m} \Gamma(s, t).$$

For our random function, we define Γ_f to be covariance function (6.2) and calculate the required derivatives to use in the covariance functions just presented for design points and exact derivatives.

We previously described how to calculate the covariance between two function evaluations in Section (6.4). However, to we present those calculations again here using the notation from the above useful result.

To calculate the covariance between two evaluations of a random function $f(\mathbf{x})$, define

$$\text{Cov}(f(\mathbf{s}), f(\mathbf{t})) = \Gamma_f(\mathbf{s}, \mathbf{t}) = \sigma^2 \exp \left\{ - \sum_{i=1}^d a_i^2 (s_i - t_i)^2 \right\} \quad (6.5)$$

$$\text{and} \quad \text{Var}(f(\mathbf{t})) = \Gamma_f(\mathbf{t}, \mathbf{t}) = \sigma^2, \quad (6.6)$$

where $\mathbf{s}, \mathbf{t} \in \mathbb{R}^d$ are the evaluation points, $f : \mathbb{R}^d \mapsto \mathbb{R}$, and the a_i represent shape parameters. The scale parameter σ^2 is included above for completeness; however, in practice we let $\sigma^2 = 1$ for implementing these results and those that follow, without loss of generality.

Next, for our first application of the stochastic process result, we illustrate calculating the covariance between a function evaluation at $\mathbf{s}$ and the first derivative in the k^{th} direction of the function evaluated at $\mathbf{t}$:

$$\begin{aligned} \text{Cov} \left(f(\mathbf{s}), \frac{\partial}{\partial t_k} f(\mathbf{t}) \right) &= \frac{\partial}{\partial t_k} \Gamma_f(\mathbf{s}, \mathbf{t}) \\ &= 2 a_k^2 (s_k - t_k) \exp \left\{ - \sum_{i=1}^p a_i^2 (s_i - t_i)^2 \right\}. \end{aligned} \quad (6.7)$$

Also, the variance of the first derivative of the function in the k^{th} direction evaluated at $\mathbf{t}$ is

$$\begin{aligned} \text{Var} \left(\frac{\partial}{\partial t_k} f(\mathbf{t}) \right) &= \frac{\partial^2}{\partial s_k \partial t_k} \Gamma_f(\mathbf{s}, \mathbf{t}) \Big|_{\mathbf{s} = \mathbf{t}} \\ &= 2 a_k^2. \end{aligned} \quad (6.8)$$

Given these results, we can now finish computing all of the remaining elements in Σ and the calculations for $\text{Cov}\{\mathbf{z}'\mathbf{y}\}$ in (6.4) to evaluate any of the first derivative approximations. Additional calculations for the second and mixed derivatives are similar, and the necessary derivatives are given in Appendix A.

6.6.1 A Sidenote on Design Scale

As was mentioned in Sections (5.1) and (5.3), the Spendly and RSM approaches are both based on statistical designs that select the function evaluation locations. Even the Del12F approach can be represented in a design matrix notation. Also mentioned earlier was the fact that the least squares analysis of the Del12F design produces the same derivative approximations as those used in the Del12F procedure for the second and mixed derivatives.

Since each method can be represented as a type of designed experiment, it is interesting to compare their sampling patterns in this light. The simplex designs are upper triangular with design points at 0, $h/2$, and h , where the Del12F design points are located at $-h$, 0, and h , in each direction. Only the RSM designs sample points further than a distance of h from the origin, placing evaluations at $-5h/4$, $-h$, 0, h , and $5h/4$ in each direction. Sample point allocation plays an important role in designed experiments as it affects the variance of the estimated model coefficients (even before the data are collected). Therefore, a different scaling multiplier c may yield derivative estimates with better (or worse) variances. Finding this value of c for each of the methods we are considering ensures that no method is arbitrarily penalized simply because of its scaling. So for example, if the Spendly method gives more precise approximations with evaluation points at 0, $ch/2$, and ch , it would be interesting to find this value of c and then use it in making comparisons with the other methods and their optimized value of c .

Some of our results for Del12f suggest that a considerably larger step size may give better precision. However, even though we seem capable of calculating optimal design scalings, the fact remains that these derivative approximation methods are used in practice

“as is” (i.e. without adjusted scaling), and using the model to predict c would take us away from what is commonly practiced. Since we wish to compare the derivative approximations as they are currently in use, we will leave $c = 1$ for each method. Furthermore, this may be a moot point anyway as in practice we will not know the value of a anyway, so we cannot use it to guide our choice of c .

6.7 Computing Derivative Approximation Variances

Returning now to computing the derivative approximation variances, we calculate the variances using (6.4) for dimensions ranging from 2 up to 20, examining first, second, and mixed derivatives for each method at two levels of precision (round-off error variance). That is, we define single and double precision calculations as letting $\sigma_r = 1\text{e-}7$ and $\sigma_r = 1\text{e-}16$, respectively, in **D** from eq. (6.1). Also, we define a *problem* as the task of calculating the variance of a particular derivative approximation minus the true derivative value for a first, second, or mixed derivative using (6.4), for a given precision level and dimension, with $a = c = \sigma^2 = 1$. For a particular problem, we will calculate the average variance of the derivative over all the different directions. This is necessary since, unlike the Del12F and Spendly methods, the RSM designs are not symmetric in their arguments. For example, in 5 dimensions, we obtain 5 first derivative approximation variances (one for each of the 5 directions) which are averaged, and the square root of this average then used for comparison.

For small problems in 2 and 3 dimensions, the variances were calculated analytically and these results were used to find the optimal (variance minimizing) step size, h . Table (6.1) summarizes these analytical results, showing that the optimal step sizes are sim-

ilar to those suggested by Dennis and Schnabel (1983) discussed in Section (4.5). This fact gives us confidence that the random function model is relevant to what is done in practice, and supports using it to compare the methods.

In the summary table, three groups of derivative variances and step sizes are shown: one group for three finite difference formulas, one group for two Spendly derivative formulas, and one for a few small- n derivative approximations. In the third group, one derivative is labelled DEL12F-LS representing the approximation obtained by using least-squares on the DEL12F design. This result is an alternative to the central difference. The optimal step-sizes were calculated from the variances and give insight into the balance between function noise (a) and computer round-off error (σ_r^2) (recall that we set $\sigma = 1$ even though it is shown in these results). For example, fixing the level of round-off error (selecting double precision, say) and increasing the function noise clearly requires a smaller step-size.

These analytical derivations worked well enough for small dimensional problems, but soon became unwieldy in higher dimensions. We then used FORTRAN 95 to obtain the higher-dimension variances numerically. The computing involved is somewhat complex and we developed a strategy to help keep track of all the necessary results and calculations. We will illustrate this strategy as it applies to first derivative calculations, and these calculations can then be used as a template for computing second and mixed derivative results.

As described earlier, each of the approximation methods we consider here can be represented as if based on an experimental design. We let the matrix $\mathbf{X}$ represent the design matrix with a vector of 1s (an intercept column) appended on the left. For example, the design matrix for a 2-dimensional problem would have 6 columns: an intercept(1s), x_1 , x_2 ,

x_1^2 , x_2^2 , and $x_1 * x_2$. The rows under x_1 and x_2 contain the coordinates of the abscissa where the particular derivative method will sample the function. The remaining columns are functions of these two base columns as indicated by their column names.

For d dimensions, let n be the number of points in the design, $\mathbf{t}_i$, $i = 1, \dots, n$, represent the design points (abscissae), and let $\mathbf{t}_0$ represent the point at which we desire the derivative. For convenience, let $p_i = f(\mathbf{t}_i)$ and $p'_k = U'_k(\mathbf{t}_0)$, $k = 1, \dots, d$ (i.e. the first derivative of U in the k^{th} direction). We construct the following partitioned matrix containing all the necessary covariances for our calculations:

$$\begin{bmatrix}
 \begin{array}{cccc|cccc}
 Var(p_1)+\sigma_r^2 & Cov(p_1, p_2) & \cdots & Cov(p_1, p_n) & Cov(p_1, p'_1) & \cdots & & Cov(p_1, p'_d) \\
 Cov(p_2, p_1) & Var(p_2)+\sigma_r^2 & \cdots & Cov(p_2, p_n) & Cov(p_2, p'_1) & \cdots & & Cov(p_2, p'_d) \\
 \vdots & & \ddots & \vdots & \vdots & & & \vdots \\
 Cov(p_n, p_1) & Cov(p_n, p_2) & \cdots & Var(p_n)+\sigma_r^2 & Cov(p_n, p'_1) & \cdots & & Cov(p_n, p'_d)
 \end{array} \\
 \hline
 \begin{array}{cccc|cccc}
 Cov(p'_1, p_1) & Cov(p'_1, p_2) & \cdots & Cov(p'_1, p_n) & Var(p'_1) & \text{huge} & \cdots & \text{huge} \\
 Cov(p'_2, p_1) & Cov(p'_2, p_2) & \cdots & Cov(p'_2, p_n) & \text{huge} & Var(p'_2) & \text{huge} & \cdots \text{ huge} \\
 \vdots & \vdots & \cdots & \vdots & \vdots & & \ddots & \vdots \\
 Cov(p'_d, p_1) & Cov(p'_d, p_2) & \cdots & Cov(p'_d, p_n) & \text{huge} & \cdots & \text{huge} & Var(p'_d)
 \end{array}
 \end{bmatrix} \tag{6.9}$$

With the above matrix available, we can select subsets to form $[\mathbf{\Sigma} + \mathbf{D}]$ from (6.4) and compute the desired variance. The large values (huge) present in the lower right partition of (6.9) function as a “fail-safe” when writing code to extract subsets, and are never intentionally used in any computations. If a misaligned section including one or more of these values were somehow extracted, the calculated result will be large enough to indicate a problem. In theory, these values never appear in the calculations.

Calculations for all d first derivative approximations are accomplished by iteratively extracting the appropriate row and column from the lower-left and upper-right partitions of (6.9), respectively, and appending them to the upper-left partition as the bottom-most row and right-most column. Finally, the lower-right element in $[\mathbf{\Sigma} + \mathbf{D}]$ comes from extracting the appropriate element from the diagonal in the lower-right partition of (6.9).

With the covariance matrix $[\mathbf{\Sigma} + \mathbf{D}]$ assembled, we now only require $\mathbf{z}$ to compute variance (6.4). Recall that $\mathbf{z}$ contains the coefficients for the linear combination of function evaluations minus the exact derivative. Using Modified Gram-Schmidt Orthogonalization (see Monahan, 2001) we can factor $\mathbf{X}$ ($n \times p$) as $\mathbf{X} = \mathbf{QR}$, where $\mathbf{R}$ is upper-triangular and $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}_p$. The design matrix $\mathbf{X}$ is constructed in “coded form” with values ranging from about -1.6 to 1.6, depending on the derivative approximation method. Coding is commonly done to avoid precision problems in least squares calculations (i.e. inverting $(\mathbf{X}^T \mathbf{X})$). However, the derivative approximations are formed using step sizes like $h = \pm(\text{machine epsilon})^{(1/3)}$, not $h \approx \pm 1$, so we really need to scale each column of $\mathbf{X}$ by h or h^2 . This scaling can be written in matrix notation as $\mathbf{XH}$, where $\mathbf{H}$ is a diagonal matrix with the proper scale factors for the columns of $\mathbf{X}$ on its diagonal. Now the coefficients for the linear combinations forming the least-squares solution vector are found in the rows of $\mathbf{H}^{-1} \mathbf{R}^{-1} \mathbf{Q}^T$ ($p \times n$) (possibly more familiar as $\mathbf{H}^{-1} (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$). Based on the ordering of columns in $\mathbf{X}$ (recall the 2-dim example: intercept(1s), x_1 , x_2 , x_1^2 , x_2^2 , and $x_1 * x_2$), we skip the first row of $\mathbf{R}^{-1} \mathbf{Q}^T$ and then extract the next d rows to iteratively use as $\mathbf{z}$ in (6.4) (after appending -1 to $\mathbf{z}$).

At this point, we expected to obtain the variances of interest. However, upon switching to floating-point arithmetic, we immediately encountered precision problems calculating the quadratic form (6.4). These errors (a majority at least) come from computing $e^{-\lambda}$ when λ is very small. Since we require computing this particular function a very large number of times, the errors accumulate to produce unusable results.

To understand the source of the errors, we considered the familiar Taylor-series expansion of $e^{-\lambda} \approx 1 - \lambda + \lambda^2/2$. If $-\lambda = -a^2h^2$ (two points are h units apart, say) then the relative error for this computation is $(1 - a^2h^2 + a^4h^4/2)(1 + \epsilon)$, which is on the scale of 1 (ϵ is the machine epsilon). On the other hand, if the leading 1 in the expansion could be removed, the relative error would reduce to $(-a^2h^2 + a^4h^4/2)(1 + \epsilon)$, which is on the scale of h^2 and considerably smaller.

Now, since $\mathbf{X}$ contains an initial column of 1s (the intercept column), the columns in $\mathbf{Q}$ from the Gram-Schmidt factorization of $\mathbf{X}$ (excluding the first column here and in what follows) are orthogonal to $\mathbf{1}$ ($\mathbf{Q}$ is orthonormal). Therefore, if $\mathbf{z}$ is a column from $\mathbf{Q}$, then $\mathbf{z}^\top \mathbf{1} = 0$.

We can now reexpress (6.4) in an alternate form as follows:

$$\begin{aligned}
 \text{Var}(\mathbf{z}^\top \mathbf{y}) &= \mathbf{z}^\top \text{Cov}(\mathbf{y}) \mathbf{z} \\
 &= \mathbf{z}^\top (\mathbf{\Sigma} + \mathbf{D}) \mathbf{z} - \mathbf{0} \\
 &= \mathbf{z}^\top (\mathbf{\Sigma} + \mathbf{D}) \mathbf{z} - \mathbf{z}^\top \begin{bmatrix} \mathbf{1}_n \mathbf{1}_n^\top & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} \mathbf{z} \\
 &= \mathbf{z}^\top \left(\mathbf{\Sigma} + \mathbf{D} - \begin{bmatrix} \mathbf{1}_n \mathbf{1}_n^\top & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} \right) \mathbf{z}.
 \end{aligned} \tag{6.10}$$

The exponential functions in the upper-left ($n \times n$) partition of Σ can now be written as

$$\begin{aligned}
 e^{-\lambda} - 1 &= \left(1 - \lambda + \lambda^2/2 - \lambda^3/3! + \lambda^4/4! - \dots + (-1)^k \lambda^k/k! + \dots \right) - 1 \\
 &\approx -\lambda + \lambda^2/2 - \lambda^3/3! + \lambda^4/4! \text{ which is computed via Horner's rule:} \\
 &\approx -(1.d0 - (1.d0 - (1.d0 - \lambda/4.d0) * \lambda/3.d0) * \lambda/2.d0) * \lambda.
 \end{aligned}$$

The final expression above is written to maximize precision and is an exact line taken from our FORTRAN 95 code. In addition to modifying the exponential functions as described, we also used multiple precision (16-byte real variables) in FORTRAN 95. Implementing these two solutions, we computed and examined the derivative approximation variances with $\sigma_r^2 = (1e-7)^2$ (i.e. our model definition of single precision round-off error). These results seemed much more reasonable and it was clear that the computer precision difficulties were no longer affecting our results. The single precision results then served as our test patterns for the double precision results when $\sigma_r^2 = (1d-16)^2$. Since the general patterns remained the same and only the magnitude of the results changed, we are confident that the double precision results are sound as well. The results from our computations are presented graphically in the following section where we finally compare the derivative approximations.

6.8 Derivative Comparison Results – Which Approximation is Better?

6.8.1 Recommendations

With the derivative variances available, a few recommendations can be made. The derivative root average variance by dimension plots show that the mixed and second derivatives are approximated with the least precision by all three methods. The RSM approach excels at the second derivative, but suffers somewhat when approximating mixed derivatives. For first derivatives, the RSM approach works well for problems up to 10 dimensions, but suffers dramatically for 11 or more dimensions. Still, this is of less concern as the first derivatives are estimated with much higher precision than the second or mixed derivatives. Furthermore, given the option of diagnostic feedback with regard to scaling and a reduced sample size compared to the Del12F approach, the RSM designs seem reasonably attractive. The Del12F method performs well overall, especially for estimating mixed derivatives. This precision, however, comes at the cost of many function evaluations relative to the other methods. Though utilizing the smallest sample size, the Spendly approach tends to offer the least precision overall, most notably when estimating second derivatives.

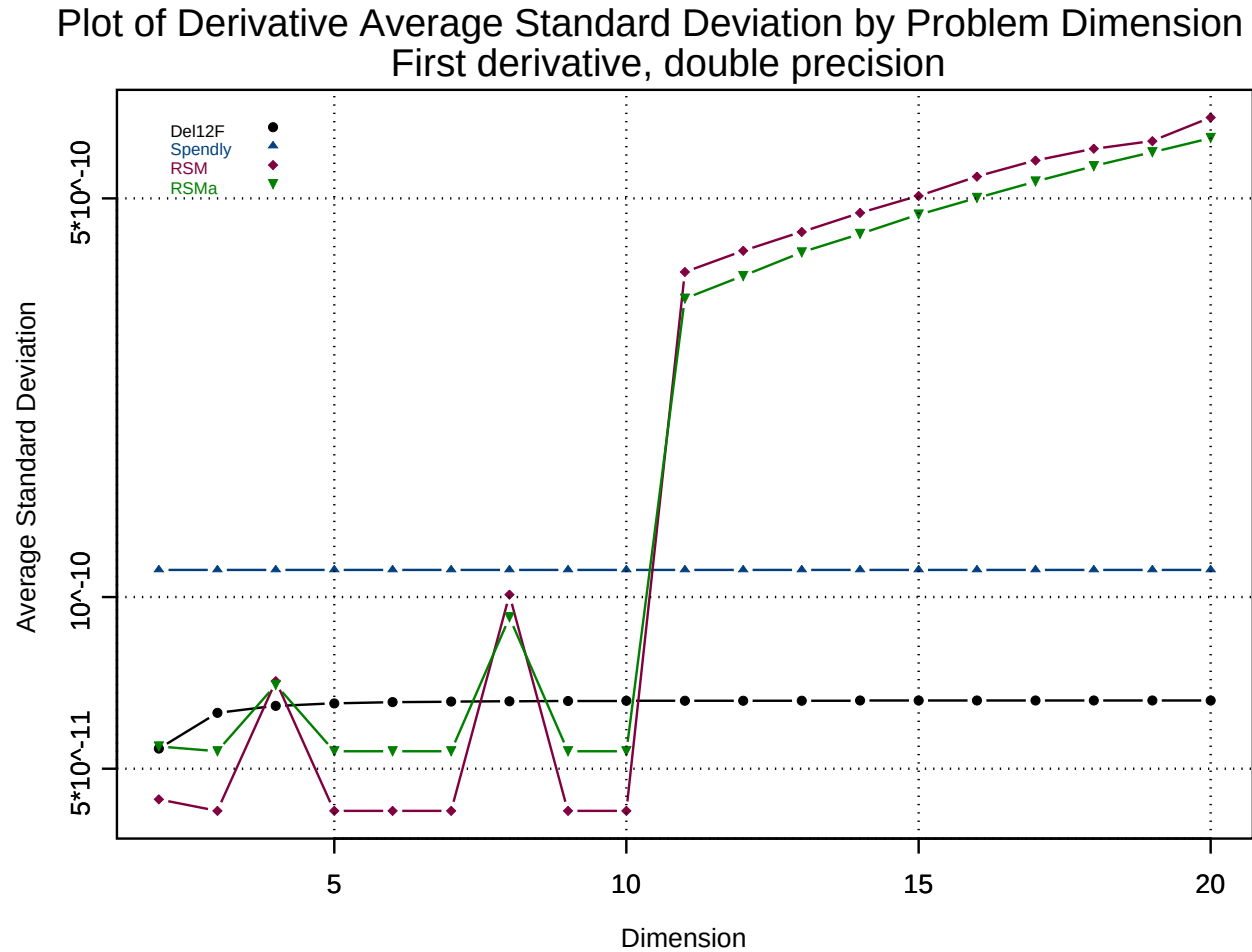


Figure 6.6: Plot of Root Average Derivative Approximation Variance by Problem Dimension for First Derivatives in Double Precision.

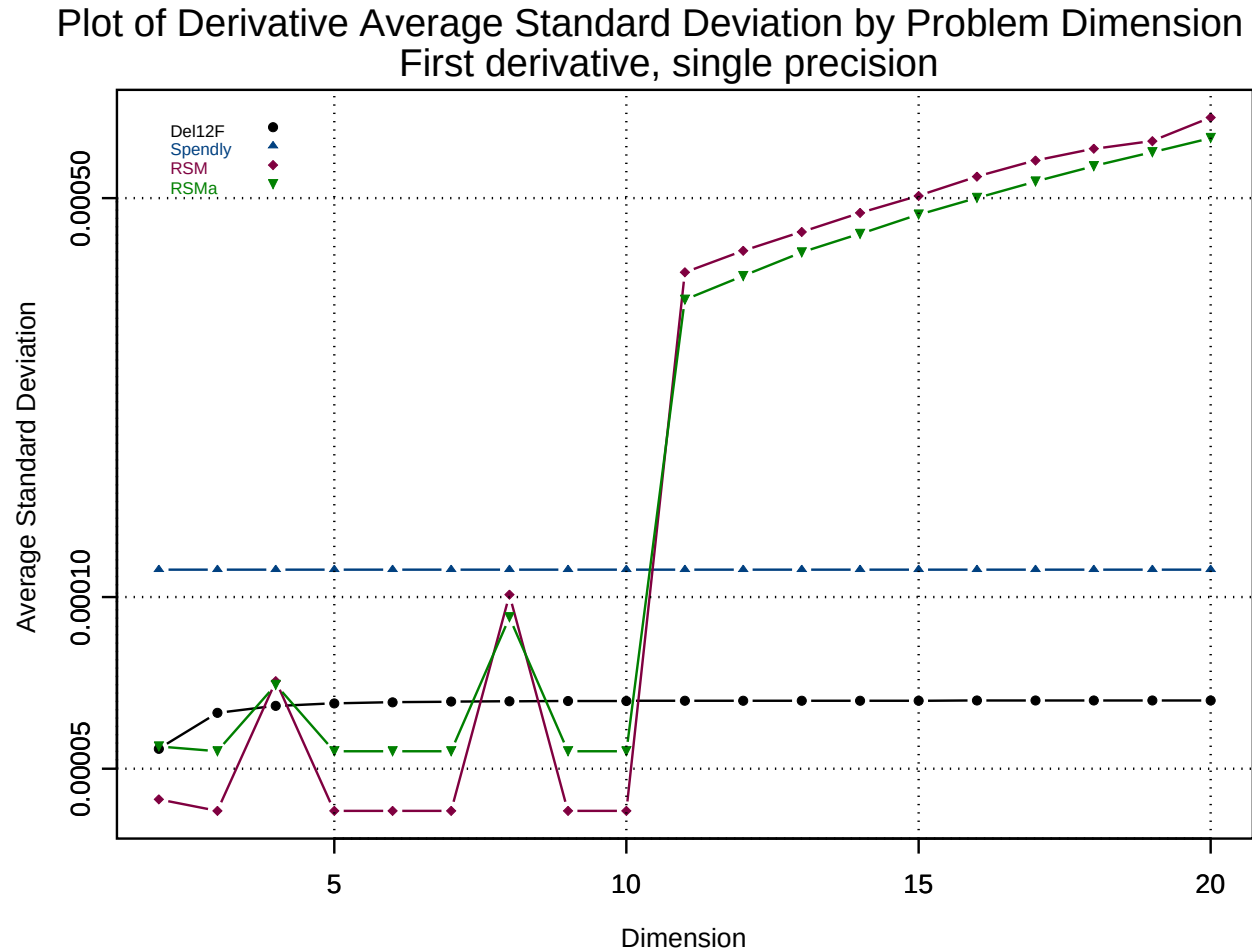


Figure 6.7: Plot of Root Average Derivative Approximation Variance by Problem Dimension for First Derivatives in Single Precision.

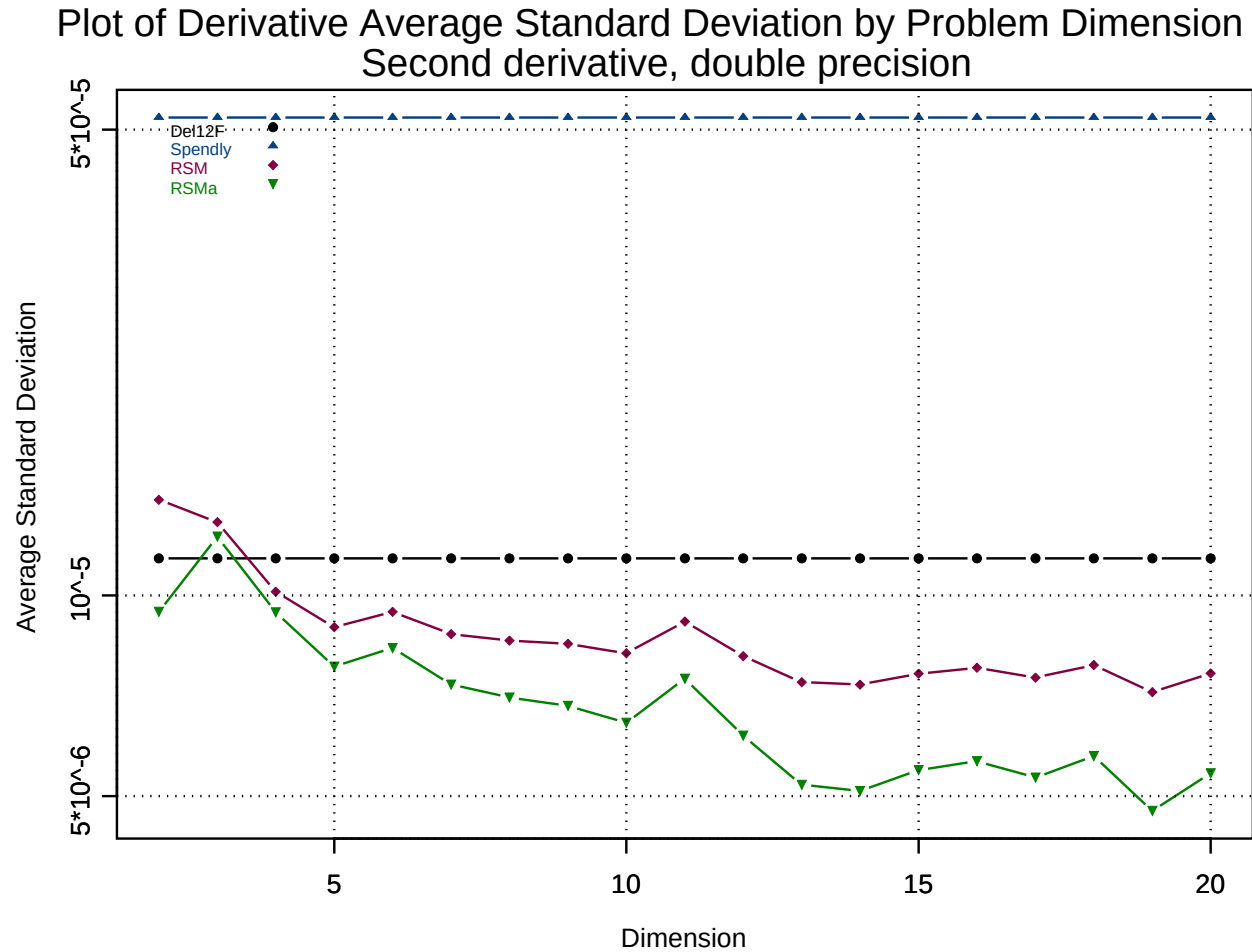


Figure 6.8: Plot of Root Average Derivative Approximation Variance by Problem Dimension for Second Derivatives in Double Precision.

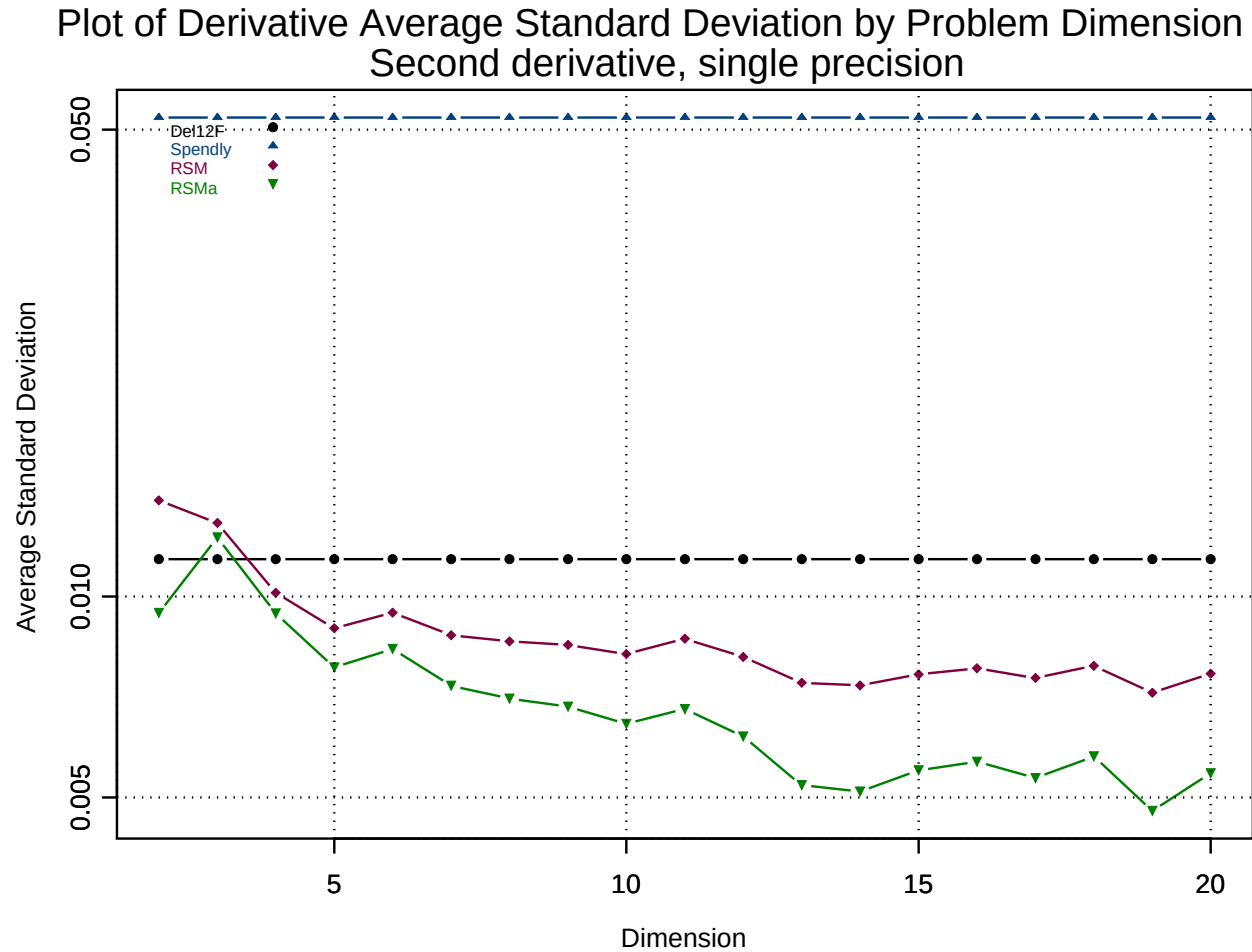


Figure 6.9: Plot of Root Average Derivative Approximation Variance by Problem Dimension for Second Derivatives in Single Precision.

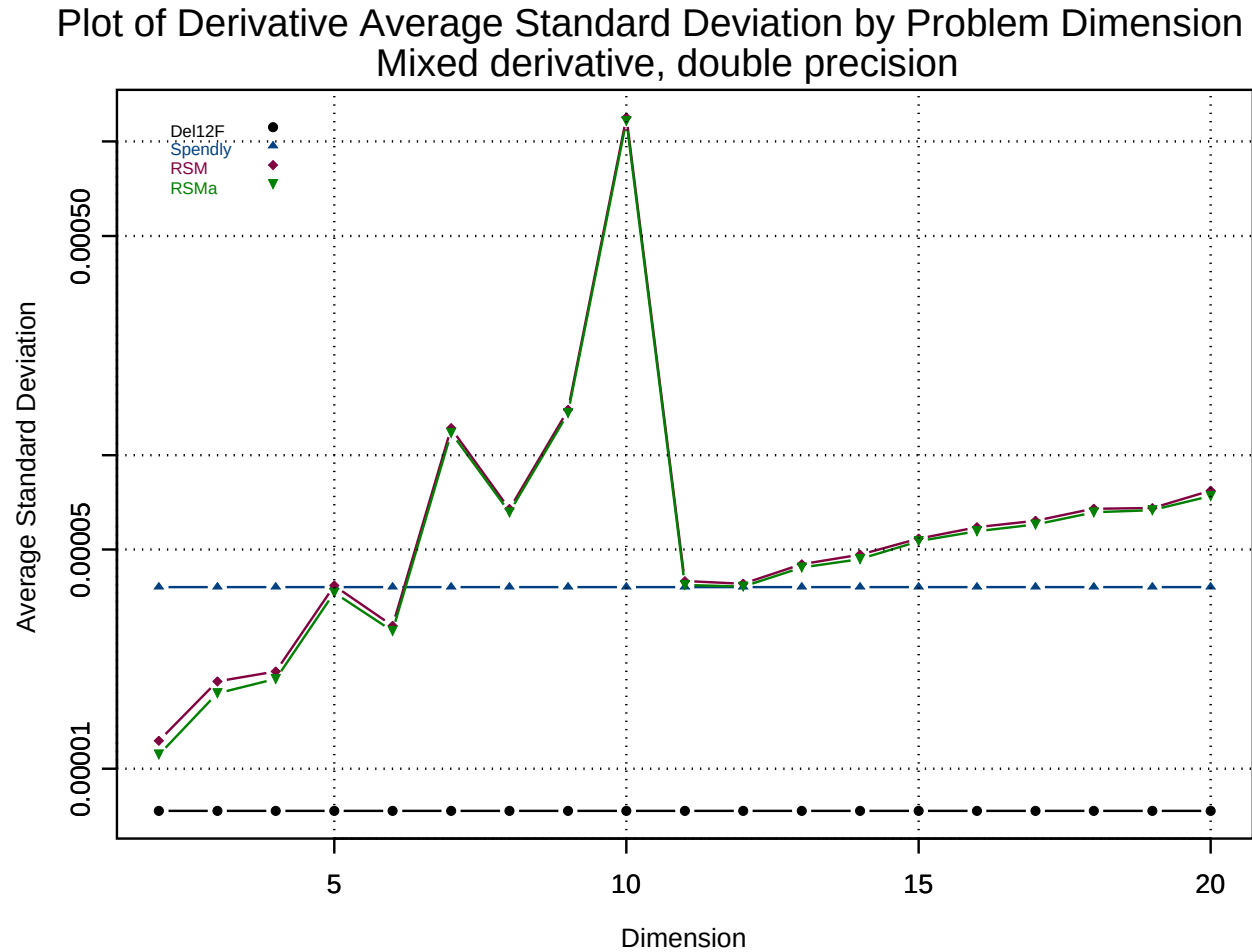


Figure 6.10: Plot of Root Average Derivative Approximation Variance by Problem Dimension for Mixed Derivatives in Double Precision.

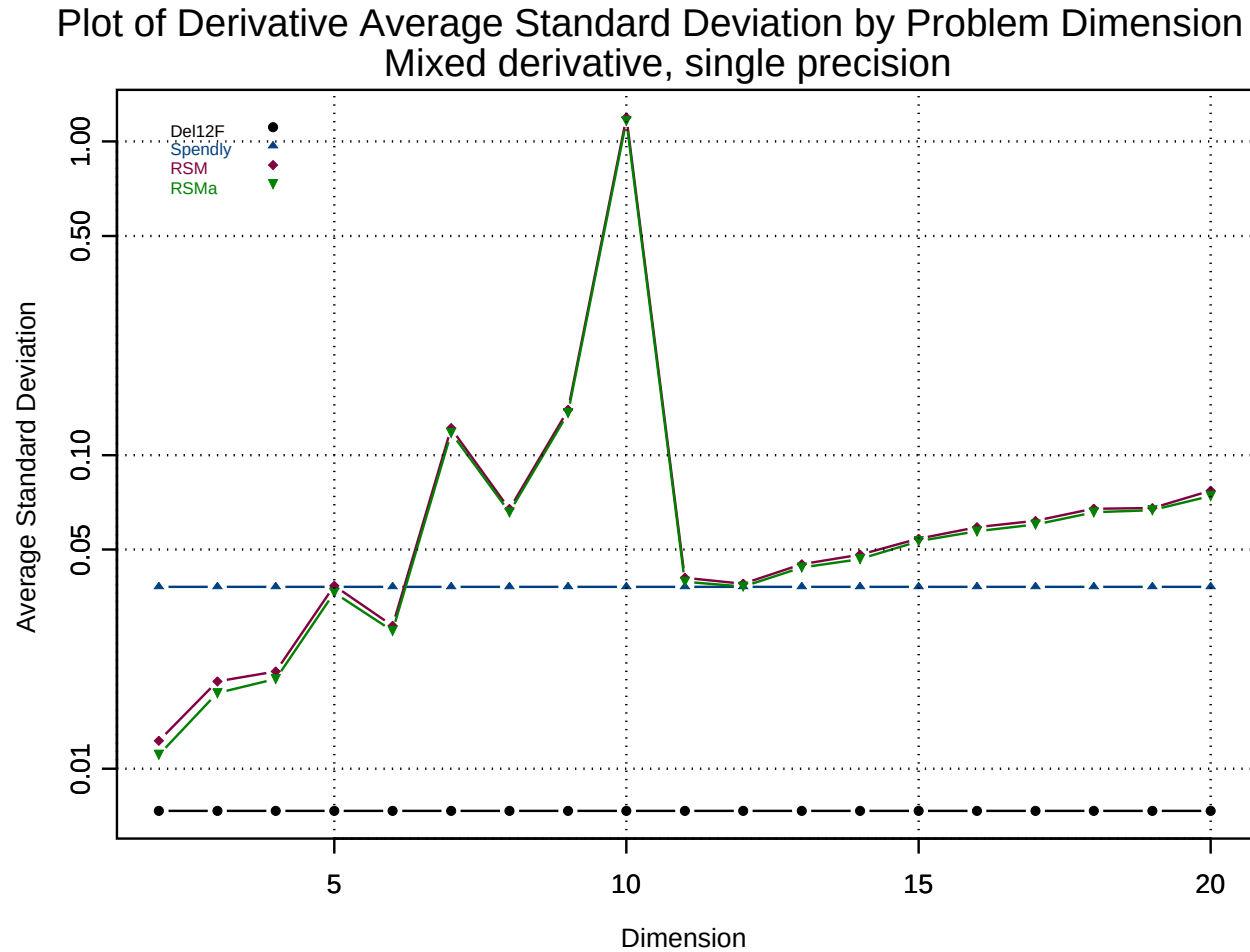


Figure 6.11: Plot of Root Average Derivative Approximation Variance by Problem Dimension for Mixed Derivatives in Single Precision.

6.9 Diagnostics

As mentioned previously in Section (5.3.3), the base or *inner* RSM designs are augmented with additional *star* points to form an *outer* design. The *star* points are intended to provide a check for model goodness-of-fit. Put another way, they can provide diagnostic feedback on variable scaling for the problem of approximating derivatives. For a general overview, we start by first fitting a quadratic model using the inner design points, and then employing the estimated model to predict the function evaluations at the augmented star points. We then compare the function evaluations at the axial points to the predictions and evaluate the model fit. For a given derivative approximation, this comparison is made in each direction to provide scaling information on each variable involved. If the variable scaling is such that we are fitting the model in a quadratic region, the predictions should reasonably match the corresponding function evaluations.

6.9.1 Forecast Variances

As done earlier with the derivative approximations, the diagnostic feedback comes by making use of the design point covariance matrix in calculating variances of the design point predictions minus the true function evaluations. These variances are then used to form a test for scaling problems in the design variables.

We begin by obtaining another QR factorization of the design matrix, this time using the Householder algorithm (see Stewart (1973)):

$$\mathbf{U}_p \cdots \mathbf{U}_1 (\mathbf{X} | \mathbf{I}_{n_1}) = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{Q}_1 \\ \hline \mathbf{0} & \mathbf{Q}_2 \end{array} \right],$$

where n_1 is the number of function evaluations required for the *inner design* and p is the number of columns in $\mathbf{X}$.

The matrix $\mathbf{Q}_2$ contains the function evaluation linear combination(s) that correspond to the sums of squares for error in a multiple regression (i.e. $\text{SSE} = \|\mathbf{Q}_2 \mathbf{y}\|^2$). To test for scaling problems in a particular direction, we construct a matrix $\mathbf{B}$ by augmenting $\mathbf{Q}_2$ with the two linear combinations that predict the star points in that direction minus the actual function evaluations. We also augment $\mathbf{y}$ with the two star-point evaluations to give:

$$\mathbf{B} = \left[\begin{array}{c|cc} \mathbf{Q}_2 & 0 & 0 \\ \hline \ell_1^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top & -1 & 0 \\ \ell_2^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top & 0 & -1 \end{array} \right], \quad \mathbf{y}^* = \left[\begin{array}{c} \mathbf{y} \\ y_{\ell_1} \\ y_{\ell_2} \end{array} \right], \quad (6.11)$$

where the ℓ_i are linear combinations of the regression coefficients $\mathbf{b} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$. Under regular regression assumptions,

$$\mathbf{B} \mathbf{y}^* \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{\Sigma}_{By}), \quad \text{and} \quad \boldsymbol{\Sigma}_{By} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_2 + \mathbf{X}_0 (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}_0^\top \end{bmatrix},$$

where

$$\mathbf{X}_0^\top = \begin{bmatrix} \ell_1 \\ \ell_2 \end{bmatrix}.$$

Now, we plan to use the elements of $\mathbf{B} \mathbf{y}^*$ to calculate a pseudo F-statistic for testing scaling variability against the mean square error (MSE). By scaling variability, we refer to the variability associated with the star-point predictions minus the actual function evaluations

at the star points. If the variability associated with these two linear combinations is significantly greater than MSE, we will conclude that there is a problem with the variable scaling in the direction of the axis on which the star-points are located.

Since we are operating outside the standard regression assumptions, we “standardize” $\mathbf{By}^*$ using the elements from the lower right corner of $\mathbf{\Sigma}_{By}$ in an effort to diagonalize the covariance matrix for $\mathbf{By}^*$. This is done with the hope that the ratio of sums of squares based on elements of $\mathbf{By}^*$ (eventually used to form the pseudo F-statistic) will have as little correlation as possible.

Specifically, using a Cholesky root (see Stewart (1973)), we can rewrite $\mathbf{\Sigma}_{By}$ as

$$\mathbf{\Sigma}_{By} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{L L}^\top \end{bmatrix},$$

and use $\mathbf{L}^{-1}$ to rescale $\mathbf{B}$:

$$\mathbf{B}^* = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{-1} \end{bmatrix} \mathbf{B}.$$

Finally, using the elements of

$$\mathbf{B}^* \mathbf{y}^* = \begin{bmatrix} \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix},$$

we calculate a pseudo F-statistic as:

$$F = \frac{\|\mathbf{z}_3\|^2/2}{\|\mathbf{z}_2\|^2/(n-p)}, \quad (6.12)$$

with 2 and $n - p$ degrees of freedom, where n is the number of function evaluations in the *inner* design and p is number of columns in the design matrix $\mathbf{X}$.

Pattern	Corresponding a -vector	Explanation
A	$a_i = 1$	no scaling errors
B	$a_1 = 10, a_3 = .5, a_i = 1$ otherwise	One dimension scaled badly, another slightly under-scaled
C	$a_d = 10, a_i = 1$ otherwise	Highest dimension scaled badly
D	$a_1 = 2, a_3 = 2, a_i = 1$ otherwise	Two dimensions slightly off-scale

Table 6.2: Table of Simulated Scaling Patterns

6.9.2 Diagnostic Simulations

In order to evaluate the pseudo F-statistic (6.12), we create four patterns representing different variable scaling scenarios and calculate the test statistic for each. These test statistics are compared the 90th percentile from an F distribution with the appropriate degrees of freedom and a count of the number exceeding this critical F-value is taken. This process is repeated for dimensions 3, 6, 9, 12, 15, and 18. The test cases are constructed using covariance function (6.2) with the values for $\mathbf{a}$ depending on the pattern. Table (6.2) describes the values for $\mathbf{a}$ used in each pattern.

For example, in 5 dimensions, pattern C tests the sensitivity for detecting scaling problems in only one direction. The corresponding $\mathbf{a}$ used with (6.2) is $\mathbf{a} = (1, 1, 1, 1, 10)^T$.

For both single and double precision, we ran 999 iterations of each test case and counted the number of pseudo F-statistics exceeding the appropriate critical F with $\alpha = 0.1$. Figures (6.12) through (6.19) below illustrate the results.

In summary, we observe good power to detect situations where the variable scaling is off by a factor of 10 or more, but that power seems to diminish as the dimensions of the problem increases. Also we note that the test seems to be somewhat conservative. In

pattern **A** the variables are all correctly scaled in each direction and we see far fewer than 10% of the test statistics falling above the critical value. That is, the size of the test is less than α .

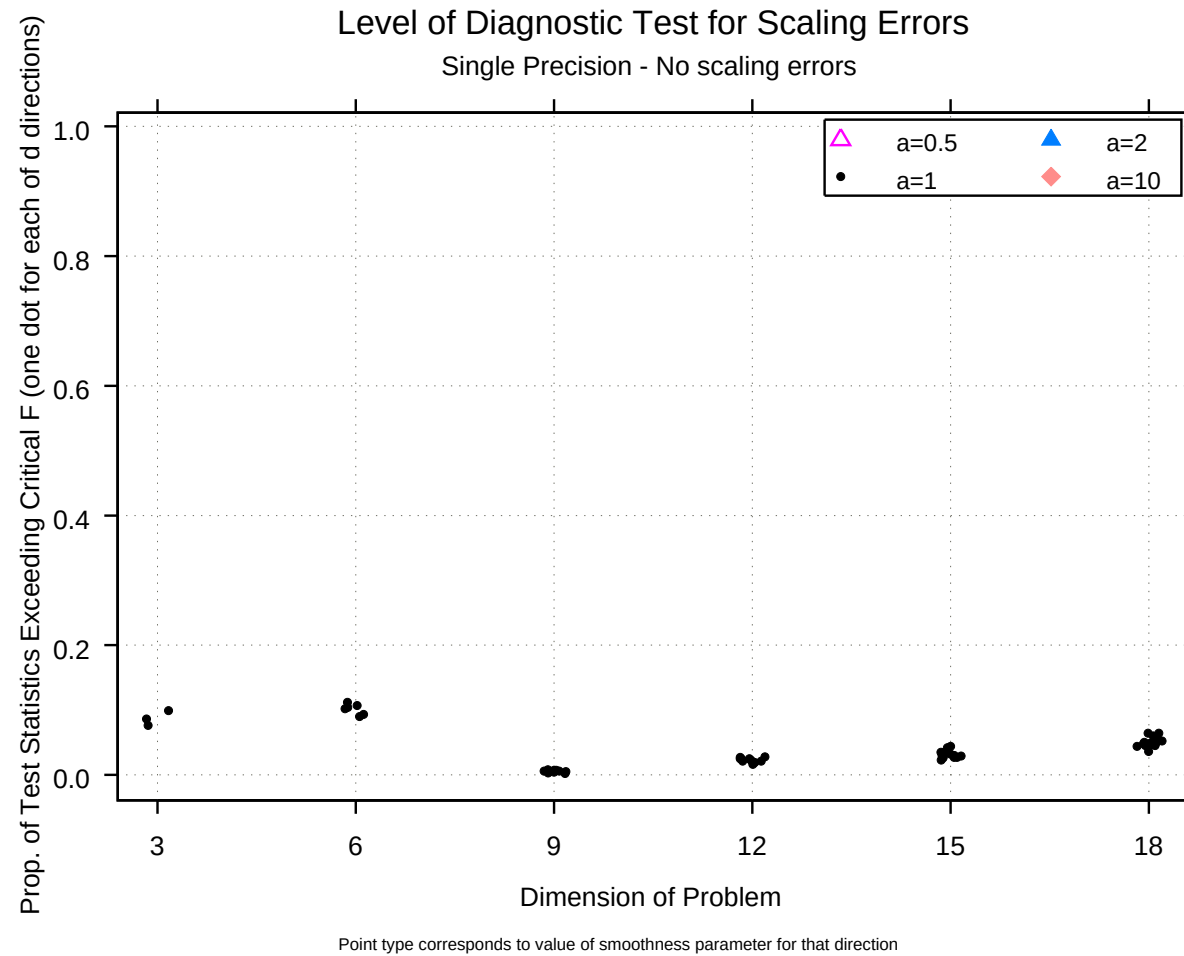


Figure 6.12: Plot of Diagnostic Test Level for Scaling Errors Using Single Precision – No Scaling Errors.

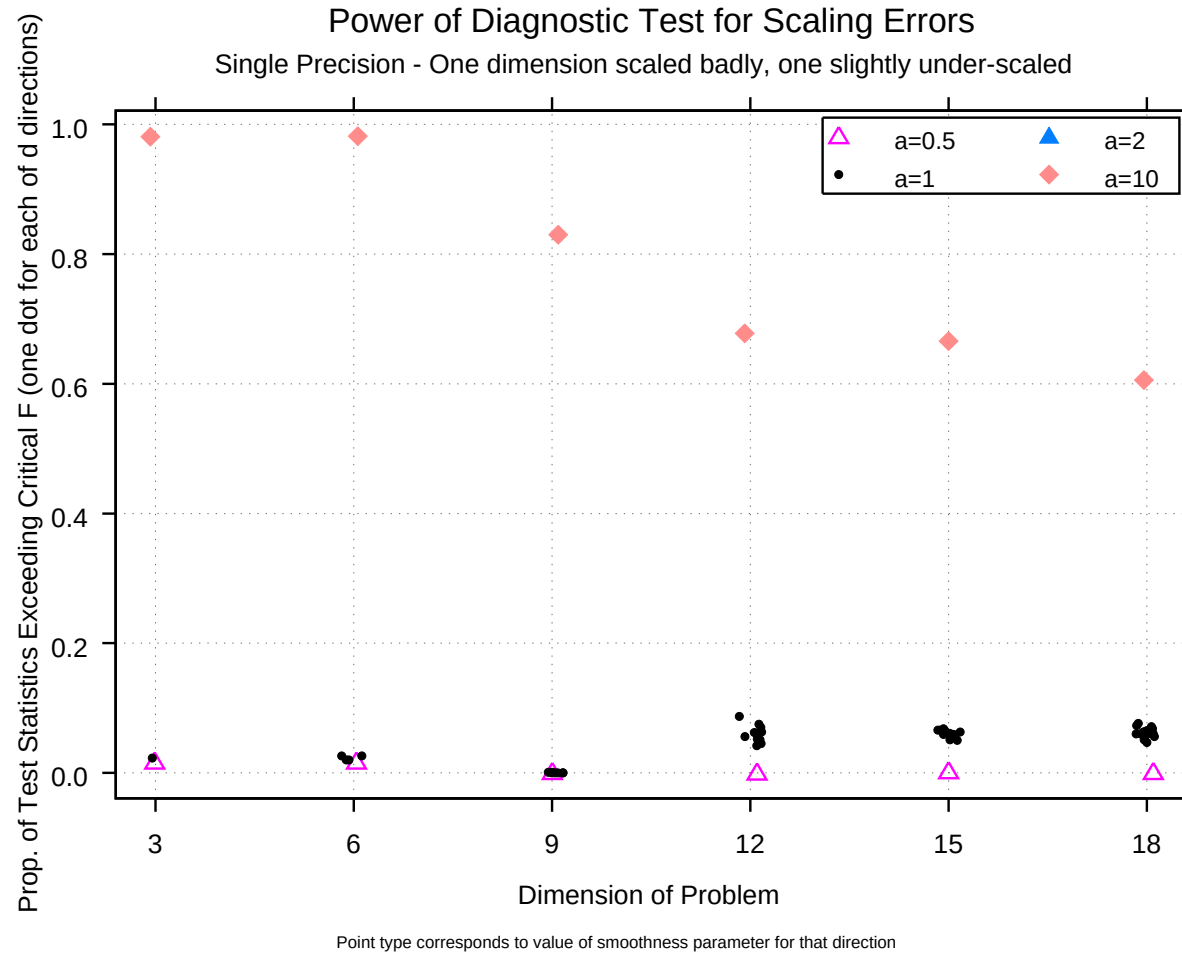


Figure 6.13: Plot of Diagnostic Test Power for Scaling Errors Using Single Precision – One Dimension Scaled Badly.

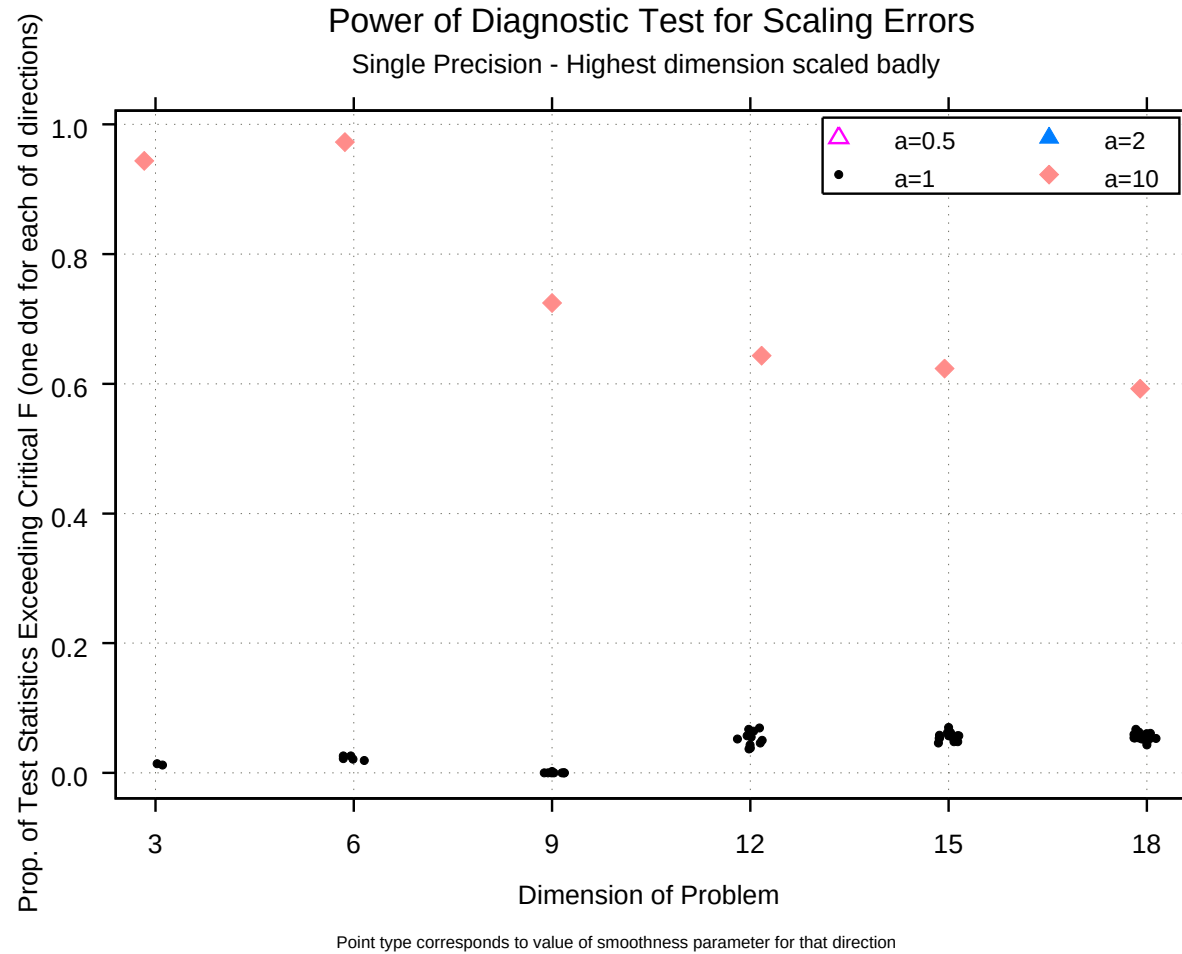


Figure 6.14: Plot of Diagnostic Test Power for Scaling Errors Using Single Precision – Highest Dimension Scaled Badly.

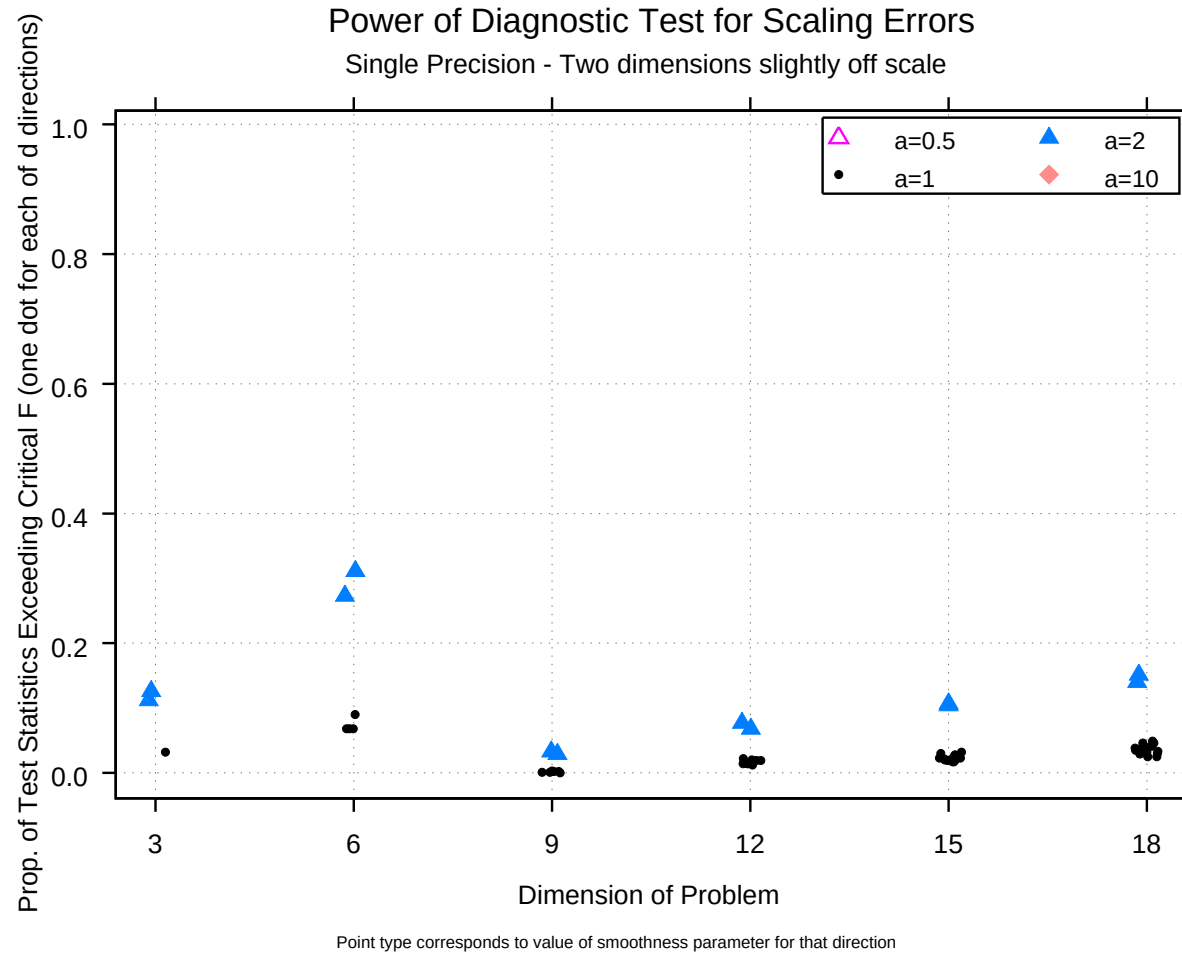


Figure 6.15: Plot of Diagnostic Test Power for Scaling Errors Using Single Precision – Two Dimensions Slightly Off-Scale.

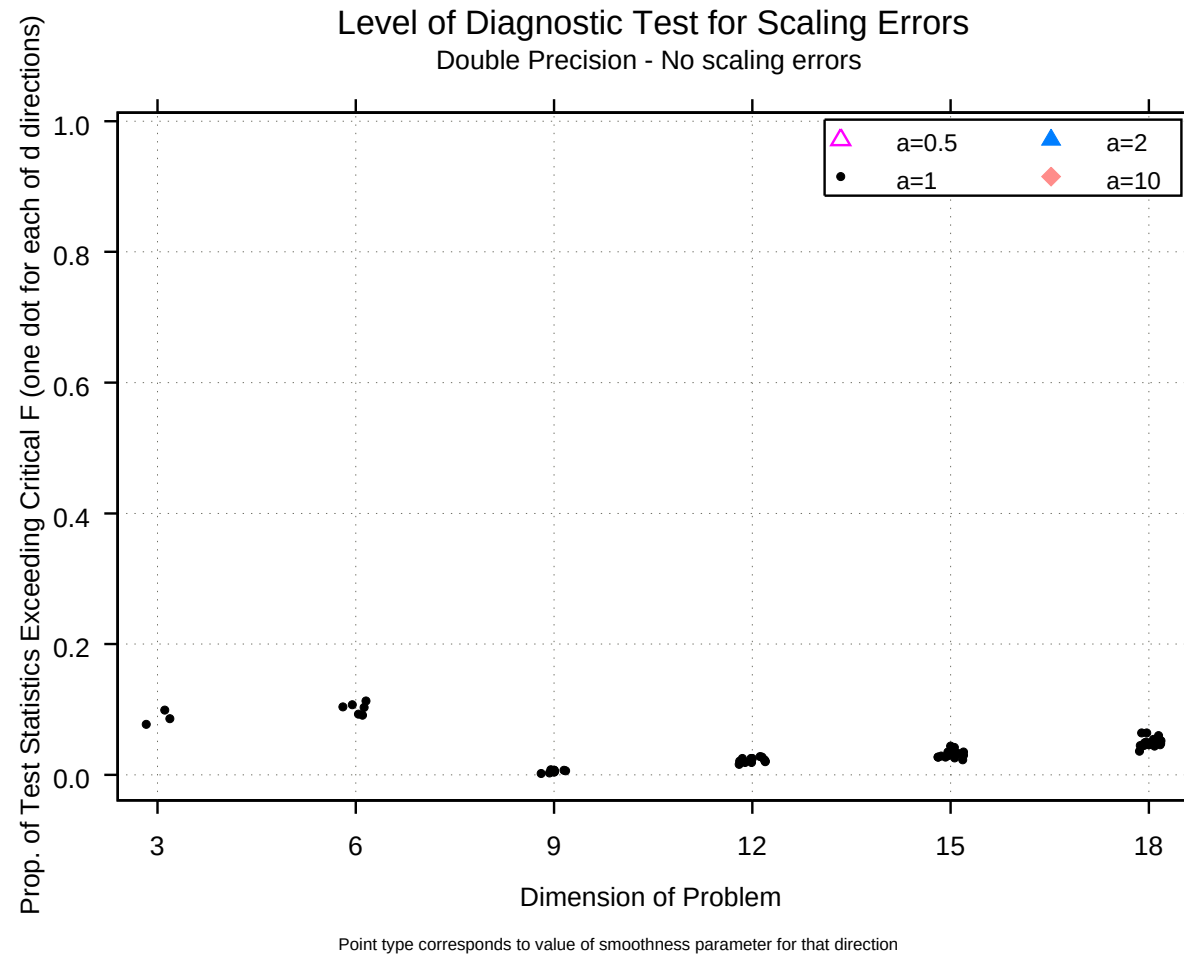


Figure 6.16: Plot of Diagnostic Test Level for Scaling Errors Using Double Precision – No Scaling Errors.

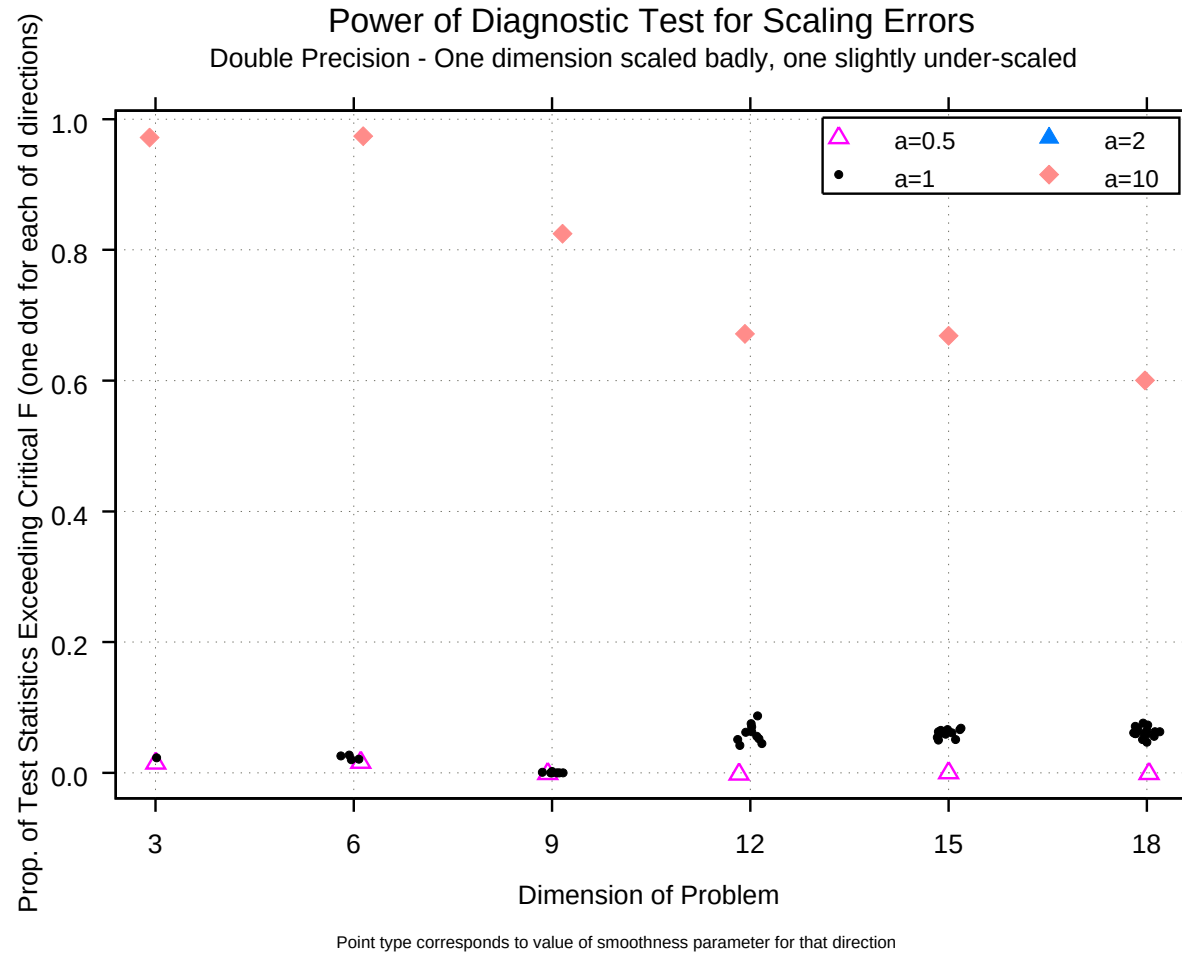


Figure 6.17: Plot of Diagnostic Test Power for Scaling Errors Using Double Precision – One Dimension Scaled Badly.

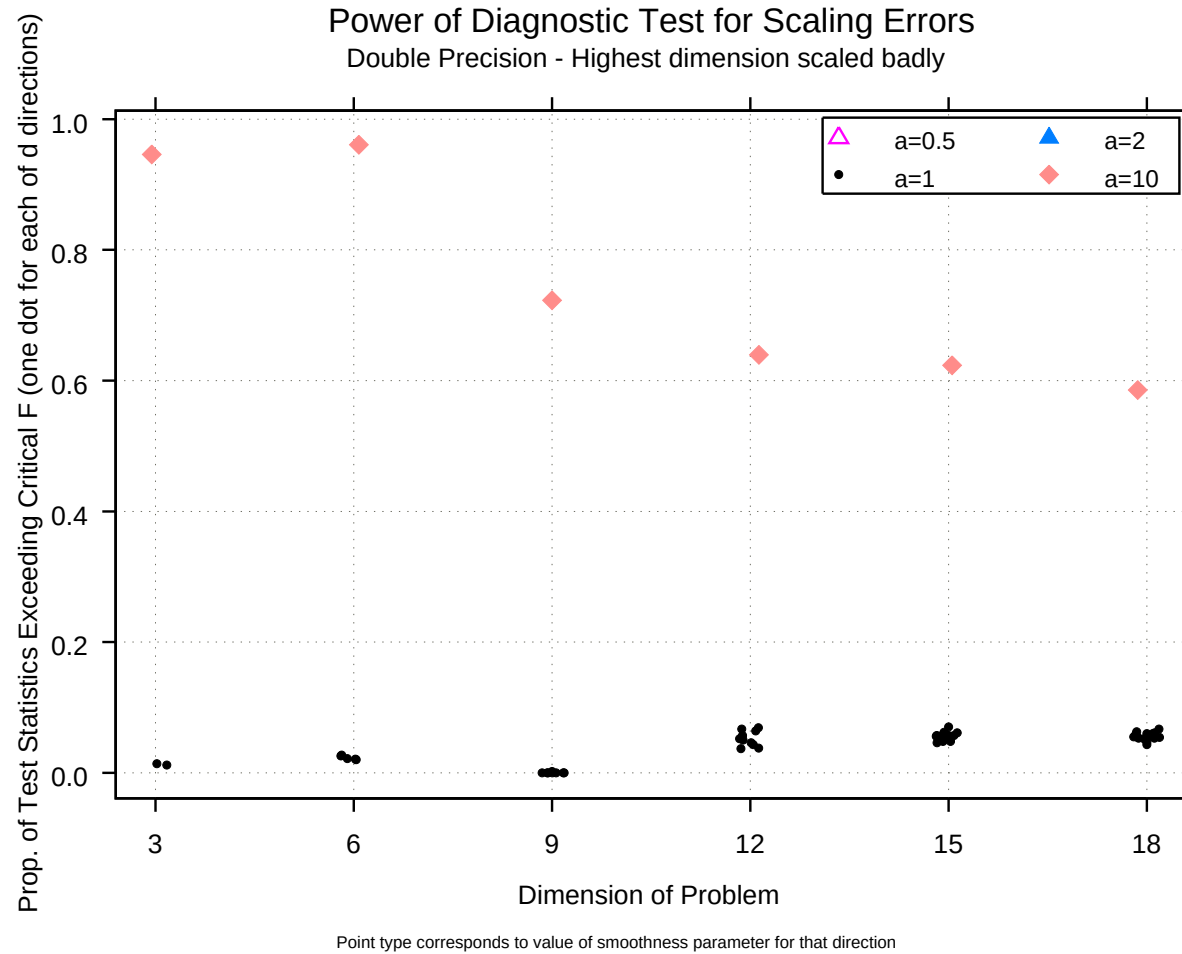


Figure 6.18: Plot of Diagnostic Test Power for Scaling Errors Using Double Precision – Highest Dimension Scaled Badly.

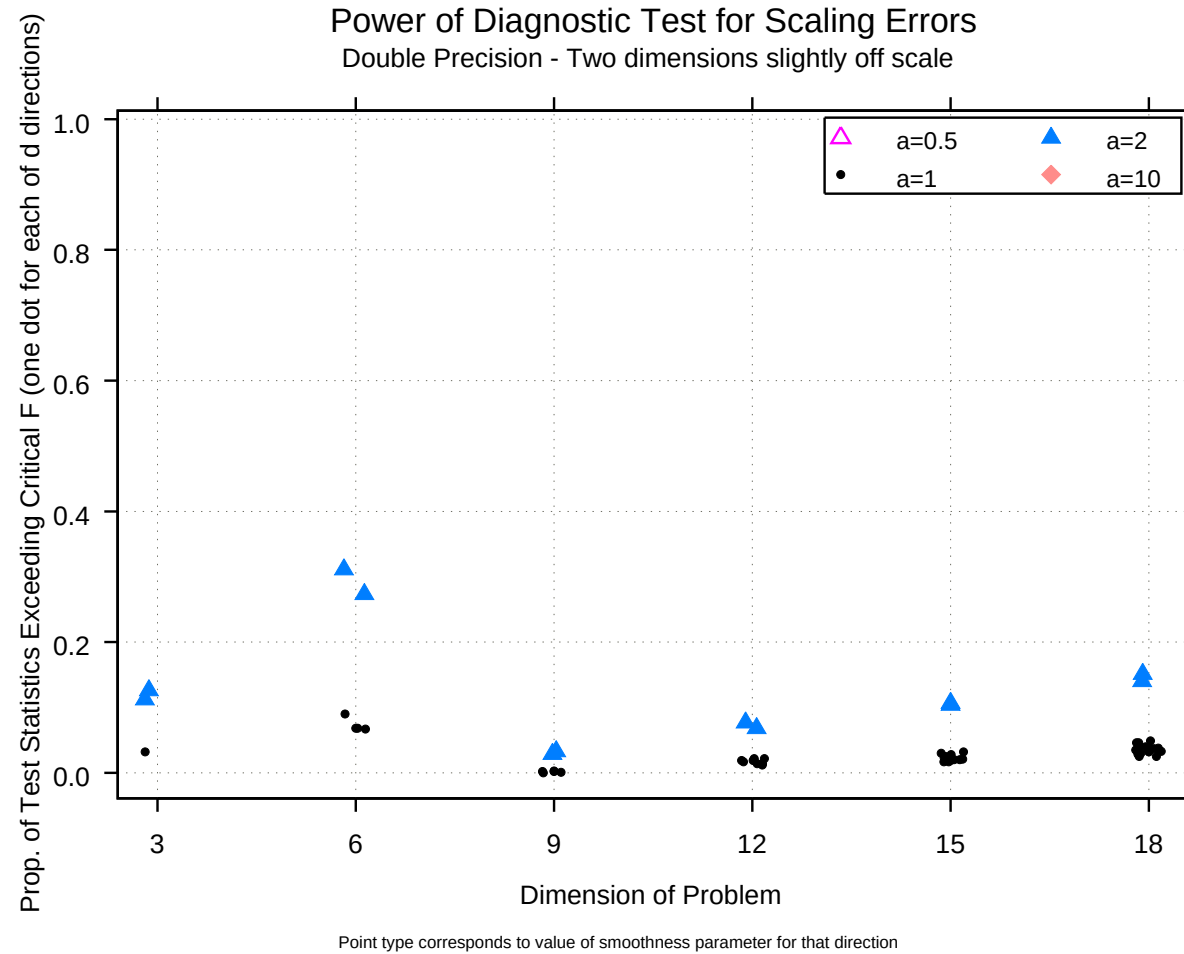


Figure 6.19: Plot of Diagnostic Test Power for Scaling Errors Using Double Precision – Two Dimensions Slightly Off-Scale.

Chapter 7

Conclusions and Future Work

The results of this research show that RSM designs are useful for numerical differentiation. These designs require fewer function evaluations than the Del12F method, but more than the Spendly method. We described a random function model that provides a useful framework in which these derivative approximation methods can be compared. Using this model for comparisons, we avoid the issues of creating a test-bed of specific functions, while taking into consideration the effect of round-off errors due to floating point arithmetic. This lets us draw general conclusions about the performance of the three derivative approximations we consider. The model can also help in selecting optimal step sizes that minimize the variance of a derivative approximation. These optimal step sizes are similar to those recommended in the literature, which builds confidence in the model's usefulness.

Using the model to evaluate the approximation methods, we showed that the Del12F method has very good precision for estimating first, second, and mixed derivatives, and that its precision is consistent across the number of dimensions. The Spendly method

also showed consistency, but as a trade-off for its considerably smaller sample sizes, it has uniformly lower precision than the Del12F method and generally lower precision than the RSM method. The RSM designs demonstrate good precision estimating first derivatives for smaller dimensions, but the precision degrades as the size of the problem grows. Still, this is of less concern as the first derivatives are estimated with much greater precision than the second and mixed derivatives. The RSM method excels at estimating second derivatives, outperforming the Del12F method in almost every case. When estimating mixed derivatives, the RSM method does relatively poorly in dimensions 7 through 10, but rivals the Spendly method otherwise.

Finally, we also described a diagnostic test with the RSM designs that can detect variable scaling problems. This pseudo F-test shows good power for detecting problems with scaling that is off by a factor of at least 10. Examining four test cases with different variable scaling configurations, we showed that the test is actually somewhat conservative and that its power declines in problems with an increasing number of dimensions.

7.1 Future Work

Further work remains to study how the scaling diagnostics can be effectively applied in practice. It may also be interesting to further evaluate the inherent design scaling for the various methods. Perhaps a different scaling would result in better precision. Specifically for the Del12F method, some of our results indicated that a fairly large scaling adjustment may result in increased precision. Additional work can also be done to include the RSM derivative approximations and the scaling diagnostics in optimization algorithms.

Regarding the random function model, other covariance functions can be examined, perhaps to allow for functions of varying smoothness or other characteristics beyond those we considered.

Finally, as the RSM method makes use of experimental designs, alternate design optimality criteria (besides D-optimality) could be considered to see if any improvements are possible. Such criteria are implemented in software like `gosset` (see Hardin and Sloane (1993)), which allows searching out D- as well as I- and A-optimal (or nearly optimal) designs for a wide range of response surface problems. This could be especially useful with approximating mixed derivatives.

Bibliography

Abramowitz, M. and I. A. Stegun (Eds.) (1972). *Handbook of Mathematical Functions*.

New York: Dover Publications, Inc.

Burden, R. L. and J. D. Faires (1993). *Numerical Analysis Fifth Edition*. Boston: PWS

Publishing Company.

Conn, A. R., K. Scheinberg, and P. L. Toint (1997). Recent progress in unconstrained optimization without derivatives. *Math. Prog. Ser. B* 79, 397–414.

Cressie, N. (1991). *Statistics for Spatial Data*. New York: John Wiley & Sons, Inc.

Davis, P. J. (1963). *Interpolation and Approximation*. New York: Blaisdell Publishing Company.

Dennis, Jr., J. E. and R. B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. New Jersey: Prentice-Hall.

Draper and Lin (1990). Small response-surface designs. *Technometrics* 32, No. 2.

Griewank, A. (1989). On automatic differentiation. In M. Iri and K. Tanabe (Eds.), *Mathematical Programming: Recent Developments and Applications*, pp. 83–108. Kluwer Academic Publishers.

- Hardin, R. H. and N. J. A. Sloane (1993). A new approach to the construction of optimal designs. *J. Statistical Planning Inference* 37, 339–369.
- Hartley, H. O. (1959). Smallest composite designs for quadratic response surfaces. *Biometrics*, 611–624.
- Iri, M. (1991). History of automatic differentiation and rounding error estimation. In A. Griewank and G. F. Corliss (Eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, Philadelphia, PA, U.S.A.
- Khuri, A. I. and J. A. Cornell (1987). *Response Surfaces*. New York: Marcel Dekker, Inc.
- Lawson, J. and J. Erjavec (2000). *Modern Statistics for Engineering and Quality Improvement*. California: Duxbury.
- Loève, M. (1963). *Probability Theory*. New York: Van Nostrand.
- Monahan, J. (2001). *Numerical Methods of Statistics*. New York: Cambridge University Press.
- SAS (2001). *The SAS System for Windows version 8*. Cary, NC: SAS Institute.
- Spendly, W., G. R. Hext, and F. R. Himsforth (1962). Sequential application of simplex designs in optimization and evolutionary operation. *Technometrics* 4, 441–461.
- Stewart, G. W. (1973). *Introduction to Matrix Computation*. New York: Academic Press.
- Tolsma, J. E. and P. I. Barton (1998). On computational differentiation. *Computers chem. Engng.* 22 4/5, 475–490.
- Westlake, W. J. (1965). Composite designs based on irregular fractions of factorials. *Biometrics*, 324–336.

Wexler, A. S. (1988). An algorithm for exact evaluation of multivariate functions and their derivatives to any order. *Computat. Statist. Data Anal.* 6, 1–6.

Appendix A

Covariance Function Derivatives

A.1 Covariances between the function and its first derivative

$$\begin{aligned}\text{Cov}\left(f(\mathbf{s}), \frac{\partial}{\partial t_k} f(\mathbf{t})\right) &= \frac{\partial}{\partial t_k} \Gamma_f(\mathbf{s}, \mathbf{t}) \\ &= 2 a_k^2 (s_k - t_k) \exp \left\{ - \sum_{i=1}^p a_i^2 (s_i - t_i)^2 \right\}\end{aligned}$$

$$\begin{aligned}\text{Var}\left(\frac{\partial}{\partial t_k} f(\mathbf{t})\right) &= \frac{\partial^2}{\partial s_k \partial t_k} \Gamma_f(\mathbf{s}, \mathbf{t}) \Big|_{\mathbf{s} = \mathbf{t}} \\ &= 2 a_k^2\end{aligned}$$

A.2 Covariances between the function and its second derivative

$$\begin{aligned}
\text{Cov} \left(U(\mathbf{s}), \frac{\partial^2}{\partial t_k^2} U(\mathbf{t}) \right) &= \frac{\partial^2}{\partial t_k^2} \Gamma_U(\mathbf{s}, \mathbf{t}) \\
&= (4 a_k^4 (s_k - t_k)^2 - 2 a_k^2) \exp \left\{ - \sum_{i=1}^p a_i^2 (s_i - t_i)^2 \right\}. \\
\text{Var} \left(\frac{\partial^2}{\partial t_k^2} U(\mathbf{t}) \right) &= \frac{\partial^4}{\partial s_k^2 \partial t_k^2} \Gamma_U(\mathbf{s}, \mathbf{t}) \Big|_{\mathbf{s} = \mathbf{t}} \\
&= 12 a_k^4.
\end{aligned}$$

A.3 Covariances between the function and its mixed derivative

$$\begin{aligned}
\text{Cov} \left(U(\mathbf{s}), \frac{\partial^2}{\partial s_j \partial t_k} U(\mathbf{t}) \right) &= \frac{\partial^2}{\partial s_j \partial t_k} \Gamma_U(\mathbf{s}, \mathbf{t}) \\
&= 4 a_j^2 a_k^2 (s_j - t_j)(s_k - t_k) \exp \left\{ - \sum_{i=1}^p a_i^2 (s_i - t_i)^2 \right\}. \\
\text{Var} \left(\frac{\partial^2}{\partial s_j \partial t_k} U(\mathbf{t}) \right) &= \frac{\partial^4}{\partial s_j^2 \partial t_k^2} \Gamma_U(\mathbf{s}, \mathbf{t}) \Big|_{\mathbf{s} = \mathbf{t}} \\
&= 4 a_j^2 a_k^2.
\end{aligned}$$

Appendix B

RSM Design Tables

The following listings contain the corner-points for the RSM designs. They are indexed by the number of dimensions and stored in a transposed format to save space (i.e. each column is a point in the design). To form the entire design, $2d$ additional face and $2d$ axial points must be added, as well as one point at the design center. For example, the following table shows the complete design for 2 dimensions:

RSM Design for Two Dimensions

X_1	X_2	Point Type
1.0	-1.0	corner
1.0	1.0	corner
-1.0	0	face
1.0	0	face
0	-1.0	face
0	1.0	face
-1.25	0	axial
1.25	0	axial
0	-1.25	axial
0	1.25	axial
0	0	center

The designs are also available upon request.

d=2	d=3	d=4	d=5	d=6
++	++--	+++++--	++++++--	++++++--
-+	+-+-	++-+-	+-----	++++-+-
	--+-	+-+--	-----	++-+-
		---+-	++++-+-	++-+-
			-----	++-+-
			++++++--	++-+-
				++-+-

d=7

```

++++-+-+--+-+--
++++-+-+--+-+--
+-+--+-+--+-+--
-+++--+-+--+-+--
+-+--+-+--+-+--
++-+-+--+-+--
+-+--+-+--+-+--

```

d=8

```

++++-+-+--+-+--
++++-+-+--+-+--
++-+-+--+-+--
---++++-+-+--
-++++-+-+--+-+--
++++-+-+--+-+--
++-+-+--+-+--
+-----+-+--+-+--

```

d=9

```

++-+++-+---+--+--++-++++-+-+---++-
+-++++++---++-+---++++-+-+---+++-
--+++++-----++-+---++++-+-+---++++
---+++++-----++-++-+---++-+---++
--++-++-++++-+-+--+--++-+---++-+---++
-++-++-++++-+-+--+--++-+---++-+---++
++-++-++++-+-+---+---++-+---++-+---++
+---+---++++-+-+---+---++-+---+++++
-++-++++++-+---+--+--++-+---++++-+

```

d=10

```

++++-++++-+---++-+---++-++-+---+---
++-+++-+--+++-+---++-++-+---+---++-
+-+++-+--+++-+---++-++-+---+---++-
++++-+--+++-+---++-++-+---+---++++-
+---++-++-+---++-++-+---+---++++-++-
--+-++++-+--+++-+---++-+---++++-++-
-+-++-+---++-+---++-+---++++-++-+-
-+++-+---++-+---++-+---++++-++-+-
+++-+---++-+---++-+---++++-++-+-
--+-++-+---++-+---++-+---++++-++-+-

```

d=11

```

-----+++++
-----+++++
-----+++++
----++-++++-++++-++++-++++-++++-++++
-+++-+--+++-++-++-++-++-++-++-++-++-
--++-+---++-+---++-+---++-+---++-+---
--++++-++++++-+---++++-+---++-+---++-
-+-+---++-+---++-++-++-++-++-++-++-++
+-+---+---++-++-++-++-++-++-++-++-++
++-+-+---++-+---++-++-++-++-++-++-++
+-+---+---++-++-++-++-++-++-++-++-++

```

d=12

```

-----+++++
-----+++++
-----+++++
----++++-++++-++++-++++-++++-++++
--++-+---++-++-++-++-++-++-++-++-
--+-++++-++-++-++-++-++-++-++-++-
-+-++++-++-++-++-++-++-++-++-++-
-+-++++-++-++-++-++-++-++-++-++-

```

++-----+--+++-+-+---++++---+---++-+-+---+++++-----+--+
-+---++-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+---+---++-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---

+++++
+++++
--+++++
++-----++
++---++---++
++---+-+---+-+
-+-+---+-+---
+-+---+-+---+-+
+++---+-+---
--+-+---+-+---
+++---+-+---
+++---+-+---

d=13

+---++++-+-+-----+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+++---+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
-++++-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+++---+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+---+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
--+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
--+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
++---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
-----+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---
+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---

--+-+---+-+---+-+---+-+---+-+---
-+-+---+-+---+-+---+-+---+-+---+-+---
++-----+-+---+-+---+-+---+-+---
---+-+---+-+---+-+---+-+---+-+---
++---+-+---+-+---+-+---+-+---+-+---
++---+-+---+-+---+-+---+-+---+-+---
+-+---+-+---+-+---+-+---+-+---+-+---
---+-+---+-+---+-+---+-+---+-+---+-+---
+-+---+-+---+-+---+-+---+-+---+-+---
+++-----+-+---+-+---+-+---+-+---

++-+--+++-+--+++-+--+
 -+--+++-+--+--+--+--+
 ++++-+--+--+--+--+--+

d=14

[illegible][illegible]

d=15

A handwriting practice sheet with ten rows of dashed lines. Each row contains a series of plus signs (+) placed at regular intervals along the dashed line, intended for tracing or copying practice.

[illegible]

A 20x20 grid of dashed lines. '+' symbols are placed at the following intersections (row, column) starting from (0,0) at the top-left:

- Row 0: Column 10
- Row 1: Columns 5, 10, 15
- Row 2: Columns 2, 7, 12, 17
- Row 3: Columns 1, 4, 8, 11, 14, 18
- Row 4: Columns 0, 3, 6, 9, 13, 16, 19
- Row 5: Columns 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 6: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 7: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 8: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 9: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 10: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 11: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 12: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 13: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 14: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 15: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 16: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 17: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19
- Row 18: Columns 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19
- Row 19: Columns 0, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19

-----++

+++++
+++++
+++++
+++++
+++++
--++
++--
---+
-+--
-+-+
-+-+
-+-+
+--+
+--+
-+-+
-+-+

Handwriting practice sheet with 12 rows of dashed lines. Each row contains a series of plus signs (+) and minus signs (-) placed at regular intervals along the dashed lines, intended for tracing or copying practice.

A 20x20 grid of dashed lines with '+' symbols at the intersections, forming a coordinate plane for graphing. The grid is composed of 20 horizontal and 20 vertical dashed lines, creating a series of small squares. '+' symbols are placed at every intersection point of the lines, forming a coordinate plane suitable for graphing functions or data points.

+++++
+++++
---+++++
+++-----+++++
+++--++-----+++++
+++--+--+-+-----+++++
-+-+-+-----+-+-----+
-+-+-----+-+-----+++++
+-+-----++-----+-+-----+
-+-+-+-----+-+-----+-+-----+
+-+-----+-+-----+-+-----+
-+-+-----+-+-----+-+-----+
+-+-----+-+-----+-+-----+
+-+-----+-+-----+-+-----+
+-+-----+-+-----+-+-----+
+-+-----+-+-----+-+-----+
-+-+-----+-+-----+-+-----+
-+-+-----+-+-----+-+-----+

d=18

A 20x20 grid of dashed lines. '+' symbols are placed at the intersections of the following rows and columns (using 1-based indexing from top-left):

- Row 1: Columns 15, 16, 17, 18, 19, 20
- Row 2: Columns 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
- Row 3: Columns 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 4: Columns 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 5: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 6: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 7: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 8: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 9: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 10: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 11: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 12: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 13: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 14: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 15: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 16: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 17: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 18: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 19: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
- Row 20: Columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

[illegible]

```

-+++++---++-+-+---++++-++++-++++-+-+---++++-+++++---+-+
+---+---++-++++-+-+---++++-+-+---++-+-+---++++-+++++---+-+
++++-+-+---++-++++-+-+---++-+-+---++++-+-+---++-+-+---++-+-+
++++-++++-+-+---++-+-+---++-+-+---++++-+-+---++-+-+---++++-
---++++-++++-+-+---++-+-+---++-+-+---++++-+-+---++-+-+---++-
--+-+---+---++-+-+---+-+---++-+-+---++-+-+---+-+---++-+-+---++-
--+-+---++++-+-+---+-+---++-+-+---++-+-+---++-+-+---++-+-+---++-
+-+---+-+---++-+-+---+-+---++-+-+---++-+-+---++-+-+---++-+-+
+-+---++++-+-+---+-+---++-+-+---++-+-+---++-+-+---++-+-+---++-

```

```

+++++
-----+++++
+++++-----+++++
-+++++-----+++++
+-----+++++-----+++++
+---++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-++-
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
--+-+---++-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
--++++-++-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+

```

d=20

```

-----
-----+++++
-----+++++-----+++++
-----+++++-----+++++-----+++++
---+++---+-+---+++---+++---+++---+++---+++---+++---+++
--+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
-+++-----+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
-+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
--+++---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+++---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+++---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+
+++---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+---+-+

```