The Pennsylvania State University

The Graduate School

Department of Statistics

ON CONVERGENCE OF THE NELDER-MEAD SIMPLEX ALGORITHM FOR UNCONSTRAINED STOCHASTIC OPTIMIZATION

A Thesis in

Statistics and Operations Research

by

John James Tomick

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 1995

PII Redacted

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 95

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
On Convergence of the Nedler-Mead Simplex Algorithm for Unconstrained Stochastic Optimization

5. FUNDING NUMBERS

6. AUTHOR(S)
John James Tomick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Students Attending:
Pennsylvania State University

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA
94-151

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPRTMENT OF THE AIR FORCE
AFIT/CI
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW 190-1
Distribution Unlimited
MICHAEL M. BRICKER, SMSgt, USAF
Chief Administration

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

19950103 052

14. SUBJECT TERMS

15. NUMBER OF PAGES
200

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT

18. SECURITY CLASSIFICATION OF THIS PAGE

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT

We approve the thesis of John James Tomick.

	Date of Signature
[illegible] Steven F. Arnold Associate Professor of Statistics Thesis Co-Advisor Co-Chair of Committee	11/22/94
[illegible] Russell R. Barton Associate Professor of Industrial Engineering Thesis Co-Advisor Co-Chair of Committee	11/22/94
[illegible] Tom M. Cavalier Associate Professor of Industrial Engineering	11/22/94
[illegible] Bruce G. Lindsay Distinguished Professor of Statistics	11/22/94
Susan Murphy Susan A. Murphy Assistant Professor of Statistics	11/22/94
[illegible] James L. Rosenberger Professor of Statistics Head of the Department of Statistics	22 Nov 1994

Abstract

The Nelder-Mead simplex method is a direct search algorithm that has found widespread use in the optimization of nonlinear functions. Originally designed for deterministic optimization, the method is robust with respect to small perturbations in the function's values; and therefore, this method has been used for optimizing stochastic functions as well. However, if the random perturbations in the function's values are large enough the method may terminate before reaching the optimizer of the expected function. We prove convergence of the simplex to a point *with probability 1* for constant functions with additive noise for 1- and 2-dimensional functions and provide empirical evidence for the same result in higher dimensions. This result implies that as the amount of noise increases, differences between the expected function's values at the vertices of the simplex become obscured and the probability of terminating early is increased. Also, we demonstrate empirically and analytically the probability of early termination on an unbounded univariate linear function with additive noise. We propose a new modification for the Nelder-Mead simplex method for use in stochastic optimization. This new method reduces to the original method when the noise level is negligible or nonexistent; and therefore, it is as efficient as the Nelder-Mead method when the noise level is low. And in Monte Carlo experiments, our proposed method continued to reduce the expected function for as long as the simulations were run; whereas the original method and the previously best know modified simplex methods terminated early.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this paper we present an analysis of the convergence properties of the simplex algorithm proposed by Nelder and Mead (1965) when applied to the minimization of a function in the presence of noise. We are interested in the performance of the algorithm for solving the nonlinear unconstrained minimization problem:

$$\min_{\mathbf{x}\in\Re^n} E[f(\mathbf{x})],$$

where $f : \Re^n \longrightarrow \Re$ is an unknown stochastic function. The knowledge we obtained from our analysis has led us to propose a new simplex algorithm for stochastic applications. Finally, we offer experimental results that demonstrate the superiority of our proposed algorithm over the Nelder-Mead algorithm.

1.1 Background

Today, there is a plethora of methods available to the analyst for optimizing nonlinear functions that cannot be solved analytically. These methods are necessarily iterative in nature and the user must supply some starting position or initial guesses for the parameters. Many of these methods use first-order or even second-order derivatives to determine a search direction to improve the value of the objective function, and they can be found in popular textbooks on nonlinear optimization such as those by Dennis and Schnabel (1983), Fletcher (1981), Gill, Murray, and Wright (1981) and Luenberger (1984). Zangwill (1969) presents a unified approach to proving convergence for such methods. However, these techniques "are totally deterministic in nature and when applied to problems affected by 'noise', whether it be error in measurement or uncertainty in prediction, they are either unable to reach an optimum at all or they may reach a false optimum" (Young, 1976, pp. 517-518). On the other hand, there exist techniques specifically designed to converge globally *almost surely* under certain conditions to a local optimizer when noise is present. These techniques are known as stochastic approximation methods, which were first introduced by Robbins and Monro (1951).

1.2 Direct-Search Methods

Distinct from derivative-based search methods and stochastic approximation methods is a class of techniques known as direct-search methods. In contrast to other optimization techniques which require derivatives of the objective function to determine a search direction, a direct-search method relies solely on the value of the objective function on a set of points. According to Torczon (1993), Hooke and Jeeves (1961) introduced the term "direct search." The advantages of direct-search methods over derivative-based methods include the following: (1) the calculations are simple, (2) the storage requirements are relatively low; (3) few adjustable parameters need to be supplied; and, (4) the algorithms are effective when evaluation errors are significant because they operate on the worst rather than the best point.

Reklaitis, Ravindran, and Ragsdell (1983) further classify direct-search methods into two classes: heuristic techniques and theoretically-based techniques. The heuristic techniques are search methods that have been constructed from geometric intuition; whereas, the theoretically-based techniques "have a mathematical foundation that allows performance guarantees, such as convergence, to be established, at least under restricted conditions" (Reklaitis et al., 1983, p. 75). The Nelder-Mead algorithm is a heuristic direct-search method. An example of a theoretically-based direct-search method is the conjugate direction method of Powell (1964). Direct-search methods are used for both deterministic and stochastic applications. They are effective techniques in deterministic applications especially when derivatives are unavailable or are computationally intensive (Parkinson & Hutchinson, 1972; Olsson, 1974). Also, they are robust with respect to small perturbations in the function's values; and therefore, they are used often in applications where noise is present (Barton & Ivey, 1993; Thompson, 1989).

1.3 History of the Nelder-Mead Simplex Algorithm

Nelder and Mead's simplex algorithm is based on the earlier work of Spendley, Hext, and Himsworth (1962). (Note, one should not confuse this method with the simplex method of linear programming developed by Dantzig (1963).) The algorithm developed by Spendley et al. employs a regular simplex which moves in the direction nearest to the direction of steepest ascent (for maximization) or steepest descent (for minimization) by replacing the worst vertex in the simplex with its mirror image across the face formed by the remaining vertices. A simplex is a polytope in n-dimensional space with $n+1$ vertices, each of which are are connected to all other vertices (*e.g.* a triangle in $\Re^2$, a tetrahedron in $\Re^3$, etc.). A regular simplex is one with edges of equal length. A simplex design has the minimum number of points required to estimate the gradient direction of the response surface in $\Re^n$ is $n+1$; and, furthermore, Brooks and Mickey (1961) demonstrate that this is the optimum number of points for maximizing the improvement in the estimate of the gradient direction per unit of effort even in the presence of noise. Nelder and Mead incorporated additional rules that allow the simplex to change its shape and size in order to conform to the behavior of the function in the local region of the simplex. The rate of progress toward the function's optimizer is presumably increased by the inclusion of the expansion and contraction steps, and the shrinkage step is included to permit convergence of the simplex to a point once the simplex has surrounded the optimizer.

It appears that Nelder and Mead had deterministic applications in mind when they modified the original simplex method of Spendley et al.. This belief is supported by the stopping criteria that they included in their algorithm (see Barton & Ivey, 1993) and the fact that they tested their algorithm on deterministic functions only. However, the algorithm by Spendley et al. is an implementation of the concept of evolutionary operation propounded by Box (1957). Evolutionary operation is experimental (or stochastic) optimization by nature. This has led others to infer that the Nelder-Mead method was designed for stochastic optimization, which is evidenced by the following statement:

> Although originally developed for experimental optimization, the simplex method [of Spendley et al.] and its modifications have been used extensively for mathematical optimizations such as the nonlinear least-squares fitting of data. (p. 282 A Deming & Morgan, 1973)

1.4 The Nelder-Mead Simplex Algorithm

Nelder and Mead's algorithm begins with the function's values on a set of $n + 1$ points in the parameter space. This set of points in the parameter space defines a polytope in $\Re^n$ which has $n+1$ vertices and is called a simplex. The algorithm proceeds through a sequence of operations on the simplex to direct it presumably toward a local optimum. Assuming that the optimum is a minimum, the algorithm does this by replacing the worst vertex in the simplex with a new point that has a lower function value through one of the following operations: reflection, expansion or contraction. If all of these operations fail to find a new point to replace the worst point in the simplex, then the entire simplex shrinks towards the vertex with the lowest function value. There is some disagreement about the interpretation of the rules set out by Nelder and Mead (1965); and therefore, there are several variants found in papers and computer implementations of the Nelder-Mead simplex algorithm. Additionally, some have proposed variations for other reasons. These variants are discussed and summarized in Section 1.4.2. A general consensus of the rules in the original algorithm are provided by Barton and Ivey (1993); these same rules can be found in Khuri (1987), too. An outline of the rules for the Nelder-Mead simplex algorithm for function minimization is as follows:

1. **Initialization.** For a function of n parameters, choose $n+1$ points to form an initial simplex in the parameter space ($\Re^n$). Usually this initial simplex is a regular $(n+1)$-sided polytope (*e.g.* an equilateral triangle in $\Re^2$, a regular tetrahedron in $\Re^3$, etc.). Evaluate the function at each vertex of the simplex and rank order the vertices according to their respective function values in ascending order. That is, let $\mathbf{x}_1$ represent the vertex with the lowest function value, $y_1 = f(\mathbf{x}_1)$, let $\mathbf{x}_2$ represent the vertex with the second lowest function value, y_2, etc., yielding the ordered set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \mathbf{x}_{n+1}\}$. Let $\mathbf{S}^0$ denote this initial simplex (ordered set).

2. **Simplex Update.** Let k denote the iteration number of this stage, which begins by removing the vertex with the highest function value $\mathbf{x}_{n+1}$ from the current simplex $\mathbf{S}^{k-1}$. Then find the centroid of the remaining n points of $\mathbf{S}^{k-1}$, which is denoted by $\mathbf{x}_{centroid}$. The centroid of n points in $\Re^n$ is the point defined by the arithmetic average of those points as given by Equation 1.1.

$$\mathbf{x}_{centroid} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{1.1}$$

Next, generate a new point by reflecting $\mathbf{x}_{n+1}$ through $\mathbf{x}_{centroid}$ according to Equation 1.2.

$$\mathbf{x}_{reflect} = (1+\alpha)\mathbf{x}_{centroid} - \alpha\mathbf{x}_{n+1} \tag{1.2}$$

where the reflection coefficient, α, is usually taken to be 1. Then depending upon the rank of $\mathbf{x}_{centroid}$ among the vertices of the current simplex, one of the following operations will be performed on the current simplex $\mathbf{S}^{k-1}$ (see Figure 1.1 for an illustration of these operations in $\Re^2$):

   (a) **Expansion.** If $y_{reflect} < y_1$, then the reflection is extended in the same direction using Equation 1.3.

$$\mathbf{x}_{expand} = \gamma\mathbf{x}_{reflect} + (1-\gamma)\mathbf{x}_{centroid} \tag{1.3}$$

   where the expansion coefficient, γ, is usually taken to be 2.

      i. **Accept.** If $y_{expand} < y_1$, then $\mathbf{x}_{expand}$ replaces $\mathbf{x}_{n+1}$ in the simplex;

      ii. **Reject.** else $\mathbf{x}_{reflect}$ replaces $\mathbf{x}_{n+1}$ in the simplex.

   (b) **Reflection.** If $y_1 \leq y_{reflect} \leq y_n$, then $\mathbf{x}_{reflect}$ replaces $\mathbf{x}_{n+1}$ in the simplex.

(c) **Contraction.** If $y_{reflect} > y_n$, then a contraction of the simplex is attempted. If $y_{reflect} \leq y_{n+1}$, then $\mathbf{x}_{reflect}$ replaces $\mathbf{x}_{n+1}$ and $y_{reflect}$ replaces y_{n+1} before a contraction or a shrinkage is performed. The contraction point is calculated using Equation 1.4.

$$\mathbf{x}_{contract} \;=\; \beta\mathbf{x}_{n+1} + (1-\beta)\mathbf{x}_{centroid} \tag{1.4}$$

where the contraction coefficient, β, is usually taken to be $\frac{1}{2}$.

i. **Accept.** If $y_{contract} \leq y_{n+1}$, then $\mathbf{x}_{contract}$ replaces $\mathbf{x}_{n+1}$ in the simplex;

ii. **Reject.** otherwise, shrink the entire simplex towards $\mathbf{x}_1$ by replacing all the other vertices, $\mathbf{x}_i$, $i = 2, \ldots, (n+1)$, using Equation 1.5.

$$\mathbf{x}_i \;=\; \delta\mathbf{x}_i + (1-\delta)\mathbf{x}_1 \tag{1.5}$$

where the shrinkage coefficient, δ, is usually taken to be $\frac{1}{2}$.

In all of our analyses and discussions to follow we assume the standard values for the coefficients of reflection, expansion, contraction and shrinkage.

3. **Return to Update Stage or Terminate.** At this time either one new point has replaced $\mathbf{x}_{n+1}$ given an expansion, a reflection or contraction has occurred, or n new points have replaced $\{\mathbf{x}_2, \ldots, \mathbf{x}_{n+1}\}$ given a shrinkage occurred. The new set of points is reordered according to their respective function values and this new set is denoted $\mathbf{S}^k$—the simplex at the end of the k^{th} iteration. If the stopping criterion is satisfied, then the algorithm terminates; otherwise, another iteration of the update stage is performed.

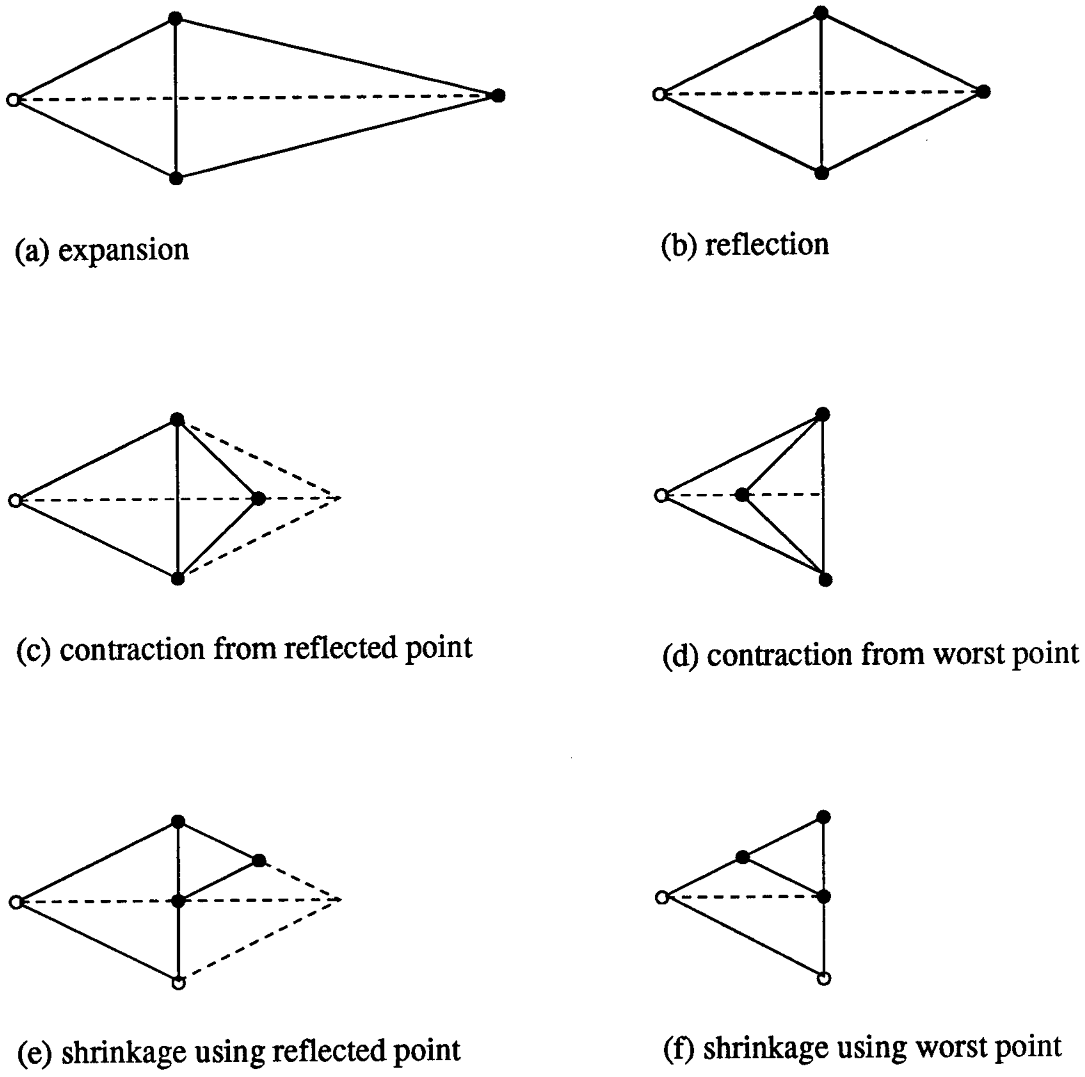

(a) expansion
(b) reflection
(c) contraction from reflected point
(d) contraction from worst point
(e) shrinkage using reflected point
(f) shrinkage using worst point

Figure 1.1: Simplex Operations in $\Re^2$ (solid dots indicate new simplex)

1.4.1 Stopping Criterion

Nelder and Mead proposed the standard error stopping criterion given in Equation 1.6 below.

$$\sqrt{\frac{1}{n+1}\sum_{i=1}^{n+1}(y_i - \bar{y})^2} \quad < \quad 10^{-8} \tag{1.6}$$

As Barton and Ivey (1993) point out, Nelder and Mead did not have stochastic functions in mind, where "the standard deviation of the function's values across all simplex points reflects inherent stochastic variations as well as differences in (expected) function values." They discuss the resulting problems making this an ineffective stopping criterion for use with stochastic functions. For their computational experiments, they used a stopping criterion based on the simplex size proposed by Dennis and Woods (1987). This criterion is

$$(1/\Delta) \max \|\mathbf{x}_i - \mathbf{x}_1\| \quad \leq \quad \epsilon \tag{1.7}$$

where the maximization is over all points i in the current simplex, $\Delta = \max(1, \|\mathbf{x}_1\|)$ and $\epsilon = 10^{-4}$. Other stopping criterion have been offered by Parkinson and Hutchinson (1972). One of these related to that of Powell (1964) is a measure of how far the simplex has moved. This measure is written as follows:

$$\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i^k - \mathbf{x}_i^{k+1}\|^2 \tag{1.8}$$

where vertex $\mathbf{x}_i^{k+1}$ replaces vertex $\mathbf{x}_i^k$ during the k^{th} iteration of the simplex update stage. With the exception of a shrinkage, only one vertex is exchanged during an iteration, which means that all but one of the terms in Equation 1.8 are zero except for a shrinkage. This stopping criterion inspired the one in Equation 1.7.

1.4.2 Implementation Details

Aside from the different stopping criteria that have been offered, variations of the Nelder-Mead algorithm in the simplex update stage can be found in the various computer implementations. Some of these variations may have resulted from a simple misunderstanding of the rules given by Nelder and Mead. For example, in the appendix to Thompson (1989) titled "A Simple Optimization Algorithm that Usually Works" the rule for a contraction uses $\mathbf{x}_1$ instead of $\mathbf{x}_{centroid}$. However, most of the following variations have been argued as improvements to the original algorithm. This section relies heavily on notes provided by Dr. Russell Barton who has investigated these variations in computer implementations.

One of the most widely implemented variation involves the acceptance test for an expansion. In the original algorithm x_{expand} replaces x_{n+1} if $y_{expand} < y_1$ (given $y_{reflect} < y_1$ of course). We denote this rule as AE1. Instead, others have proposed that x_{expand} replace x_{n+1} if $y_{expand} < y_{reflect} < y_1$, which we denote as rule AE2. One of the implementations using this stricter acceptance test (AE2) is The Sequential Simplex Program for Solving Minimization Problems by Olsson (1974). His code relies heavily upon Algorithm AS 47 by O'Neill (1971). Benyon (1976) offered the stricter acceptance test as a correction to Algorithm AS 47. Actual errors in the coding of AS 47 were found and corrected by Chambers and Ertel (1974) and Hill (1978). Other implementations using the stricter acceptance test for an expansion include IMSL routines U3POL and DU3POL, NAG routine E04CCF mark 13 revised 4/88, STATLIB 47 routine nelmin, and Indiana University QCPE program 307: STEPIT: simplex 2.9 June 1975.

Another variation to the original algorithm involves the acceptance test for a contraction. In the original algorithm a contraction is accepted if $y_{contract} \le y_{n+1}$

(given $y_{reflect} > y_n$ of course), which we denote by rule AC1. Some require that $y_{contract}$ be strictly less than y_{n+1} to accept the contraction (AC2). Implementations using rule AC2 include NAG routine E04CCF and a routine in Numerical Recipes. A still stricter acceptance test for a contraction is to replace x_{n+1} with $x_{contract}$ if $y_{contract} < y_n$ (AC3). That is, the new point must have a function value strictly less than the current second highest value in the simplex. Under the original rule (AC1) and the former stricter acceptance test (AC2) a contraction may result in a new worst point (*i.e.* $y_{contract} > y_n$). Therefore, $x_{contract}$ becomes x_{n+1} (the new worst point) and it will be replaced at the next iteration. Some of the implementations using rule AC3 include Dennis and Woods (1987) and the MATLAB routine nelder.m.

Each of these stricter acceptance rules—AE2, AC2 and AC3—have the effect of speeding up the convergence of the sequence of simplices to a limit point. Rule AE2 results in fewer expansions, and rules AC2 and AC3 result in more shrinkages than in the original algorithm. We use the stricter acceptance rule AE2 for all our research into the Nelder-Mead algorithm to simplify our analytical results presented in Chapters 3 and 4.

1.5 The Initial Simplex in $\Re^n$

One practical problem in using a simplex method is determining the location of the vertices for the initial simplex. One such starting simplex would be to locate one vertex at the origin and another at a unit's distance along each of the coordinate axes of the parameter space. However, the choice of a good starting location can greatly improve the performance of the algorithm. Therefore, prior knowledge or preliminary experiments may suggest a location far from the origin for the starting location. In that case the initial simplex could be translated to a new starting location. Furthermore, the above procedure does not yield a regular simplex as the length of any edge not connected to the origin will not have unit length. Spendley et al. (1962) provide formulas for generating a regular simplex of unit edge length with one vertex at the origin. The formulas given below allow the practitioner to specify the edge length and location of the center of mass for the starting simplex. Let d be the edge length and let $\mathbf{c}$ be the location of the center of mass. Then define the coordinates of the first vertex using Equations 1.9 - 1.11.

$$\mathbf{x}_1(j) = \mathbf{c}(j) - \frac{p + (n-1)\cdot q}{n+1} \tag{1.9}$$

where $j = 1, \ldots, n$ and

$$p = d \cdot \left(\frac{\sqrt{n+1} + n - 1}{n\sqrt{2}} \right) \tag{1.10}$$

$$q = d \cdot \left(\frac{\sqrt{n+1} - 1}{n\sqrt{2}} \right) \tag{1.11}$$

Then the remaining vertex coordinates are defined by Equation 1.12.

$$\mathbf{x}_i(j) = \mathbf{x}_1(j) + \begin{cases} p & \text{if } j = i-1 \\ q & \text{if } j \neq i-1 \end{cases} \tag{1.12}$$

where $i = 2, \ldots, n+1$ and $j = 1, \ldots, n$. The resulting simplex will be regular with edge length d and center of mass at $\mathbf{c}$.

1.6 Extensions for Deterministic Optimization

In Section 1.4.2 we reviewed minor changes to the rules for accepting various operations in the Nelder-Mead simplex algorithm. In this section we review some of the more interesting and significant extensions or improvements to the algorithm proposed by Nelder and Mead. These extensions have been proposed to improve the performance of the Nelder-Mead algorithm on deterministic functions. Although we did not investigate the performance of these modified algorithms for stochastic optimization, we include a review of these algorithms for completeness.

Many of these improvements have been suggested by those working in analytical chemistry, where the simplex method has been widely used since the publication of two articles by Deming and Morgan (1973, 1974), where the authors names are appear in reverse order in the latter article. They termed the Spendley et al. algorithm the sequential simplex method and the Nelder-Mead algorithm the modified simplex method. As a result, many authors now refer to the Nelder-Mead algorithm as the Modified Simplex Method (MSM), and some authors refer to Spendley et al.'s algorithm as the Basic Simplex Method (BSM). However, Deming and Morgan (1973) did not explicitly include the operation of shrinkage for the Nelder-Mead method; and therefore, many applications of MSM do not include a shrinkage operation.

First, Routh, Swartz, and Denton (1977) proposed what they called the Super Modified Simplex (SMS) method. Their proposed improvement of the MSM (*i.e.* the Nelder-Mead algorithm) is a major modification eliminating the fixed step lengths and combining the reflection, contraction and expansion operations into a single step. Their algorithm evaluates the function at the centroid given by Equation 1.1 and evaluates the reflected point as in MSM. Then the algorithm fits a second-order polynomial to the points $\mathbf{x}_{n+1}$, $\mathbf{x}_{centroid}$ and $\mathbf{x}_{reflect}$. In addition, the curve is extrapolated beyond $\mathbf{x}_{n+1}$ and $\mathbf{x}_{reflect}$ by a percentage of the vector $\mathbf{x}_{n+1} - \mathbf{x}_{reflect}$. The derivative of this polynomial is evaluated and the location of the optimum value within this interval becomes the new vertex except if the new vertex is too close to the centroid, which would essentially terminate further progress in one or more dimensions. This safety restriction is lifted when approaching the termination of the search. They compared their algorithm to three slightly modified versions of the Nelder-Mead method which had been offered previously. These modified versions each included resampling of the best vertex when retained for $n + 1$ successive iterations. An additional modification was included in the second version where the second worst vertex was reflected if the function's value at the contracted vertex was the new worst vertex. The third version always accepts a contraction (*i.e.* no shrinkages). Their algorithm outperformed all three slightly modified versions of the Nelder-Mead algorithm in experimental tests.

Van Der Wiel (1980) claimed a substantial improvement to the SMS algorithm in terms of decreasing the number of experiments (*i.e.* function evaluations) required, as in the optimization of parameter settings of an analytical instrument. All combinations of three modifications to the SMS algorithm were investigated using five two-parameter response surfaces with no added noise. These modifications were the substitution of a Gaussian fit in lieu of the polynomial fit, a weighted centroid, and substitution of an estimate of the response at the centroid in lieu of an actual evaluation. The Gaussian fit and the estimated response at the centroid performed better than SMS both individually and in combination. The weighted centroid failed to improve SMS in any combination it appeared. Interestingly, the use an estimated response at the centroid was introduced again in the same journal by Parker, Cave, and Barnes (1985) without reference to Van Der Wiel (1980).

Independently, Ryan, Barr, and Todd (1980) investigated a weighted centroid method (WCM) as a modification to MSM. The rationale behind a weighted centroid is to obtain a better approximation to the gradient direction for a reflection of the worst point. As reported by Van Der Wiel (1980), the weighted centroid method does follow the gradient direction more closely and shows an improvement over MSM in the first few iterations. However, closer approximation to the gradient direction results in more expansions during the early iterations resulting in a highly elongated simplex. This distortion later diminishes the ability of the simplex to search in directions orthogonal to the elongation. Contraction operations eventually restore some degree of regularity to the simplex at the cost of wasted function evaluations and the convergence criteria may be met before the simplex can change direction. Two solutions are offered to the degeneracy problem which are termed controlled weighted centroid method (CWC) and orthogonal jump weighted centroid method (OJWC). Tests on functions suggest that both CWC and OJWC are initially better than MSM and SMS, but MSM performs the best near the optimum of the function.

Van Der Wiel, Maassen, and Kateman (1983) investigated the problem of degeneracy of the simplex when using a weighted centroid method. The OJWC method of Ryan et al. (1980) uses the determinant of the simplex to detect degeneracy, but Van Der Wiel et al. demonstrate that it cannot be used to detect degeneracy because the determinant of the next simplex is independent of the centroid, weighted or not. They propose using a measure of the symmetry of the simplex to restrict the coefficient for expanding the simplex determined by the Gaussian or polynomial fit procedure. The measure of symmetry is the n^{th} root of the absolute value of the determinant divided by the radius of the sphere passing through all vertices of the simplex which is scaled by the number of parameters to yield a value in the range from 0 to 1. A regular simplex has a symmetry of 1. With this control the safety interval around the centroid is unnecessary except when the symmetry falls below a specified level resulting from a boundary violation. Empirical optimization of the symmetry criterion yielded an optimal value of about 0.5 independent of the noise level.

Silver (1981) suggested that more significant improvements to algorithms with geometric interpretations may be found by altering the space rather than the algorithm. That is, he suggests translating an algorithm to non-Euclidean hyperbolic space which holds the prospect of avoiding problems with speed of convergence and false convergence resulting from plateaus in the response surface. These problems are related to the path of convergence of an algorithm. Within Euclidean space, the path of convergence can be changed by changing starting positions for the search algorithm or by changing the geometric rules of the algorithm. The former is inconvenient in experimental studies and the latter has not resulted in significant improvements. However, different convergence paths can be obtained in hyperbolic space with the same starting position by varying the metric constant. He discusses how such a translation of the Nelder-Mead simplex algorithm to non-Euclidean space can be made, but references other sources for many of the details. However, there is as yet no method for determining what value of the metric constant is best for a particular objective function.

In two articles Betteridge, Wade, and Howard (1985a, 1985b) present another simplex algorithm and the results of comparative evaluations with 8 other simplex algorithms including BSM, MSM, SMS, CWC, and OWJC. They call their algorithm the composite modified simplex method (CMS). The CMS algorithm uses an adaptive weighted centroid that approaches the unweighted centroid of the Nelder-Mead method as the response values become close together (as in the region of the optimum). It also discards the shrinkage operation and uses the strict acceptance rule for an expansion. Additionally, the CMS algorithm incorporates a Lagrange polynomial fit whenever a failed expansion or a successful contraction occurs indicating that an optimum lies within the interval of the current simplex. It includes a safety interval to prevent accepting the new point too close to the centroid as in the SMS algorithm of Routh et al..

Next, Smith (1986) offered a modified algorithm that moves more directly down a slope than the Nelder-Mead algorithm. This algorithm reflects a high centroid (average of all but the best point) through a low centroid (average of all but the worst point, which is identical to the Nelder-Mead centroid). If the reflected point's value is the best, then an extension (or expansion) is attempted. If the extended point's value is better than the reflected point's value, then the extended point becomes the reflected point and an extension is repeated until the extended point's value is no longer an improvement over the reflected point's value. His method can save many iterations over the Nelder-Mead method if the path from the initial starting simplex to the optimum is not severely curved, particularly in cases where there are a large number of parameters. However, the price for its directness is an increase in computation time and memory.

A simplex algorithm by Cheok, Hu, and Loh (1988) uses the strict acceptance rule for an expansion, and if successful, takes the expanded point as a new initial point and begins the simplex search with a new simplex around this expanded point. They claimed superior performance of this algorithm for their application to identification of parameters of a simulated servomotor system with stick-slip friction.

Marsili-Libelli and Castelli (1987) provided another simplex algorithm and claimed it outperforms the Nelder-Mead simplex algorithm. Theirs substitutes a unidirectional optimal search algorithm for the fixed step-length reflection. Therefore, there is no longer any distinction between a reflection and an expansion. Instead they perform a line search along the direction from the worst vertex through the centroid of the remaining vertices using the Fibonacci interval elimination method. Obviously, the number of function evaluations is increased whenever an expansion is performed. However, their algorithm results in fewer but more efficient expansions. Their empirical studies show a savings in the number of iterations yielding a smaller number of total function evaluations for the new algorithm, particularly when the minimum lies at the bottom of a narrow "valley" in the parameter space.

Finally, Subrahmanyam (1989) proposed a better way to handle inequality constraints when using the Nelder-Mead simplex algorithm than the traditional approach of setting the function value at an infeasible point to some large value. Equality constraints can be converted to inequality constraints. Using the traditional approach the algorithm is likely to terminate prematurely when a boundary is reached especially when the feasible region is nonconvex. Subrahmanyam proposed what he terms a delayed reflection. If a vertex becomes infeasible, its function value is not increased until the next iteration is completed.

1.7 Known Convergence Results

Although the Nelder-Mead simplex method is easy to implement and has gained widespread acceptance for minimizing a function, very little is known about its convergence properties. The term "convergence" with respect to the Nelder-Mead method has had two distinct meanings: (1) convergence of the simplex to a single point, or set of points with equal function values; and (2) convergence of a convex combination of the function values to the optimal function value. Presumably one would hope that if the simplex were to converge to a single point, that this point would be the optimizer of the function, but that is not necessarily true.

Woods (1985) presents preliminary convergence results for a slightly modified Nelder-Mead algorithm employing a stricter acceptance rule for an expansion of the simplex (see Section 1.4.2). Under appropriate conditions, he proves convergence of the simplex to a connected set of limit points, but did not establish any properties for the limit point $\mathbf{x}^*$, *e.g.* $\nabla f(\mathbf{x}^*) = 0$. After working on a proof of convergence, Dennis (1994) now believes that the results obtained by Woods may be all one can say analytically about the algorithm's convergence. In fact, Dennis and Woods (1987) provide an example for which a slightly modified Nelder-Mead algorithm does not converge to the minimizer of a strictly convex function.

Recently, Torczon (1993) established a global first-order stationary point convergence theory for a subclass of the heuristic direct-search methods that she termed "pattern-search" methods. This subclass includes her multidirectional search algorithm (Torczon, 1991; Dennis & Torczon, 1991), response surface methodology (Box & Wilson, 1951) and the Hooke and Jeeves' pattern search method (Hooke & Jeeves, 1961). However, the Nelder-Mead algorithm does not meet the definition of a "pattern-search" method. Furthermore, there does not appear to be a way to modify her proof so as to incorporate it either.

Barton (1980) compared properties of convergence to the optimizer of four search methods on univariate functions with noise. These methods were Spendley et al.'s simplex method, two modified versions of the simplex method and the stochastic approximation method of Kiefer and Wolfowitz (1952). One of the modified simplex methods, termed the double simplex method, involves resampling the two vertices following a movement of the simplex to allow for a homogeneous Markov process on a linear function contaminated by noise. The other modified version, termed the variable step simplex method, decreases the length of the simplex (*i.e.* a line segment) with a corresponding increase in the number of samples at each vertex as the optimum is approached. This version prevents a decrease in the signal-to-noise ratio as the function levels off near the optimum. Empirical results demonstrate that the rate of convergence for the Kiefer and Wolfowitz method make it impractical if the starting values are far from the optimum, even though the method is guaranteed to converge *with probability one.* The double simplex method demonstrates better performance than the simplex method when the noise is large. While the variable step simplex method was the most robust its efficiency is less than that of the preceding two methods. However, “it does well on more kinds of problems in a ‘reasonable’ number of observations than do the others” (p. 16 Barton, 1980).

More recently, Barton and Ivey (1993) empirically tested the stochastic convergence of several variants of the Nelder-Mead algorithm. All of the methods tested exhibit the phenomenon of converging to a false optimum, which they termed “premature convergence,” when noise is present. That is, the simplex method is subject to false moves when the signal—a function of the magnitude of the gradient of the response in the region of the simplex and the size of the current simplex—is small with respect to the noise. A reflection or expansion signaled by noise is not a problem as the algorithm is somewhat self-correcting. However, a contraction or a shrinkage signaled by noise only serves to increase the effect of the noise at the next iteration. In fact, a shrinkage of the simplex (using the usual coefficient of 1/2) results in a decrease in the volume of the simplex by a factor of $(1/2)^n$. Therefore, a simplex may shrink prematurely, which is a signal that an optimum has been located; but, if the true function were not obscured by noise, then the simplex would have continued to make progress toward a true optimum. Thus, the algorithm is said to have converged prematurely.

1.8 Overview

The goals we set for our research included a theoretical proof that the simplex of the Nelder-Mead method converges to a point on stochastic functions; and that there is a positive probability that this point is not the optimizer of the function. Secondly, we set out to develop a modified simplex method with better performance characteristics on stochastic functions.

In Chapter 2 we review several journals for applications of Nelder-Mead in both deterministic and stochastic settings. In Chapter 3 we present a proof that the Nelder-Mead simplex converges to a point on stochastic functions with constant expectation in one and two dimensions. In Chapter 4 we use Markov chain theory to demonstrate that the Nelder-Mead simplex converges to a point on a one-dimensional unbounded stochastic function with linear expectation. This analysis is followed by our proposed modified method for univariate stochastic optimization. In Chapter 5 we report the results of an experiment designed to compare the performance of our modified simplex method with Nelder-Mead, another modified simplex method and the Kiefer-Wolfowitz stochastic approximation method. In Chapter 6 we develop extensions for our modified method to n dimensions; and finally, we report the test results of two implementations of our modified method on multivariate stochastic functions.

Chapter 2

Applications and Motivation

To motivate the purpose of this research we review several articles published in recent years that report using a simplex algorithm in an applied problem—most of these used a slightly modified version of the Nelder-Mead simplex algorithm. This review is by no means exhaustive as "there have been over 2000 citations to the original paper ... [with] applications ... from physics to biology to manufacturing process optimization" (Barton & Ivey, 1993). We present the review of these articles in science and technology to demonstrate the use of the Nelder-Mead simplex algorithm as an optimization technique with wide applicability to nonlinear optimization problems both in a deterministic setting and in the presence of noise. Having impressed upon the reader the widespread use of the Nelder-Mead simplex method, and its use for stochastic optimization in particular, leads naturally to the motivation for this research into its stochastic convergence properties.

2.1 The Nelder-Mead Method and Curve Fitting

Many investigators have used the Nelder-Mead algorithm to minimize a sum of squared residuals function as a nonlinear least-squares curve fitting technique. A paper by Caceci and Cacheris (1984) titled "Fitting Curves to Data: The Simplex Algorithm is the Answer" is cited in many of the papers mentioned below. One major shortcoming of many implementations of the Nelder-Mead simplex algorithm when used for curve fitting is that error estimates of the fitted parameters are not provided. This is addressed by Caceci and Cacheris who suggest using a Monte

Carlo simulation approach to perturb the final parameter estimates using a Gaussian distribution with mean zero and standard deviation given by their program called Simp. They suggest generating a sufficiently large number of simulated experimental point sets and then re-running their simplex algorithm implementation on each of these sets. The standard deviation of a fitted parameter is estimated by the sample standard deviation of the final parameter estimates from these multiple runs.

Another approach to error estimation of the fitted parameters is addressed by Phillips and Eyring (1988). They implemented a proposal in an appendix to the original article by Nelder and Mead for constructing an estimate of the curvature matrix of second derivatives at the minimum without the evaluation of derivatives. They propose a new criterion for protecting against a lack of curvature across the final simplex due to unnecessarily strict stopping criteria. If the curvature criterion is not met, then the final simplex is expanded by doubling the distance between the centroid and each vertex and the criterion is checked again. Although it is natural to think of noise when least-squares curve fitting is mentioned, this is not a stochastic application of the Nelder-Mead simplex algorithm. Once the data have been obtained, the fitting of these data to a curve is completely deterministic. Next we review some of the many deterministic applications that have appeared in the literature followed by a review of some of the stochastic applications.

2.2 Deterministic Applications

There are many references describing the use of the Nelder-Mead method for deterministic applications in chemistry. The use of Nelder-Mead in analytical chemistry is largely credited to Deming and Morgan (1973). In this paper they illustrate its utility as a least-squares curve fitting technique. In another paper Morgan and Deming (1974) use the Nelder-Mead simplex method to optimize the multivariate response surface of the manual colorimetric method of Carr and Drekter for determination of total cholesterol in the blood. Deming and Parker (1978) provide a review of simplex optimization in analytical chemistry through 1976. None of those applications are reviewed here except the two already mentioned above. Instead we review some of the more recent applications of Nelder-Mead that have been published.

In one such application, Busing and Matsui (1984) offered the Nelder-Mead simplex algorithm as an alternative to the Newton-Rhapson method for solving the minimum energy structure of a crystal. A mathematical model of the crystal structure incorporating the application of external forces is simulated and minimized, but the simulation is deterministic.

Burgess and Hayumbu (1984) used the Nelder-Mead simplex method to locate the optimal values for four parameters in a trace element determination technique known as neutron activation analysis. The simplex optimization was applied to a response function generated from experimental data. (A stochastic application would require the experiments to be guided by the simplex optimization). Later, Burgess (1985) extended this work to the determination of multiple trace elements in a single sample.

Rhines and Arnold (1988) reported the use of multiple indicators in a fiber-optic ammonia-gas sensor for the first time. The primary goal of their investigation was to develop a practical optimization strategy for selecting multiple indicators for a fiber-optic sensor. A sensor response function is optimized by a simplex method for the indicator solution to identify the best combination of two indicators.

Jang and Rajeshwar (1988) used the Nelder-Mead simplex method in conjunction with computer simulation to investigate the influence of thermal resistances on the peak resolution of a thermogram produced by a differential scanning calorimetry cell. Differential thermal analysis is a technique for identifying and quantitatively analyzing the chemical composition of substances by observing the thermal behavior of the sample as it is heated. The Nelder-Mead simplex method was used to determine the values of the parameters for successive simulation runs. However, the simulation appears to have been completely deterministic using differential equations.

A curve-fitting application in ellipsometry, a technique for studying film growth in electrochemical systems, is given in a paper by De Smet and Ord (1989). Also, the minimization of the sum of squared residuals for the problem of source location of an earthquake can be found in papers by Rabinowitz (1988), and Prugger and Gendzwill (1988). Vance, Hassani, and Mottahed (1988) use the Nelder-Mead simplex algorithm to minimize the sum of absolute deviations for the earthquake source location problem. Still another approach to this problem using the Nelder-Mead simplex algorithm is found in a paper by Prothero, Taylor, and Eickemeyer (1988). Other curve-fitting applications of the Nelder-Mead simplex algorithm include the calibration of seismometers (Steck & Prothero, 1989), the calculation of fluid-mineral equilibria (Wood, 1993), and the material properties of bonds and composite laminates (Karim & Mal, 1990).

All of the applications of the Nelder-Mead simplex algorithm in this section are deterministic applications even though many involve experimental data. Our research interest is in the performance of the Nelder-Mead method on stochastic objective functions arising for example, from laboratory experiments or simulation experiments. In this setting, the Nelder-Mead algorithm generates the parameter values for the next run of the experiment. The response cannot be known precisely due to process variation, measurement error, or simulated noise.

2.3 Stochastic Applications

A method for determining the amount of calcium in the blood was investigated by Olansky, Parker, Morgan, and Deming (1977) using the Nelder-Mead simplex method to optimize reagent concentrations. That is, the simplex method was used under experimental conditions to determine the concentrations of reagents to optimize three objectives for the serum calcium determination. The objectives included in the simplex optimization were to maximize calcium sensitivity while minimizing magnesium sensitivity and protein sensitivity and were combined into a single objective function.

Rainey and Purdy (1977) used a modified simplex method to experimentally optimize the volumes of solvent components to separate the peaks in a chromatogram of a mixture of other components. The volumes of solvent components were generated by the rules of the Nelder-Mead simplex algorithm and the response was obtained by a laboratory experiment.

Ebdon, Cave, and Mowthorpe (1980) employed the Nelder-Mead simplex algorithm to maximize the signal-to-background ratio of different plasmas for optical emission spectrometry over five operating parameters: the power in the plasma, the observation height, the injector, and plasma and coolant gas flow rates. The rules of the algorithm were used to generate new settings for these parameters to search for the largest signal-to-background ratio.

Ibarra and Lazaro (1985) described using a modified Nelder-Mead simplex method to determine the settings for three significant variables in successive experimental trials involving the sulfonation of coal for use in water treatment technologies. The three variables were temperature, reaction time, and the ratio of sulfonic acid (volume) to coal (mass). The published ranges of these variables for coal sulfonation were scaled to a 0-1 scale and an initial regular simplex with unit edge lengths was used to determine the variable settings for the first four experiments.

Mass spectrometry, or mass spectroscopy, denotes a field of physics in which the motion of ions (charged atoms, molecules or fragments of molecules) in electric and magnetic fields is used to sort ions according to their mass-to-charge ratios. A mass spectrometer is an instrument that uses electricity to detect ions. One such device is known as a Fourier transform ion cyclotron resonance (FT-ICR) mass spectrometer, which is capable of obtaining very high resolution mass measurements in the narrow band mode. A problem associated with these measurements is the requirement for extremely precise and time-consuming instrument tuning to obtain the maximum resolution possible. The problem arises from a large number of dependent instrument parameters that must be manipulated in the tuning process, which are highly specific for the ion signal being observed. Elling, de Koning, Pinkse, and Nibbering (1989) introduced a formal optimization of FT-ICR instrument tuning using the Nelder-Mead simplex method. Previously, "the most successful operators have tuned by intuition developed by long experience, often using only a subset of the relevant parameters. Even with such experience the day to day tuning results are not always reproducible" (Elling et al., 1989, p. 330).

Lawitts and Biggers (1991) applied the Super Modified Simplex algorithm of Routh et al. to the optimization of mouse embryo culture media composed of 10 components. A concentration was chosen for each component to define the initial medium from which 10 other media were generated. These 11 media defined the vertices of the initial simplex. The media yielding the worst response was identified experimentally. The next media to be tested was identified by the simplex algorithm (a reflection of the worst vertex), etc. This simplex procedure identified four components which at high concentrations are detrimental to embryo development (as the concentrations of these media were greatly reduced in the new media generated by the simplex optimization).

The minimization of a weighted objective function of multiple economic criteria for a machining operation is accomplished experimentally using the Nelder-Mead simplex algorithm in a paper by Agapiou (1992). For simplicity and purposes of being able to visualize the response, only two variables—cutting speed and feed rate—were considered for optimization.

Finally, a problem for astronomers is the alignment of segmented mirrors in large optical telescopes in the presence of atmospheric turbulence. Mehta and Allen (1993) present computer simulations and experimental results which demonstrate the utility of a remote method for aligning a segmented mirror known as far-field optimization. This technique utilizes the Nelder-Mead simplex algorithm to minimize the residual phase error in the presence of static or dynamic turbulence.

The applications reviewed in this section are stochastic applications of modified simplex algorithms based on the algorithm of Nelder and Mead. The number of stochastic applications of the Nelder-Mead simplex algorithm is likely to increase as the use of this technique finds its way to other disciplines.

2.4 Motivation for this Research

As one can see from the preceeding list of applications, the simplex method of Nelder and Mead has become a widely used tool for nonlinear optimization problems. In fact, it is "the most popular direct-search method based on published applications ..." (Barton & Ivey, 1993). The efficiency and simplicity of the Nelder-Mead simplex algorithm has made it a very popular choice among practitioners. However, some of the claims of convergence for this algorithm are misleading. Some of these misleading remarks from the articles cited above include: "this method ... always converges" (Prothero et al., 1988, p .1190); "The simplex algorithm always converges to a minimum, regardless of starting model, ..." (Steck & Prothero, 1989, p. 1618); and "Divergence is impossible" (Caceci & Cacheris, 1984, p. 344). Traditionally, "convergence" of an iterative nonlinear optimization technique means that the algorithm will locate the global optimum (or a local optimum) under specified conditions in the limit as the number of iterations is allowed to go to infinity.

The only known proof of convergence, which is for a slightly modified Nelder-Mead algorithm, is the one provided by Woods (1985) which we discussed in Section 1.7. Since all simplex operations in the Nelder-Mead algorithm retain the current best vertex, the method cannot diverge to infinity in the parameter space unless the objective function is unbounded below (for minimization). This is probably what is meant by the above claims that the method cannot diverge (unlike the Newton-Rhapson method when the intial guesses for the parameters are too far from the optimal values). However, even a proof that Nelder-Mead converges to the optimum under deterministic conditions would not guarantee its performance in stochastic optimization.

Our research into the convergence properties of Nelder-Mead on stochastic functions is motivated by its practical application in this setting. Our first goal is to provide the practitioner with an understanding that experimental noise may result in the algorithm terminating at a "false optimum." Along with providing the practitioner with an understanding that this problem exists, we also seek to provide a basis for determining how serious this problem can be for a general nonlinear function. We do this by investigating the phenomenon of false convergence in the presence of noise on linear functions, since a linear function is a first-order local approximation of the general function in the region of the simplex. Finally, we have endeavored to discover a better simplex algorithm which would lessen the severity of the false convergence phenomenon while maintaining the relative advantages of the Nelder-Mead method.

Chapter 3

Convergence on Stochastic Functions with Constant Expectation

As noted earlier in Chapter 2, the Nelder-Mead simplex method has found widespread acceptance in the optimization of stochastic functions as well as deterministic functions. Researchers often choose the Nelder-Mead algorithm for two very practical reasons: (1) no assumptions about the objective function or its derivatives are needed; and (2) the algorithm's conceptual simplicity makes it easy to understand and program. Even though it was originally designed with deterministic functions in mind, it is robust with respect to small perturbations in the function's values.

As Barton and Ivey (1993) have noted, the Nelder-Mead method relies solely on the ranks of the vertices based on the function's values. Therefore, small perturbations that do not affect these ranks have no effect on the algorithm's search trajectory. However, if the perturbations in the function's values are large enough to affect the relative ranks of the simplex vertices, then an incorrect pivot choice may be made. For example, the simplex may shrink when the correct action may be to reflect. The factors affecting the relative ranks of the simplex vertices on a stochastic function include the magnitude of the noise, the size and shape of the current simplex and the magnitude of the gradient of the function in the local region of the simplex. When an incorrect shrinkage occurs due to noise, the noise will have an even greater effect on the relative ranks at the next iteration because the simplex is now smaller (assuming that the function is locally convex). A smaller simplex means that the differences in the function's expected values are smaller, which allows the noise to play a more dominant role in determining the relative ranks of the vertices. Such a situation can lead the algorithm to converge at a point that is not the true optimum.

In this chapter we prove that the algorithm converges *with probability one* (*w.p. 1*) when applied to a function of 1 or 2 parameters where random noise is the only factor affecting the relative ranks of the points in the simplex. By convergence we mean that a subsequence of the sequence of simplices converges to a single point; that is,

$$\lim_{k \to \infty} \mathbf{S}^k = \{\mathbf{x}^*, \ldots, \mathbf{x}^*\}.$$

Additionally, we prove that the shrinking of the simplex is sufficiently fast enough to guarantee that the distance the simplex traverses in the parameter space is finite. We suspect that this is the case in higher dimensional parameter spaces as well and provide some empirical evidence to support this conjecture.

We assume an unknown stochastic function $f : \Re^n \longrightarrow \Re$, with constant expectation and noise component having a continuous distribution independent of the location of the simplex in the parameter space. Symbolically, we write $f(\mathbf{x}) = \mathbf{g}(\mathbf{x}) + \mathbf{W}(\mathbf{x})$, where $g(\mathbf{x}) = \mathbf{c}$ (a constant) and $W(\mathbf{x}) = \mathbf{W}$ is a random variable with a continuous distribution. Since the function's values are independent identically distributed (iid) random variables, we will denote them by Y_i. We assume a continuous distribution to avoid the possibility of two points in the parameter space having the same function value. Also, repeated observations at the same point in the parameter space will yield different realizations. To further simplify our calculations we assume independence for the function's values at the vertices of the simplex across iterations. This can be achieved by resampling the vertices at the beginning of each iteration.

3.1 Diameter of a Simplex

We define the size of a simplex to be the length of its longest edge. This measure of simplex size was chosen by Hensley, Smith, and Woods (1988) for their analysis of simplex distortions under repeated reflections. Formally, Hensley et al. define the size of the simplex $\mathbf{S} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{n+1}\}$ to be the diameter of this set of points, which is defined as

$$d(\mathbf{S}) \;=\; \max_{0 \leq i < j \leq n+1} \|\mathbf{x}_i - \mathbf{x}_j\|. \tag{3.1}$$

Since the actual position of the simplex at any future iteration is a random variable, we define the size of the simplex at the end of the k^{th} iteration to be the random variable D_k defined by Equation 3.2 below:

$$D_k \;=\; d(\mathbf{S}^k). \tag{3.2}$$

We chose this definition of simplex size because of its conceptual simplicity and the knowledge that convergence of the length of the longest edge to zero implies convergence of the simplex to a single point. We considered the volume of the simplex as the measure of simplex size; however, proving the volume of the simplex converges to zero does not imply that the simplex has converged to a single point. For example, if the simplex collapses in only one dimension so that it is contained in an $(n-1)$-dimensional subspace of $\Re^n$, then the volume of the simplex in $\Re^n$ would be zero. And yet, the simplex would not be a point; therefore, it could continue to move within that subspace of the parameter space.

3.2 Effect of Simplex Operations on Simplex Size

In order to prove that the size of the simplex in the Nelder-Mead algorithm converges to zero we need to understand how each of the simplex operations affects it. In the case of a simplex expansion or a simplex contraction, the length of the longest edge may not change. This is more likely to happen when n is large as each of these operations replace only one vertex. The exchange of one vertex affects n edges, whereas the total number of edges is given by

$$\sum_{i=1}^{n}(n+1-i) \;=\; \sum_{i=1}^{n} i \;=\; \frac{n(n+1)}{2}.$$

Therefore, the exchange of one vertex during an expansion or a contraction may not affect the size of the simplex as we have defined it in Equation 3.1.

In contrast, a shrinkage always affects the size of the simplex by a factor equal to the shrinkage coefficient, δ, which is usually set to $1/2$. This is obvious when one of the endpoints of the longest edge is the vertex with the best function value. Since all other vertices are moved closer to $\mathbf{x}_1$ by a factor of δ, then the longest edge will be shortened by this factor δ. Moreover, the longest edge will be shortened by this same factor even if $\mathbf{x}_1$ is not one of its endpoints. Suppose the two endpoints for the longest edge in the simplex are $\mathbf{x}_i$ and $\mathbf{x}_j$ where $i \neq j \neq 1$. Then $\mathbf{x}_1, \mathbf{x}_i$, and $\mathbf{x}_j$ form a triangle in a subspace of dimension 2. If $\|\mathbf{x}_i - \mathbf{x}_1\|$ and $\|\mathbf{x}_j - \mathbf{x}_1\|$ are shortened by the factor δ, then by appealing to an argument using similar triangles $\|\mathbf{x}_i - \mathbf{x}_j\|$ is shortened by the factor δ, too.

Finally, although a reflection does not affect the volume of a simplex, it can distort the shape of a nonregular simplex in $\Re^n$ for $n \geq 3$. This distortion can cause the simplex size (Equation 3.1) to grow under repeated reflections. Hensley et al. (1988) prove that this growth is bounded; their theorem is repeated here.

Theorem 1 (Hensley-Smith-Woods) *Let* $\mathbf{S}$ *denote a nondegenerate simplex in* $\Re^n$ *with* $n \geq 3$. *Let* $\mathbf{S}^k$ *denote the result of* k *simplex reflections applied successively to* $\mathbf{S}$. *Then*

$$\frac{d(\mathbf{S}^k)}{d(\mathbf{S})} \leq \begin{cases} \sqrt{\dfrac{n+1}{2}}, & \text{if } n \text{ odd;} \\ \sqrt{\dfrac{n(n+2)}{2(n+1)}}, & \text{if } n \text{ even.} \end{cases}$$

Furthermore, this result is sharp in the sense that given any $\epsilon > 0$ *there exists a simplex* $\mathbf{S}$ *and a sequence of* k *reflections (for some* k*) so that*

$$\frac{d(\mathbf{S}^k)}{d(\mathbf{S})} > \begin{cases} \sqrt{\dfrac{n+1}{2}} - \epsilon, & \text{if } n \text{ odd;} \\ \sqrt{\dfrac{n(n+2)}{2(n+1)}} - \epsilon, & \text{if } n \text{ even.} \end{cases}$$

The bounds given in Theorem 1 are upper bounds on the amount of increase in the size of the simplex resulting from repeated reflections which are expressed as a fraction of the starting simplex size. Since any sequence of simplex reflections could be reversed, the multiplicative inverse of these bounds establish lower bounds for the amount of decrease in the size of the simplex as a fraction of its starting size due to repeated reflections.

Since a reflection can affect the size of the simplex and every iteration of the Nelder-Mead algorithm begins by evaluating a reflection of the worst vertex, we further divide a simplex interation into two halves. During the first half of an iteration $\mathbf{x}_{reflect}$ is located using Equation 1.2 and it replaces the vertex $\mathbf{x}_{n+1}$ if it has a better function value (where better means lower if we are minimizing the function). Suppose $\mathbf{x}_{reflect}$ does have a better function value; then during the second half of the iteration, the new vertex $\mathbf{x}_{reflect}$ may be expanded, left alone, contracted, or the whole simplex may be shrunk. On the other hand, if $\mathbf{x}_{n+1}$ is not replaced by $\mathbf{x}_{reflect}$, then the only possiblities for the second half of the iteration are a contraction of this vertex or a shrinkage of the simplex.

We define the variable α_k to be the fractional change in size of the simplex during the first half of the k^{th} iteration. If we let $\mathbf{S}^{k-\frac{1}{2}}$ denote the simplex after the first half of the k^{th} iteration, then

$$\alpha_k \quad = \quad \frac{d(\mathbf{S}^{k-\frac{1}{2}})}{d(\mathbf{S}^{k-1})}. \tag{3.3}$$

Next, we determine an upper bound on the fractional change in simplex size for each operation during the second half of the k^{th} iteration. A least upper bound could be determined using the dimension size; however, this is not required for our proof of simplex convergence. Instead, these upper bounds apply to all dimensions n. We define ρ_k to be the random variable on this finite space of upper bounds which is summarized in Equation 3.4.

$$\rho_k = \begin{cases} 2 & : \text{ expansion} \\ 1 & : \text{ reflection, failed expansion, or contraction} \\ 0.5 & : \text{ shrinkage} \end{cases} \tag{3.4}$$

Given a simplex operation and this finite space, ρ_k satisfies the inequality of Equation 3.5, which we prove in Section 3.3.

$$\rho_k \quad \geq \quad \frac{d(\mathbf{S}^{k})}{d(\mathbf{S}^{k-\frac{1}{2}})}. \tag{3.5}$$

3.3 Determining the Values for ρ_k

Determining the smallest value for ρ_k which satisfies Equation 3.5 independent of the current shape of the simplex is an easy exercise except for a simplex expansion which we devote to a separate section. Note that when an attempted contraction fails in the univariate case ($n = 1$), there is only one point to shrink because the simplex consists of only two points. And, the new vertex is the same point as the contracted point. Therefore, we say that a contraction cannot occur when $n = 1$ as the resulting new vertex will be the same regardless of its function value. Instead, a contraction is counted as a shrinkage when $n = 1$.

3.3.1 ρ_k for Reflection, Contraction and Shrinkage

Since a reflection is completed during the first half of an iteration, α_k accounts for any change in size resulting from a distortion of the shape. Therefore, $\rho_k = 1$ given a reflection during the k^{th} iteration. Note that a failed expansion results in the same simplex as a reflection.

Assuming the usual value of 0.5 for the contraction coefficient, a contraction of a simplex in $\Re^n$ when $n > 1$ will not decrease the size of the simplex if the worst vertex is not an endpoint of the longest edge. In this situation, the size remains unchanged. Therefore, $\rho_k = 1$ given a contraction occurred during the k^{th} iteration.

Finally, as we have already discussed, a shrinkage will cause the size of the simplex to decrease by a factor equal to the shrinkage coefficient. Assuming the usual value for the shrinkage coefficient, $\delta = 0.5$, $\rho_k = 0.5$ given a shrinkage during the k^{th} iteration.

3.3.2 ρ_k for Expansion

Assuming $\gamma = 2$ (where γ is the expansion coefficient), an expansion of a simplex in $\Re^1$ (*i.e.* a line segment) will cause a doubling of the size of the simplex. Furthermore, 2 is the smallest possible value independent of the current simplex

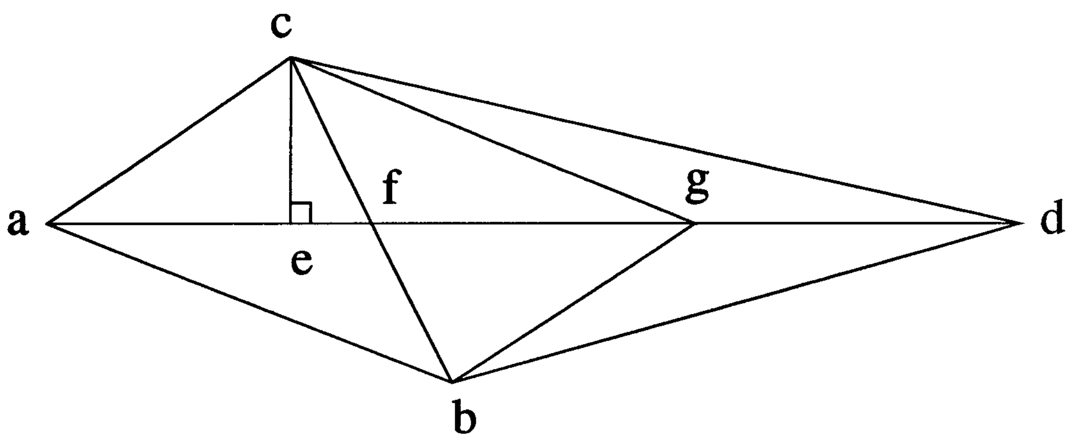


Figure 3.1: Expansion of Simplex in $\Re^2$

shape for the fraction of increase in the size of the simplex resulting from an expansion in higher dimensions as well. In fact, this bound cannot be attained for nondegenerate simplices in higher dimensions. First, consider the case where $n = 2$ and refer to the illustration in Figure 3.1 for the following discussion. Let the triangle formed by the points a, b, and c be the current simplex in $\Re^2$ which we denote by $\triangle abc$. The longest edge of this triangle is $\overline{ab}$ and the worst point of the triangle is a. The centroid of vertices b and c is f. A reflection results in the new simplex $\triangle gbc$ where $\overline{gc}$ has the same length as $\overline{ab}$. An expansion results in the new triangle $\triangle dbc$ with longest edge $\overline{dc}$. By the following geometric argument we demonstrate that $\|c - d\| < 2\|a - b\|$. Note that an expansion doubles the distance between the worst point and the centroid of the remaining points. Therefore, $\|f - d\| = 2\|a - f\| = 2\|f - g\|$. Now consider the two right triangles $\triangle ceg$ and $\triangle ced$. By the Pythagorean Theorem we have the following relationships:

$$\|c - g\|^2 \;=\; \|c - e\|^2 + \|e - g\|^2$$

$$
\begin{aligned}
&= \|c-e\|^2 + (\|e-f\| + \|f-g\|)^2 \\
&= \|c-e\|^2 + \|e-f\|^2 + 2\|e-f\|\|f-g\| + \|f-g\|^2. \qquad (3.6)
\end{aligned}
$$

$$
\begin{aligned}
\|c-d\|^2 &= \|c-e\|^2 + \|e-d\|^2 \\
&= \|c-e\|^2 + (\|e-f\| + \|f-d\|)^2 \\
&= \|c-e\|^2 + (\|e-f\| + 2\|f-g\|)^2 \\
&= \|c-e\|^2 + \|e-f\|^2 + 4\|e-f\|\|f-g\| + 4\|f-g\|^2. \qquad (3.7)
\end{aligned}
$$

Therefore, by Equations 3.6 and 3.7 we have $\|c-d\|^2 - 4\|a-b\|^2$

$$
\begin{aligned}
&= \|c-d\|^2 - 4\|c-g\|^2 \\
&= \|c-e\|^2 + \|e-f\|^2 + 4\|e-f\|\|f-g\| + 4\|f-g\|^2 \\
&\quad - 4\left(\|c-e\|^2 + \|e-f\|^2 + 2\|e-f\|\|f-g\| + \|f-g\|^2\right) \\
&= -3\|c-e\|^2 - 3\|e-f\|^2 - 4\|e-f\|\|f-g\|. \qquad (3.8)
\end{aligned}
$$

Equation 3.8 implies $\|c-d\|^2 < 4\|a-b\|^2$; and therefore, $\|c-d\| < 2\|a-b\|$ (*i.e.* an expansion cannot double the size of a simplex in $\Re^2$). In fact, given a nondegenerate simplex (*i.e.* $\|c-f\| > 0$) the fraction of increase is maximized when $\|e-f\| = 0$ (*i.e.* the longest edge is the hypotenuse of a right triangle formed by the endpoints of the longest edge and $\mathbf{x}_{centroid}$).

Now consider the case where $n > 2$. Refer to Figure 3.1 once again. As in the case where $n = 2$ let f be the centroid of all simplex vertices except the worst point, and let g be the reflected point. Let c represent another vertex of the simplex such that $\overline{cg}$ would be the longest edge of the simplex if g replaced the worst point.

Any change in the size of the simplex as a result of a distortion would be accounted for by α_k. By the earlier argument, if an expansion occurred (*i.e.* d replaced the worst point instead of g), then the simplex size after the expansion is less than twice the simplex size after a reflection. Therefore, an upper bound for the fraction of increase in size resulting from an expansion is 2.

3.4 Formulation of Probabilities

To prove the convergence of the simplex to a point, we need to know the probability masses for each value in the parameter space of the random variable ρ_k. These probabilities are directly related to the probabilities of performing one or more of the operations on the simplex. For example, the probability that $\rho_k = 2$ is equivalent to the probability that an expansion occurs during the k^{th} iteration. Determining these probabilities for the first iteration on a function with constant expected value is rather easy because the function's values at the vertices are iid. However, without resampling a dependency is introduced because the algorithm discards high function values in favor of new ones that are lower. The structure of this dependency becomes so complex that we could derive analytically only bounds on these probabilities for future realizations of ρ_k. Therefore, to simplify the proof we assume that all vertices are resampled at the beginning of each iteration to maintain independence between the values at the vertices. Furthermore, we use the strict acceptance rule for an expansion which we discussed in Section 1.4.2.

When the Nelder-Mead algorithm is initialized the starting simplex is defined by $n+1$ points in $\Re^n$ and the function's values at these points. Given a function with constant expectation these values are iid random samples. Furthermore, this condition of iid samples is renewed if we resample all vertices at the beginning of each iteration. Let $Y_{[i:n+1]}$ be the i^{th} order statistic of this set of iid random variables, $\{Y_1, \ldots, Y_{n+1}\}$. Ignoring the univariate case for this discussion, the relative rank of the next draw from the error distribution among the total of $n+2$ draws determines whether an expansion will be attempted (E), a reflection will occur (R) or a contraction will be attempted (C). Since the $n+2$ draws are iid, the probability that Y_{n+2} has any rank in the range $\{1, \ldots, n+2\}$ is $1/(n+2)$. Equations 3.9 - 3.11 illustrate the calculation of the probabilities of the events E, R and C respectively.

$$\begin{aligned} P(E) &= P(Y_{n+2} < Y_{[1:n+1]}) \\ &= P(Y_{n+2} = Y_{[1:n+2]}) \;=\; \left(\frac{1}{n+2}\right). \end{aligned} \tag{3.9}$$

$$\begin{aligned} P(R) &= P(Y_{[1:n+1]} < Y_{n+2} < Y_{[n:n+1]}) \\ &= \sum_{i=2}^{n} P(Y_{n+2} = Y_{[i:n+2]}) \;=\; \left(\frac{n-1}{n+2}\right). \end{aligned} \tag{3.10}$$

$$\begin{aligned} P(C) &= P(Y_{n+2} > Y_{[n:n+1]}) \\ &= \sum_{i=n+1}^{n+2} P(Y_{n+2} = Y_{[i:n+2]}) \;=\; \left(\frac{2}{n+2}\right). \end{aligned} \tag{3.11}$$

If an expansion is to be attempted, then the next step in the algorithm is to extend the reflection in the same direction. If the value of the function at the new point is lower than the reflected point's value, then the new point will be accepted (A_E). Under our assumptions, this is equivalent to asking if the relative rank of the latest draw is first out of a total of $n+3$ draws. If the relative rank of the latest draw is not first, then the expansion has failed (A'_E, where the prime denotes the complement of the event) and we keep the reflected point instead. Now, calculating the probability that an expansion of the simplex will occur during the first iteration

Table 3.1: Notation of Possible Events

Notation	Description of Event
E	an expansion is attempted
R	a reflected point is accepted immediately
C_R	a contraction from the reflected point is attempted
C_W	a contraction from the worst point is attempted
A_i	attempted operation $i \in \{E, C_R, C_W\}$ is accepted

follows from basic probability laws; that is, we need the probability an expansion is attempted times the conditional probability the attempt is successful given an expansion was attempted. Mathematically, we write

$$P(E \cap A_E) \;=\; P(E)P(A_E|E) \;=\; \left(\frac{1}{n+2}\right)\left(\frac{1}{n+3}\right). \tag{3.12}$$

And, in the same way we can derive the probabilities for all of the outcomes that can occur during an iteration of the algorithm. Each outcome consists of one or two of the events listed in Table 3.1. The possible combinations of these events are listed in Table 3.2, and by the same reasoning used to develop Equation 3.12 we arrive at the probabilities for each of these outcomes. These probabilities are summarized in Table 3.3.

When $n = 1$ note that the simplex is a line segment and since there are only two vertices in our simplex the one with the lowest function value ($\mathbf{x}_1$) is the same vertex having the second highest function value ($\mathbf{x}_n$). This has the effect of eliminating the possibility of accepting the reflected point immediately, and under the assumption that the error distribution is continuous we have $P(Y_1 \leq Y_{reflect} \leq$

Table 3.2: Notation of Possible Outcomes

Notation	Description of Outcome
$E \cap A_E$	a successful expansion
$E \cap A'_E$	a failed expansion
R	a reflection
$C_R \cap A_{CR}$	a successful contraction from the reflected point
$C_W \cap A_{CW}$	a successful contraction from the worst point
$C_R \cap A'_{CR}$	a shrinkage involving the reflected point
$C_W \cap A'_{CW}$	a shrinkage involving the worst point

$Y_n) = P(Y_{reflect} = Y_1) = 0$. Note that the general formula for $P(R)$ in Table 3.3 yields 0 as well.

Finally, we can formulate the probabilities for the random variable of interest, ρ_1, which is a multiplicative parameter yielding an upper bound on the new simplex size following the second half of an iteration. Using the sample space of ρ_k given in Equation 3.4 and the first iteration probabilities summarized in Table 3.3 we obtain the results given in Table 3.4.

Table 3.3: First Iteration Probabilities of Outcomes

Outcome	Probability of Outcome		
	$n=1$	$n=2$	General n
$E \cap A_E$	$\frac{1}{12}$	$\frac{1}{20}$	$\left(\frac{1}{n+2}\right)\left(\frac{1}{n+3}\right)$
$E \cap A'_E$	$\frac{3}{12}$	$\frac{4}{20}$	$\left(\frac{1}{n+2}\right)\left(\frac{n+2}{n+3}\right)$
R	0	$\frac{5}{20}$	$\left(\frac{n-1}{n+2}\right)$
$C_R \cap A_{CR}$	$\frac{2}{12}$	$\frac{3}{20}$	$\left(\frac{1}{n+2}\right)\left(\frac{n+1}{n+3}\right)$
$C_W \cap A_{CW}$	$\frac{2}{12}$	$\frac{3}{20}$	$\left(\frac{1}{n+2}\right)\left(\frac{n+1}{n+3}\right)$
$C_R \cap A'_{CR}$	$\frac{2}{12}$	$\frac{2}{20}$	$\left(\frac{1}{n+2}\right)\left(\frac{2}{n+3}\right)$
$C_W \cap A'_{CW}$	$\frac{2}{12}$	$\frac{2}{20}$	$\left(\frac{1}{n+2}\right)\left(\frac{2}{n+3}\right)$

Table 3.4: Probabilities for ρ_1

ρ_1	$Prob\{\rho_1\}$		
	$n=1$	$n=2$	$n \geq 2$
2	$\frac{1}{12}$	$\frac{1}{20}$	$\frac{1}{n^2+5n+6}$
1	$\frac{3}{12}$	$\frac{15}{20}$	$\frac{n^2+5n+1}{n^2+5n+6}$
$\frac{1}{2}$	$\frac{8}{12}$	$\frac{4}{20}$	$\frac{4}{n^2+5n+6}$

Theorem 2 $E[\rho_1] < 1$.

Proof:

1. ($n = 1$): From Table 3.4 we have

$$\begin{aligned} E[\rho_1] &= 2P(\rho_1 = 2) + P(\rho_1 = 1) + 0.5P(\rho_1 = 0.5) \\ &= 2\left(\frac{1}{12}\right) + \frac{3}{12} + 0.5\left(\frac{8}{12}\right) = \frac{3}{4}. \end{aligned}$$

2. ($n > 1$): Again, from Table 3.4 we have

$$\begin{aligned} E[\rho_1] &= 2P(\rho_1 = 2) + P(\rho_1 = 1) + 0.5P(\rho_1 = 0.5) \\ &= 2\left(\frac{1}{n^2+5n+6}\right) + \frac{n^2+5n+1}{n^2+5n+6} + 0.5\left(\frac{4}{n^2+5n+6}\right) \\ &= \frac{n^2+5n+5}{n^2+5n+6} < 1 \quad \text{for all } n. \end{aligned}$$

■

3.5 Proof of Convergence

In this section we prove that the expected value of the simplex size, *i.e.* the length of the longest edge, following the k^{th} iteration is bounded above by another expectation of $2k$ multiplicative factors and the initial simplex size. Secondly, we prove that the infinite sum of simplex sizes converges *w.p. 1* for $n \leq 2$. Recall that we defined D_k to be the simplex size following the k^{th} iteration (Equations 3.1 and 3.2). Also, we defined α_k to be the multiplicative factor for the change in size occurring during the first-half of iteration k (Equation 3.3), and ρ_k to be an upper bound for the change in size occurring during the second-half of iteration k (Equation 3.5). Then, the following results were obtained for the Nelder-Mead method with resampling of the simplex vertices after each iteration on a stochastic function with constant expectation.

Theorem 3 $\quad E[D_k|D_0 = d_0] \;\leq\; d_0 E\left[\prod_{i=1}^{k} \alpha_i \rho_i\right].$

Proof: Combining Equations 3.2, 3.3 and 3.5 yield the following recursive bound on the variable simplex size:

$$D_k \leq \alpha_k \rho_k D_{k-1}. \tag{3.13}$$

Repeated applications of Equation 3.13 leads to the following relationship:

$$D_k \leq D_0 \prod_{i=1}^{k} \alpha_i \rho_i. \tag{3.14}$$

Therefore, by Equation 3.14 we have

$$\begin{aligned} E[D_k|D_0 = d_0] &\leq E\left[D_0 \prod_{i=1}^{k} \alpha_i \rho_i | D_0 = d_0\right] \\ &= d_0 E\left[\prod_{i=1}^{k} \alpha_i \rho_i | D_0 = d_0\right] \end{aligned}$$

$$= d_0 E\left[\prod_{i=1}^{k} \alpha_i \rho_i\right]$$

as d_0 is simply a scaling parameter. ■

Corollary 1 *Given $D_0 = d_0$, $\sum_{k=1}^{\infty} D_k$ converges w.p. 1 for $n \leq 2$.*

Proof:

$$\begin{aligned}
\sum_{k=1}^{\infty} E[D_k|D_0 = d_0] &\leq \sum_{k=1}^{\infty} d_0 E\left[\prod_{i=1}^{k} \alpha_i \rho_i\right] && \text{by Theorem 3} \\
&= \sum_{k=1}^{\infty} d_0 E\left[\prod_{i=1}^{k} \rho_i\right] && \text{as } \alpha_i = 1 \text{ for } n \leq 2 \\
&= \sum_{k=1}^{\infty} d_0 \left(E[\rho_1]\right)^k && \text{as the } \rho_i \text{ are iid given resampling} \\
&< \infty,
\end{aligned}$$

as this series is a geometric series with rate $E[\rho_1] < 1$ by Theorem 2. Therefore, by the First Borel-Cantelli Lemma we have the desired result. A generalization of this lemma from Billingsley (1986, p. 305) is stated below for completeness.

Lemma 1 (Borel-Cantelli) *If X_k are nonnegative and $\sum_{k=1}^{\infty} E(X_k) < \infty$, then $\sum_{k=1}^{\infty} X_k$ converges w.p. 1.* ■

We strongly believe the result holds for higher dimensional parameter spaces; however, we were unable to derive an analytical proof. Such a proof must deal with the problem that the simplex size may increase as a result of a reflection. We tried to develop an analytical bound for the product $\prod_{i=1}^{k} \alpha_i$ as in Theorem 1, but to no avail. Furthermore, each α_i depends upon the past even with resampling because α_i is a function of the current shape of the simplex as well as the relative ranks of the vertices. In Section 3.6 we present empirical evidence for our conjecture that the result still holds for higher dimensional parameter spaces.

If $\sum_{k=1}^{\infty} D_k$ converges *w.p. 1*, then $D_k \to 0$ *w.p. 1* (meaning the simplex shrinks to a point *almost surely*). Furthermore, the convergence of the series guarantees that the total distance traveled by the simplex is finite *w.p. 1*. If the distance traveled by the simplex on a constant function is not finite *w.p. 1*, then the simplex would never converge on an unbounded linear function with noise having finite variance. Observance of "false convergence" could be attributed to not having allowed the algorithm to run long enough. Instead, given this result we now hypothesized that a simplex could shrink to a point on a linear function if the noise was sufficiently large to mask the difference in expected function values across the simplex. This would result in the determination of a false optimum when clearly there is none as the function is unbounded. In the next chapter we demonstrate that there is a positive probability for false convergence on a univariate linear function with noise.

3.6 Empirical Results

We offer the following empirical evidence to support our conjecture that the simplex shrinks to a point on a constant function of more than 2 parameters with additive random noise. We simulated the operation of the Nelder-Mead simplex algorithm using the strict acceptance rule for an expansion both with and without resampling for parameter space dimensions $n = 1, \ldots, 5$. 50 replications of the first 200 iterations under each of the 10 conditions specified above were made. Each run was initialized with a unit-sized regular simplex with center of mass located at the origin. After each iteration, the simplex was translated so that its center of mass was again located at the origin to avoid problems with machine round-off errors as the simplex size was decreased. The estimated responses for each iteration were the mean size of the simplex D_i, the mean value of α_i and the mean value of the product $(\alpha_1 \cdots \alpha_i)$. Plots of these results at 10-iteration intervals are presented in Figures 3.2 - 3.6.

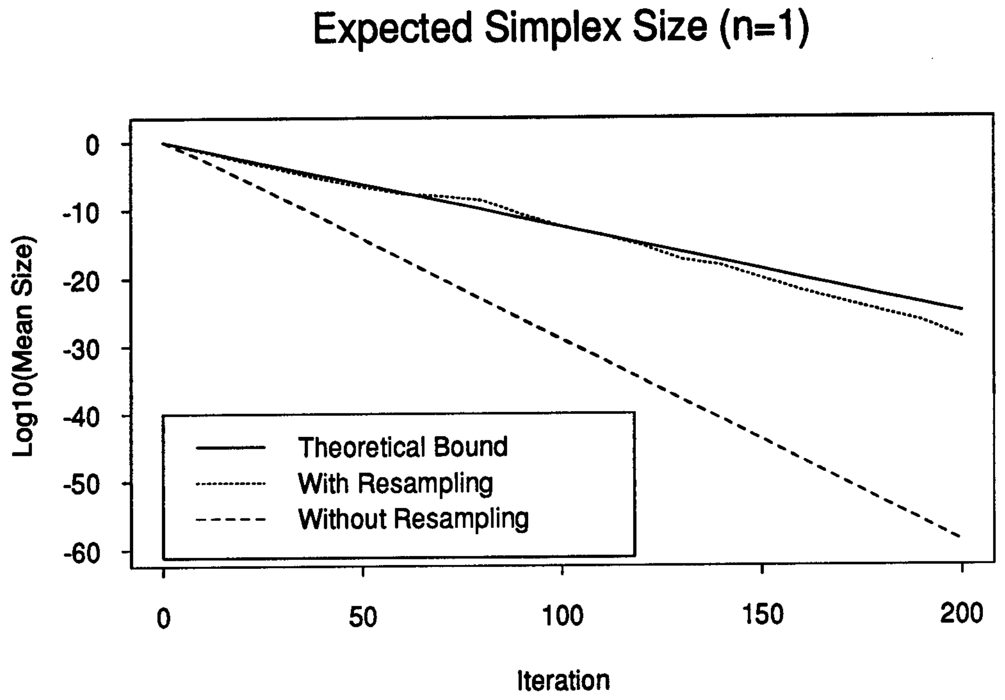
These simulations provide strong evidence for our conjecture when the initial simplex shape is regular as all other conditions of the simulation are not a factor, including the error distribution which was $N(0,1)$. That is, initial simplex size, initial simplex orientation and the distribution of the random noise have no effect on the convergence of the Nelder-Mead algorithm when the underlying function is constant. Furthermore, for an initial simplex that is not regular there exists an invertible linear transformation mapping it to a regular simplex (see Hensley et al., 1988). Therefore, our empirical results of simplex convergence are applicable even when starting with a nonregular simplex. Some interesting observations from the figures include the following:

1. In the top graph of Figure 3.2 notice how well the estimated mean simplex size with resampling approximates the geometric bound; whereas, it falls below the bound everywhere for $n > 1$ (see the top graph in Figures 3.3 - 3.6). This happens because a contraction is identical to a shrinkage when $n = 1$, and an expansion increases the simplex size by a factor less than 2 for $n > 1$.

2. Notice in the top graph of Figures 3.2 - 3.6 how the estimated mean size from the runs with resampling is an upper bound on the estimated mean size from the runs without resampling. Without resampling the probability of an expansion is decreasing while the probability of a shrinkage is increasing

until a shrinkage actually occurs. A shrinkage involves n new values while maintaining the lowest of the $n+1$ values in the simplex; therefore, a shrinkage nearly restores the first-iteration probabilities especially when n is large. This means that without resampling $E[\rho_i] < E[\rho_1]$; whereas, we have equality with resampling.

3. In the bottom graph of Figures 3.2 - 3.6 we plotted the estimates of $E[\alpha_i]$ and $E[\alpha_i \cdots \alpha_1]$ both with and without resampling. Since $\alpha_i = 1$ when $n \leq 2$, these plots are straight lines (see the bottom graphs of Figures 3.2 and 3.3). For $n > 2$ notice that $E[\alpha_i]$ appears to be approximately equal to 1 while the expectation of the product of these variables is decreasing (see the bottom graph of Figures 3.4 - 3.6).



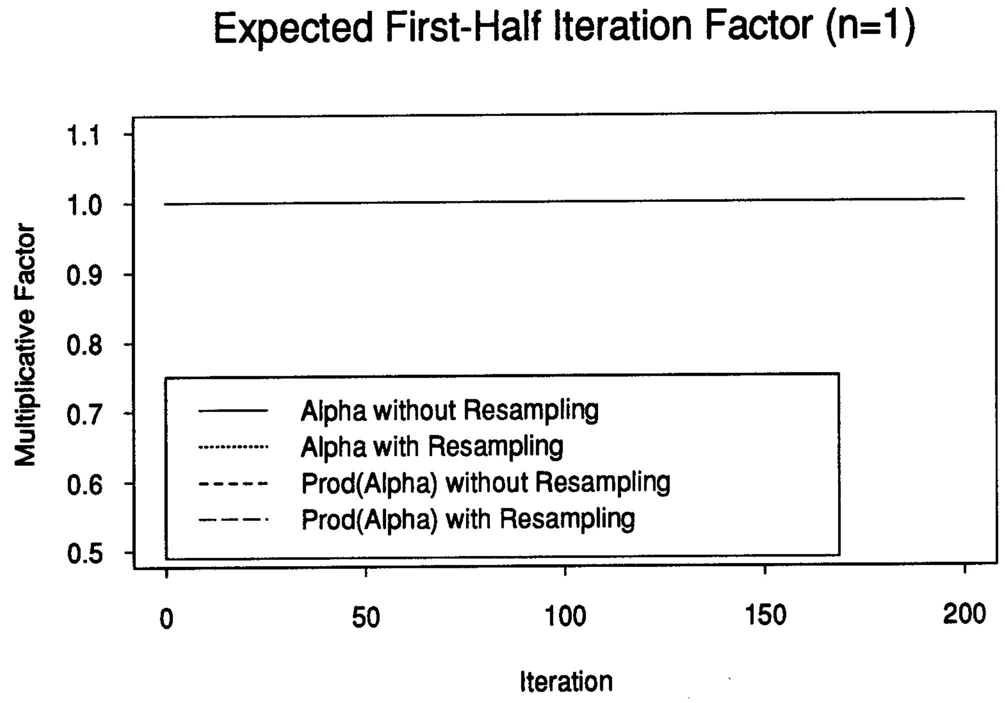
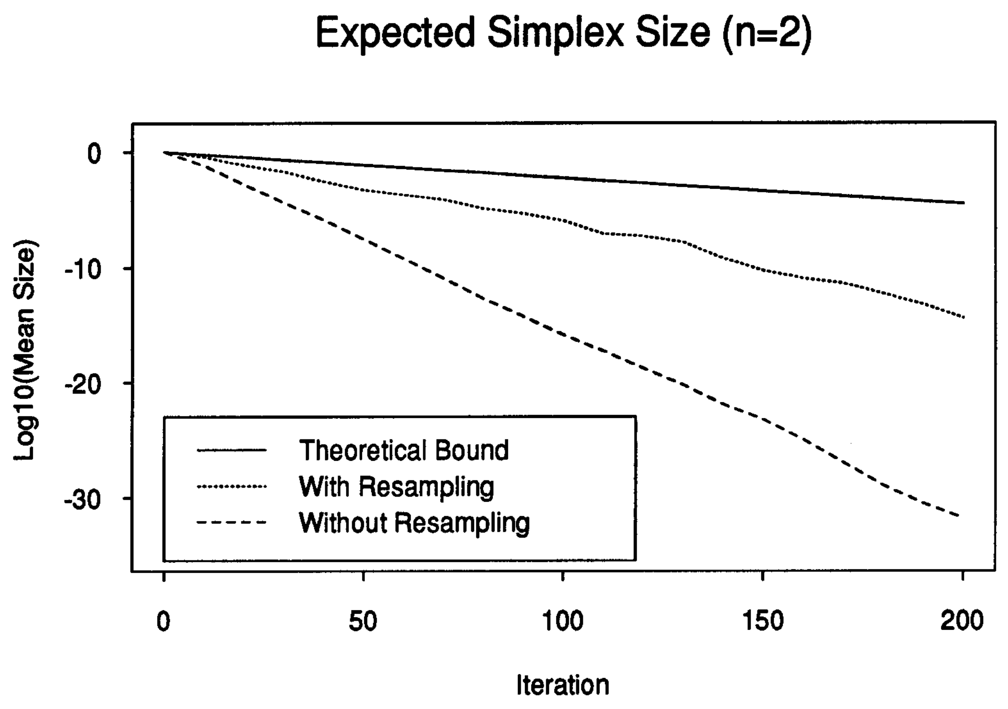


Figure 3.2: Empirical Evidence of Simplex Convergence ($n = 1$)



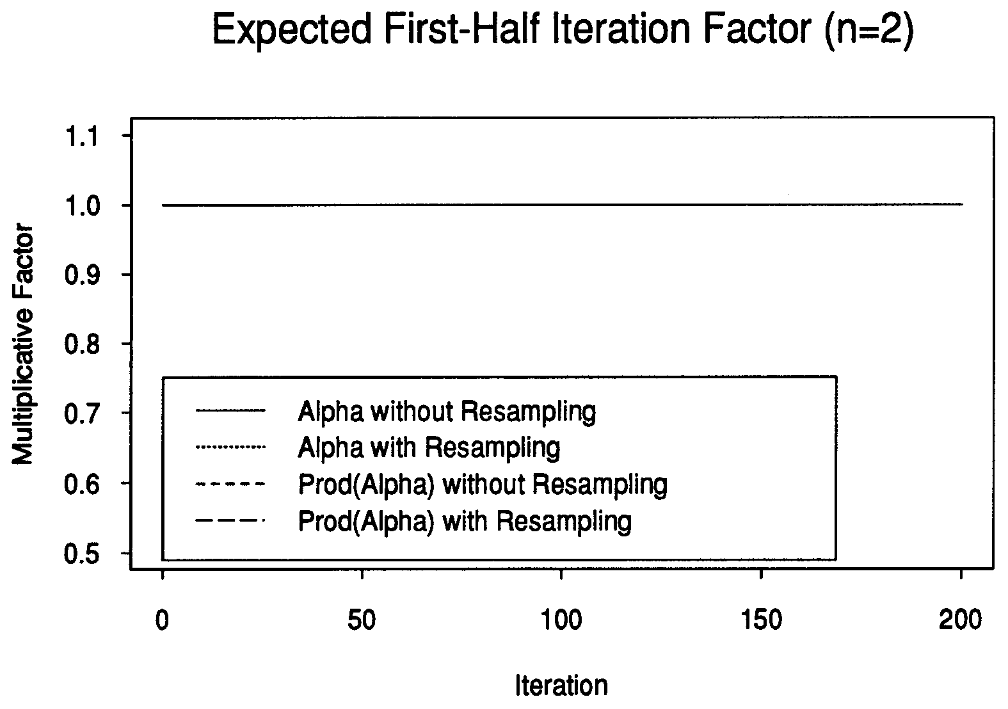
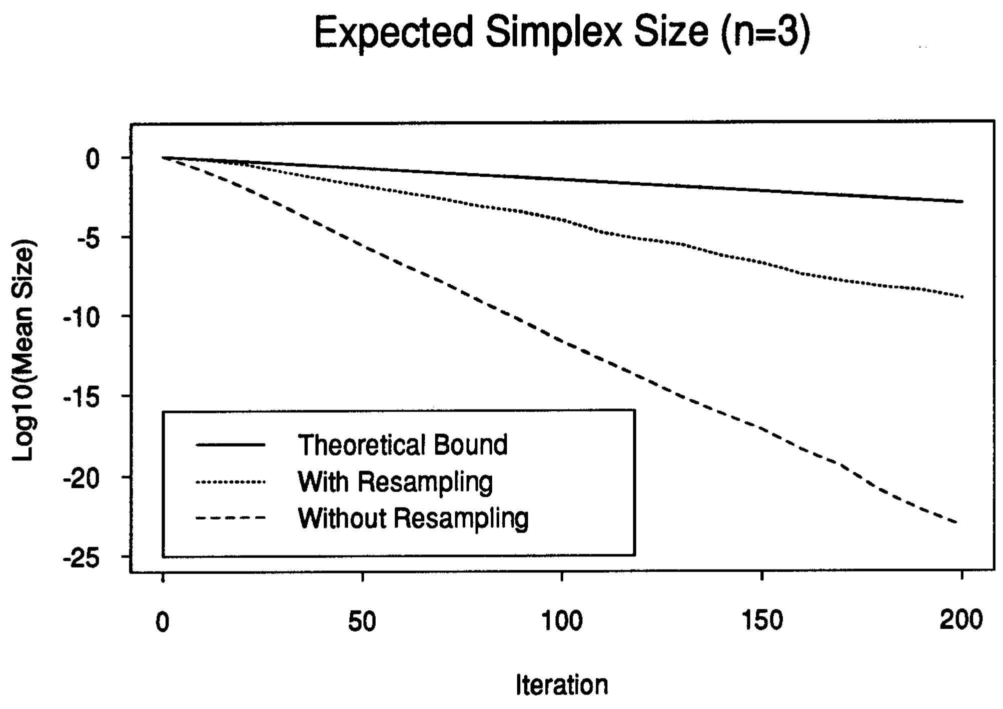


Figure 3.3: Empirical Evidence of Simplex Convergence ($n = 2$)



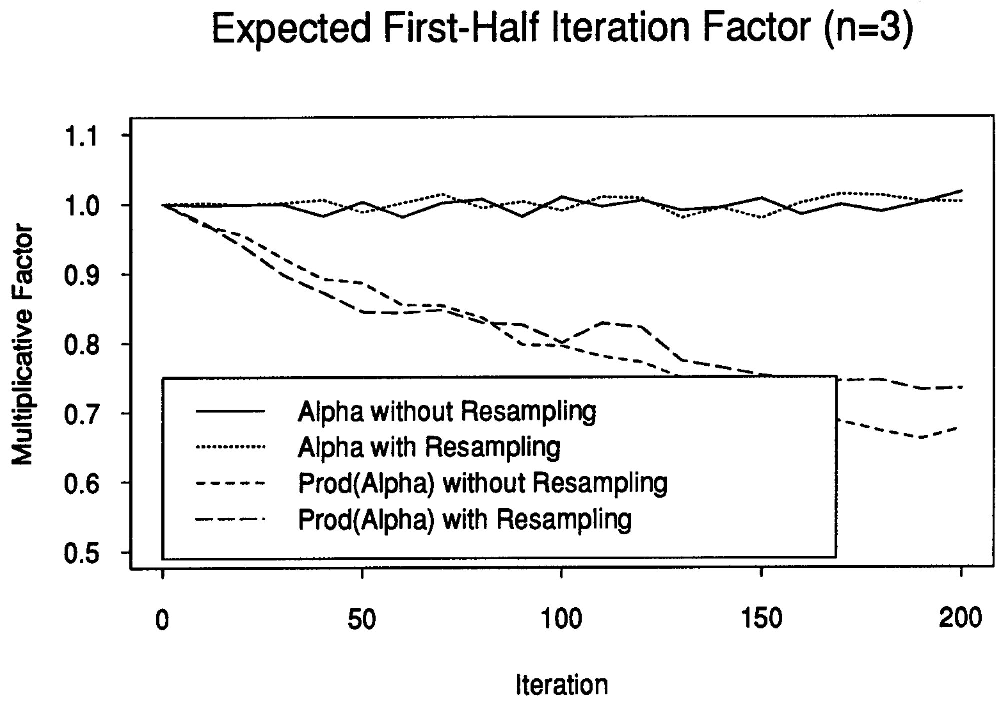
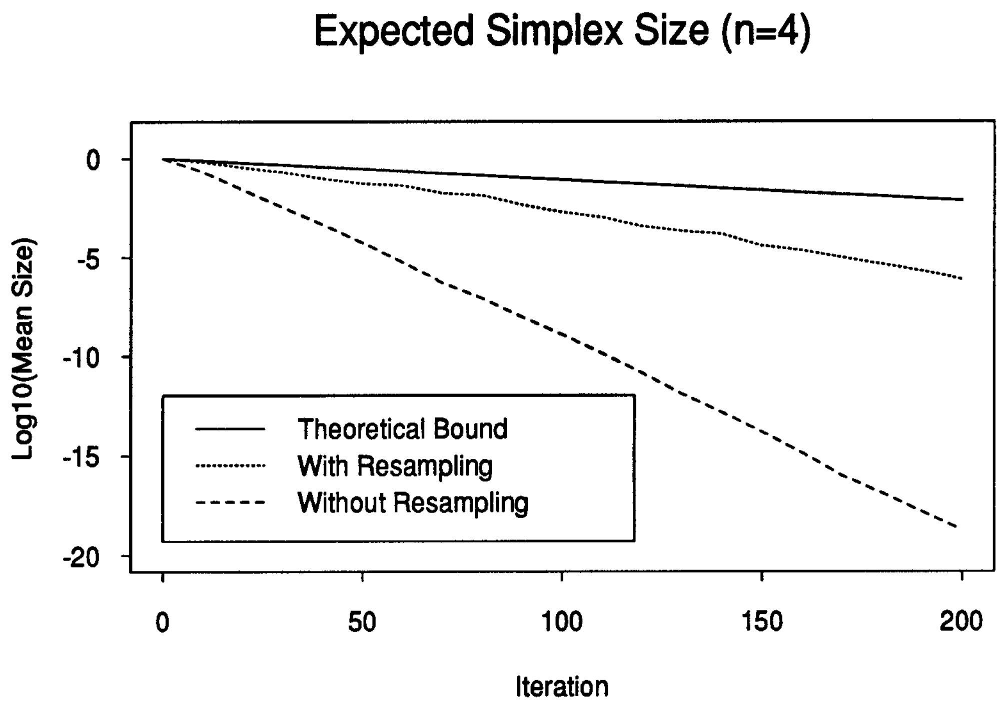


Figure 3.4: Empirical Evidence of Simplex Convergence ($n = 3$)



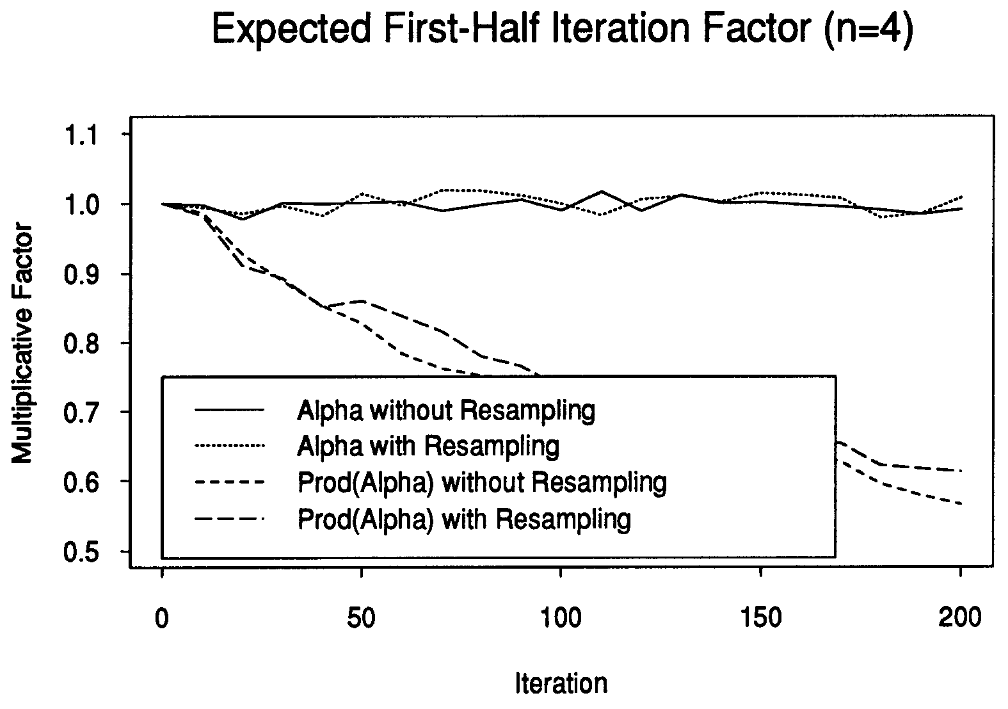
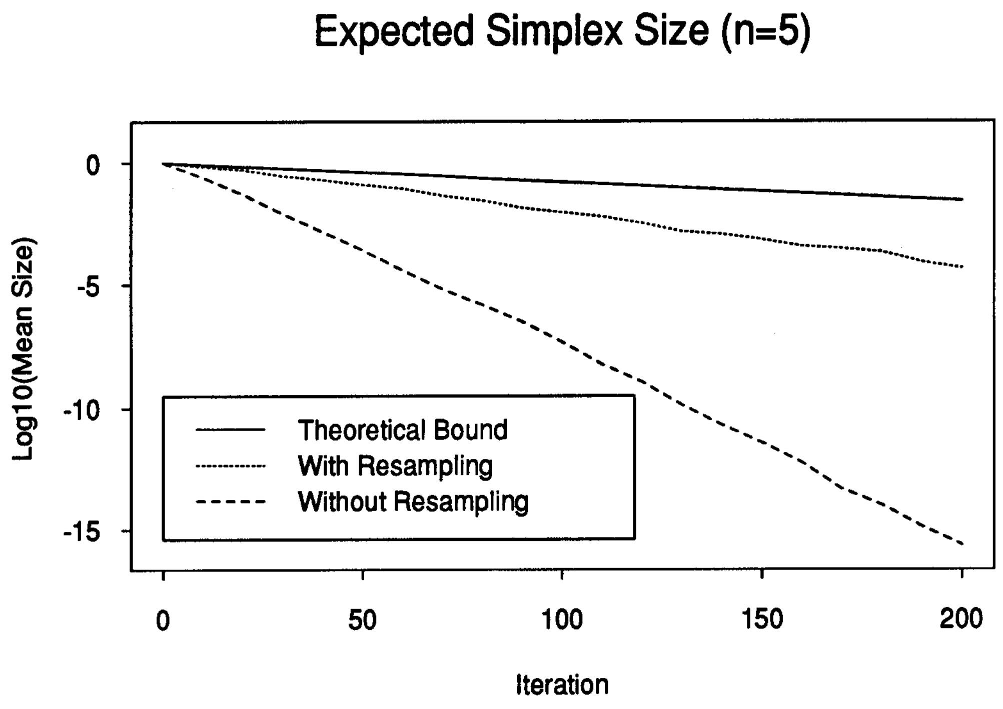


Figure 3.5: Empirical Evidence of Simplex Convergence ($n = 4$)



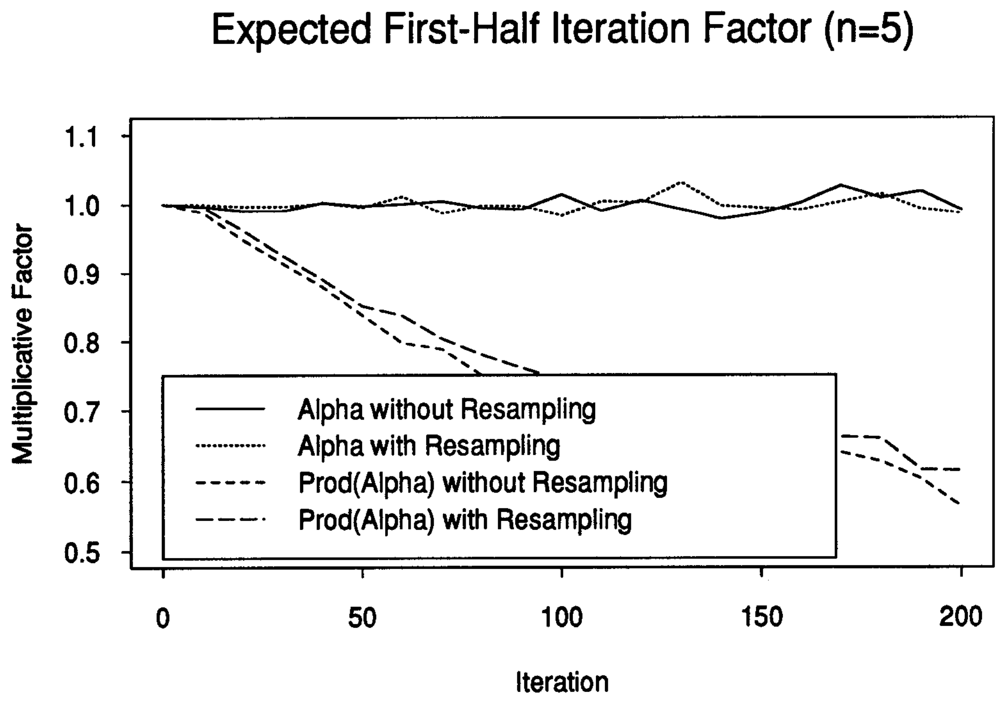


Figure 3.6: Empirical Evidence of Simplex Convergence ($n = 5$)

Chapter 4

Convergence on a One-Dimensional Linear Function with Noise

In Chapter 3 we proved that the simplex in the Nelder-Mead algorithm shrinks to a point *w.p. 1* on stochastic functions with constant expectation for $n \leq 2$, and we provided strong empirical evidence of the same result for $n > 2$. In this chapter we demonstrate both analytically and empirically that there is a positive probability that the simplex will shrink to a point on an unbounded function if noise is present. Finally, we propose a new modification to the Nelder-Mead simplex algorithm to deal effectively with the problem of false convergence.

4.1 Problem Definition

To simplify the analysis we assume a univariate linear function with additive $N(0,1)$ noise. With the introduction of slope, the convergence of the simplex to a point is no longer independent of the error distribution (as in the constant function case). Additionally, the orientation, size and shape of the simplex all have an effect upon the convergence. However, in the univariate problem there is only one orientation and only one shape for the simplex. We continue to use the strict acceptance rule for an expansion, and we consider resampling vertices needed for each comparison during an iteration rather than resampling all vertices once for each iteration. This is necessary to eliminate the dependency resulting from the slope of the linear function; that is, there is now an uphill and a downhill direction which affects the probability of an expansion, a reflection, etc. depending upon which direction the search is taking. Such a dependency requires the calculation of bivariate normal probabilities given a normal error distribution, which is an unnecessary complication.

Without noise the simplex, which is a line segment in the univariate problem, will double in size with every iteration as it proceeds downhill. With the addition of noise there is a positive probability that the simplex will search uphill and that the search will lead to a shrinkage or halving of the line segment. This probability increases for an increase in the variance of the noise, a decrease in the magnitude of the slope or a decrease in the size of the simplex. Under the normal probability model for the error distribution, these three factors can be combined to form a single variable—termed the signal-to-noise ratio—for calculating the probabilities of an expansion, a reflection or a shrinkage, which we demonstrate next in Section 4.2.

4.2 One-Step Probabilities

Consider Figure 4.1 in which we assume a linear function g with slope $\beta < 0$ and intercept β_0. Therefore, $g(x_{j+1}) = \beta_0 + \beta x_{j+1} < \beta_0 + \beta x_j = g(x_j)$ for $x_{j+1} > x_j$. Let $Y_{ij} = Y_i(x_j)$ be an observation of the function at x_j obscured by noise $\epsilon_{ij} \sim N(0, \sigma^2)$. Furthermore, let ϵ_{ij} be independent of ϵ_{kl} whenever $i \neq k$ or $j \neq l$. Then the Y_{ij}'s are independent and

$$Y_{ij} \sim N(\beta_0 + \beta x_j, \sigma^2). \tag{4.1}$$

From normal probability theory we know that the difference between any two of these random variables is also normally distributed. For example, by Equation 4.1 we know that

$$Y(x_{j+1}) - Y(x_j) \sim N(\beta(x_{j+1} - x_j), 2\sigma^2), \tag{4.2}$$

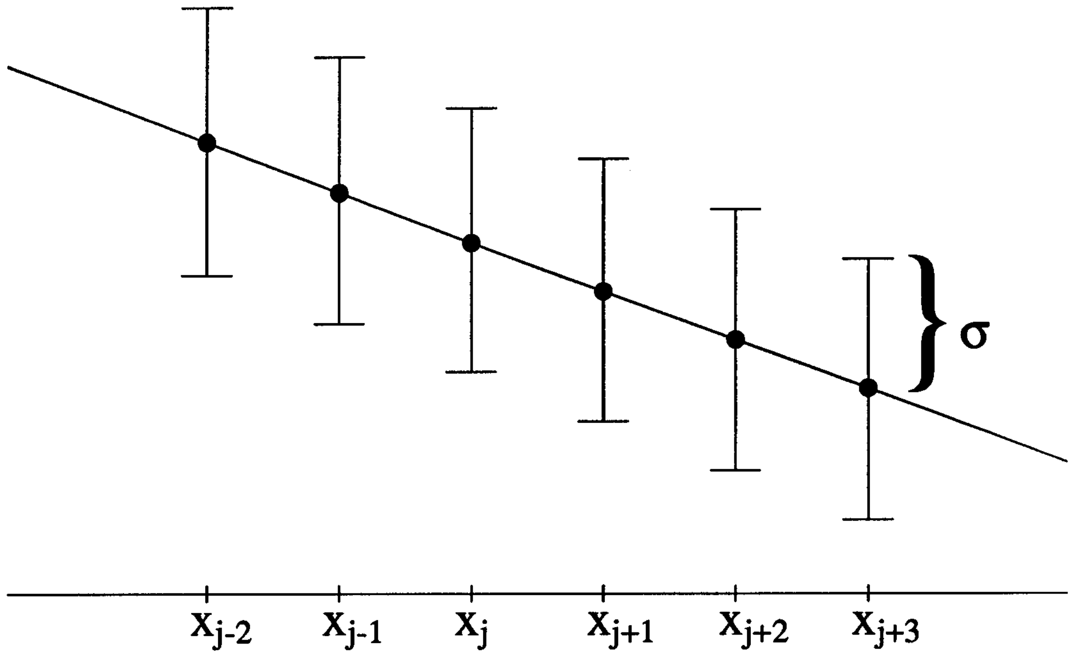


Figure 4.1: Univariate Linear Function with Noise

where we have dropped the subscript i with the understanding that each observation at a point in the parameter space includes a draw from the error distribution. Let the current simplex be defined by the two points x_j and x_{j+1}. By Equation 4.2 the probability that the direction of the search will be downhill is as follows:

$$\begin{aligned} P\{Y(x_{j+1}) < Y(x_j)\} &= P\{Y(x_{j+1}) - Y(x_j) < 0\} \\ &= P\left\{\frac{Y(x_{j+1}) - Y(x_j) - \beta(x_{j+1} - x_j)}{\sigma\sqrt{2}} < \frac{-\beta(x_{j+1} - x_j)}{\sigma\sqrt{2}}\right\} \\ &= P\left\{Z < \frac{-\beta(x_{j+1} - x_j)}{\sigma\sqrt{2}}\right\} \\ &= \Phi\left(\frac{-\beta(x_{j+1} - x_j)}{\sigma\sqrt{2}}\right), \qquad (4.3) \end{aligned}$$

Table 4.1: Probabilities of Simplex Operations on a Univariate Linear Function observed with $N(0, \sigma^2)$ Errors and Resampling for Each Comparison

Operation	Downhill	Uphill	Total
Expansion	$\Phi(a)^3$	$(1 - \Phi(a))^3$	$1 - 3\Phi(a) + 3\Phi(a)^2$
Reflection	$\Phi(a)^2(1 - \Phi(a))$	$(1 - \Phi(a))^2\Phi(a)$	$\Phi(a) - \Phi(a)^2$
Shrinkage	$\Phi(a)(1 - \Phi(a))$	$(1 - \Phi(a))\Phi(a)$	$2\Phi(a) - 2\Phi(a)^2$

where $Z \sim N(0,1)$ and Φ is the cumulative distribution function (cdf) of a standard normal random variable. In our example $\beta < 0$ and $x_{j+1} > x_j$, yielding a positive operand for the cdf. In fact, when calculating the probability of searching downhill, the smallest this operand can be physically is 0, which occurs when $\beta = 0$ for a nondegenerate simplex. Therefore, the operand is always nonnegative for the probability of searching downhill. Given a simplex consisting of any two points $\{x_i, x_j\}$, let $d = |x_i - x_j|$ be the size of the simplex. Then, Equation 4.3 can be expressed as $P(\textit{search downhill}) = \Phi(a)$, where $a = |\beta d|/(\sigma\sqrt{2})$. Assuming independence for all probability calculations we may derive formulas in terms of $\Phi(a)$ for the probability of each simplex operation. Summing the uphill and downhill probabilities for each operation yields the total probability for that simplex operation. As discussed in Chapter 3, a contraction has the same effect on the simplex as a shrinkage and a failed expansion has the same effect as a reflection. Furthermore, no immediate reflections are possible in $\Re^1$. The probabilities for the simplex operations on a linear function in $\Re^1$ are summarized in Table 4.1.

4.3 Markov Chain Analysis

Suppose we chose a value for the slope (β) and the scale parameter (σ). Next, let X_k be a state variable on the integers (i) such that the size of the simplex at the conclusion of iteration k equals $2^i d_0$, where d_0 is the initial simplex size. An expansion doubles the size of the simplex, a shrinkage halves the size of the simplex, and a reflection leaves the size unchanged. Therefore, there are three possible states that can be reached from the current state. Absent any stopping rule our state variable defines a doubly infinite random walk under Markov chain theory. The one-step transition probability matrix for the general random walk is given below.

$$\begin{array}{c|ccccccccccc|}
 & \cdots & -4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & \cdots \\
\hline
\vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
-2 & \cdots & 0 & q_{-2} & r_{-2} & p_{-2} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\
-1 & \cdots & \cdots & 0 & q_{-1} & r_{-1} & p_{-1} & 0 & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & 0 & q_0 & r_0 & p_0 & 0 & \cdots & \cdots & \cdots \\
1 & \cdots & \cdots & \cdots & \cdots & 0 & q_1 & r_1 & p_1 & 0 & \cdots & \cdots \\
2 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & q_2 & r_2 & p_2 & 0 & \cdots \\
\vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots
\end{array}$$

where the one-step transition probabilities follow from the total probabilities given in Table 4.1.

$$\begin{aligned}
p_i &= P\{X_{k+1} = i+1 | X_k = i\} \\
&= P\{\text{Expand} | d_k = 2^i d_0\} \\
&= 1 - 3\Phi\left(\frac{|\beta|\, 2^i d_0}{\sigma\sqrt{2}}\right) + 3\left[\Phi\left(\frac{|\beta|\, 2^i d_0}{\sigma\sqrt{2}}\right)\right]^2 .
\end{aligned} \tag{4.4}$$

$$\begin{aligned} r_i &= P\{X_{k+1} = i | X_k = i\} \\ &= P\{\text{Reflect} | d_k = 2^i d_0\} \\ &= \Phi\left(\frac{|\beta|\, 2^i d_0}{\sigma\sqrt{2}}\right) - \left[\Phi\left(\frac{|\beta|\, 2^i d_0}{\sigma\sqrt{2}}\right)\right]^2 . \qquad (4.5) \\ q_i &= P\{X_{k+1} = i-1 | X_k = i\} \\ &= P\{\text{Shrink} | d_k = 2^i d_0\} \\ &= 2\Phi\left(\frac{|\beta|\, 2^i d_0}{\sigma\sqrt{2}}\right) - 2\left[\Phi\left(\frac{|\beta|\, 2^i d_0}{\sigma\sqrt{2}}\right)\right]^2 . \qquad (4.6) \end{aligned}$$

4.3.1 A Gambler's Ruin Example

A gambler's ruin problem is a special type of random walk with an absorbing state at each end. This type of problem was named after the classical type of gaming situation where the player wants to know the probability of ruin (running out of money) before reaching a certain total dollar amount (starting money plus net winnings) given the amount of money he has at the start of the game. The states represent the total amount of money the player has and the one-step transition probabilities are the probabilities of losing (q_i), winning (p_i) or breaking even (r_i) if possible. From these probabilities one can calculate the probability of ruin ($X_T = 0$) given the starting state (X_0) using Equations 4.7 and 4.8.

$$\begin{aligned} P\{X_T = 0 | X_0 = i\} &= \frac{\rho_i + \ldots + \rho_N}{1 + \rho_1 + \ldots + \rho_N}, \qquad (4.7) \\ \text{where } \rho_k &= \frac{q_1 q_2 \cdots q_k}{p_1 p_2 \cdots p_k} \qquad (4.8) \end{aligned}$$

and state N represents the state at which the player would quit if his total amount of money reached N dollars. These equations can be found in most textbooks on stochastic processes such as the one by Taylor and Karlin (1984).

We use this type of problem configuration to demonstrate that there is a positive probability of the simplex shrinking to a point on an unbounded linear function with noise. First, we must select our absorbing states for this analysis. Given a fixed slope and a fixed scale parameter for the normal error distribution, the one-step transition probabilities approach the constant function case ($\beta = 0$) as the simplex decreases in size. The one-step transition probabilities for the constant function situation are $q_i = 0.5, r_i = 0.25$ and $p_i = 0.25$. If we let $\beta = -1$, $\sigma = 1$ and $d_0 = \sqrt{2}$, then the one-step transition probabilities equal those of the constant function case to six decimal places for states $i \leq -10$. Therefore, we selected state $i = -11$ to be the absorbing state representing convergence of the simplex to a point (the state of ruin for the gambler).

Similarly, as the simplex size increases the one-step transition probabilities approach the no-noise situation for which $q_i = 0, r_i = 0$ and $p_i = 1$. Given our selection of the parameters β, σ and d_0 above, the one-step transition probabilities equal those of the no-noise case to six decimal places for states $i \geq 3$. Therefore, we selected state $i = 3$ to be the absorbing state representing divergence of the simplex to infinite size. Using Equations 4.4 - 4.8 all of the probabilities are easily calculated and are summarized in Table 4.2.

Note that if we were to include more states before declaring the absorbing state for convergence where each of those states would have one-step transition probabilities identical to those of state $i = -10$, then the probabilities of reaching the state of convergence (the last column in the table) would be slightly smaller. Also, any changes in the parameters β, σ or d_0 would essentially cause a renumbering of the states (the first column in the table) without significant changes to the probabilities. In conclusion, this example illustrates the problem that the simplex in the Nelder-

Table 4.2: Gambler's Ruin Example

State (i)	q_i	r_i	p_i	$P\{X_k = -11 \mid X_0 = i\}$
-11	1.000000	0.000000	0.000000	1.000000
-10	0.500000	0.250000	0.250000	0.999391
-9	0.499999	0.249999	0.250002	0.998172
-8	0.499995	0.249998	0.250007	0.995734
-7	0.499981	0.249990	0.250029	0.990858
-6	0.499922	0.249961	0.250117	0.981108
-5	0.499689	0.249845	0.250466	0.961620
-4	0.498758	0.249379	0.251863	0.922742
-3	0.495052	0.247526	0.257422	0.845751
-2	0.480514	0.240257	0.279229	0.697690
-1	0.426684	0.213342	0.359974	0.442898
0	0.266968	0.133484	0.599549	0.140887
1	0.044465	0.022233	0.933302	0.006407
2	0.000063	0.000032	0.999905	0.000000
3	0.000000	0.000000	1.000000	0.000000

Mead algorithm can converge to a single point with positive probability even on an unbounded function if noise is present.

4.3.2 Empirical Results of False Convergence

The operand of the normal cdf in Equation 4.3 may be thought of as a signal-to-noise ratio for detecting a search direction for the next iteration. When this ratio is large, the probability of searching in the correct direction (downhill for minimization) is high. Likewise, as this operand approaches zero, the probability of searching in the correct direction approaches 0.50. By resampling the points involved in each comparison, we obtained the necessary independence to be able to use Markov chain theory. Of course, Nelder-Mead does not employ any resampling and we suspected that the resulting dependency causes the probability of convergence to be larger than what we had obtained with our analytical approach in Table 4.2.

To test this conjecture we simulated Nelder-Mead on a linear function with slope $\beta = -1$ and additive $N(0,1)$ noise. The initial simplex size was selected to yield a probability of searching downhill equal to $0.525, \ldots, 0.975$. Additional values of 0.9938, 0.9987, 0.9998, and 0.9999 were included to cover the range of the analytical results. The respective percentiles provide the starting signal-to-noise ratio. The probability of simplex convergence given the initial signal-to-noise ratio was estimated from 10,000 replications of the simulation. Each replication was terminated when the simplex size reached either of the two absorbing states from the Gambler's Ruin example (*i.e.* $2^{-11}\sqrt{2}$ for convergence or $2^{3}\sqrt{2}$ for divergence). Furthermore, to verify our analytical results we repeated the simulation experiment with resampling. The empirical results, both with and without resampling, along with our analytical results are graphed in Figure 4.2.

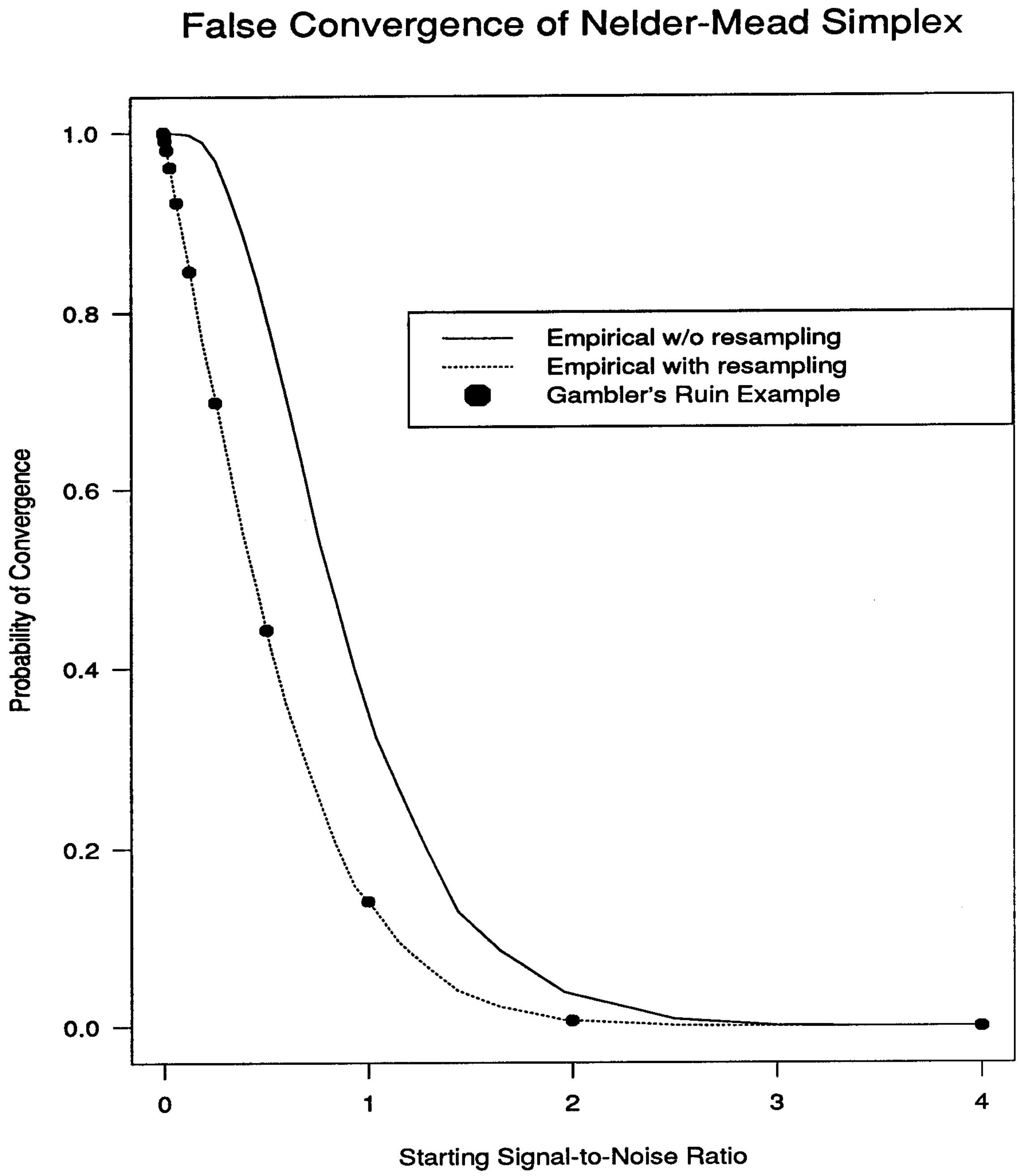


Figure 4.2: Probability of Convergence of the Nelder-Mead Simplex versus Starting Signal-to-Noise Ratio on a One-Dimensional Linear Function with Noise

4.3.3 Preventing Gambler's Ruin

To prevent the simplex from shrinking to a point on an unbounded function with noise, the probability of an expansion must be at least as large as the probability of a shrinkage ($p_i \geq q_i$) for most states i. This could be accomplished by adjusting any one of the three parameters that affect these probabilities. The one parameter that we can effectively adjust in a practical situation is the scale parameter (σ) of the error distribution. This is done by adjusting the number of observations taken at each vertex and each new trial point. By the weak law of large numbers we know that sample mean converges in probability to the expected value; that is,

$$\frac{1}{m}\sum_{i=1}^{m} Y_i(x_j) \xrightarrow{P} E[Y(x_j)],$$

where m is the sample size. More importantly, as m increases, the magnitude of the observed noise decreases. That is, $\sigma^2/m \to 0$ as $m \to \infty$. So, the larger the sample size (m) of observations taken at the point x_j, the smaller the effective noise will be when estimating the function's value at x_j. Since a decrease in the simplex size lowers the signal-to-noise ratio, a modified Nelder-Mead simplex algorithm must increase the sample size of observations taken at each vertex as the simplex size decreases to ensure further progress towards the minimum of the expected function. However, we must choose a rule for increasing the sample size wisely so as to keep the cost of the next iteration as low as possible.

For example, suppose $p_i = q_i$ for the one-step transition probabilities in the general random walk. Then the simplex size would neither converge nor diverge—it would be a null recurrent Markov chain. To solve for this condition we simply equate Equations 4.4 and 4.6 and solve for the zeroes of the resulting quadratic equation.

The two zeroes are

$$\Phi\left(\frac{|\beta|\,2^i d_0}{\sigma\sqrt{2/m}}\right) = \frac{5 \pm \sqrt{5}}{10} = 0.2763932, 0.7236068.$$

Note that the first solution is not physically possible as the operand must be non-negative. The percentile corresponding to the second solution is 0.59359. Therefore, using the same values for each of the parameters used in the gambler's ruin example ($\beta = -1, \sigma = 1, d_0 = \sqrt{2}$) we may solve for the sample size that will give us the percentile 0.59359 for each state. For example, for state $i = -2$

$$m = \left(\frac{0.59359\,\sigma\sqrt{2}}{|\beta|\,2^i d_0}\right)^2 = 5.637585,$$

which means we would need to take at least 6 observations to guarantee that the probability of expanding the simplex is at least as large as the probability of shrinking the simplex. Such a rule seems easy enough except when one realizes that β and σ are unknown for any practical problem; and the form of the expected function is unknown as well (*e.g.* linear, quadratic, concave, convex, etc.). We can estimate σ with a preliminary experiment and assume that the function is linear across the simplex, but β must be estimated for every iteration of the optimization algorithm since the underlying expected function is unknown. As a result, our solution for m may be too large if the estimate for β is poor. Therefore, we will want to increase the sample size in a more systematic way as a proportion of the current sample size. This motivates our proposed modification to the Nelder-Mead simplex algorithm.

4.4 Proposed Modification to Nelder-Mead Simplex Algorithm

Assume a similar model for the noise-corrupted observations Y_{ij} as we did in Equation 4.1 except now the unknown continuous function (g) is not linear but has a unique minimum. Then the model for new noise-corrupted observations of g at each of the two vertices x_1 and x_2 of simplex is

$$Y_{ij} \sim N(\mu_j, \sigma^2), \tag{4.9}$$

where $\mu_j = g(x_j)$. Let m_j be the sample size of observations at vertex x_j which are collected at the start of an iteration. Next, estimate $g(x_j)$ by the sample mean of the observations. Then

$$\overline{Y}_{.j} \;=\; \frac{1}{m_j}\sum_{i=1}^{m_j} Y_{ij} \sim N\left(\mu_j, \frac{\sigma^2}{m_j}\right). \tag{4.10}$$

If we increase m_j we obtain a better estimate of $\mu_j = g(x_j)$. And, better estimates of μ_j increases the probability of performing the same simplex operation that would have taken place if no noise were present. However, if this probability is large already, then more observations are probably unnecessary and wasteful. In fact, we may want to decrease the number of observations taken at the next iteration if this probability is sufficiently large enough at the current iteration. As indicated by the empirical results in Section 4.3.2 the probability of false convergence is dependent upon the probability of searching in the right direction (downhill for minimization).

4.4.1 Adjusting Sample Size Based on a Hypothesis Test

Given our model we can perform the familiar two-sample two-tailed hypothesis test for determining whether to increase or decrease the number of observations at the trial points of the current iteration. The null and alternative hypotheses are

$$\begin{aligned} H_0 &: \;\; \mu_1 = \mu_2 \\ H_1 &: \;\; \mu_1 \neq \mu_2 \end{aligned} \tag{4.11}$$

and the test statistic assuming we know σ is

$$T_1 = \frac{\overline{Y}_{.2} - \overline{Y}_{.1}}{\sigma\sqrt{1/m_2 + 1/m_1}}. \tag{4.12}$$

The distribution of the test statistic T_1 under each hypothesis is given in Equation 4.13.

$$\begin{aligned} &\text{Under } H_0\text{:} \quad T_1 \sim N(0,1) \\ &\text{Under } H_1\text{:} \quad T_1 \sim N\left(\frac{\mu_2 - \mu_1}{\sigma\sqrt{1/m_2 + 1/m_1}}, 1\right) \end{aligned} \tag{4.13}$$

The mean of the distribution under H_1 is the true signal-to-noise ratio for determining the search direction of the next simplex reflection. And, $|T_1|$ is an estimator for this ratio.

If σ^2 is unknown, then it is usually estimated by the pooled estimate of the sample variances given by Equations 4.14 and 4.15:

$$s_p^2 = \frac{(m_1 - 1)s_1^2 + (m_2 - 1)s_2^2}{m_1 + m_2 - 2}, \tag{4.14}$$

where

$$s_j^2 = \frac{1}{m_j - 1} \sum_{if=1}^{m_j} \left(Y_{ij} - \overline{Y}_{.j}\right)^2. \tag{4.15}$$

The test statistic then becomes

$$T_2 = \frac{\overline{Y}_{.2} - \overline{Y}_{.1}}{s_p\sqrt{1/m_2 + 1/m_1}} \tag{4.16}$$

with distributions

$$\begin{aligned} &\text{Under } H_0\text{:} \quad T_2 \sim t_\nu \\ &\text{Under } H_1\text{:} \quad T_2 \sim t_\nu\left(\frac{\mu_2 - \mu_1}{\sigma\sqrt{1/m_2 + 1/m_1}}\right) \end{aligned} \tag{4.17}$$

where t_ν is Student's t-distribution on ν degrees of freedom. For test statistic T_2, $\nu = (m_1 + m_2 - 2)$.

Typically one assumes the null hypothesis to be true unless there is sufficient evidence to reject it. This is the rationale behind choosing a high level of significance (*e.g.* $\alpha = 0.05$ or 0.01). In our use of this test one would believe the alternative hypothesis to be true most of the time. Therefore, we may not want such a high significance level. On the other hand, we may want a high significance level to force more observations to be taken in order to increase the probability of searching in the correct direction. Recall that we reject the null hypothesis if $|T_1| > z^{\alpha/2}$, where $z^{\alpha/2}$ is the $100(1 - \alpha/2)$ percentile of the standard normal distribution. If we fail to reject the null hypothesis, then it is likely that we have not taken enough observations at each vertex to be able to detect the correct search direction with a sufficiently high enough probability. Therefore, the result of failing to reject the null hypothesis leads to an increase in the sample size of observations at each vertex and at each new trial point. And, rejection of the null hypothesis means that the signal-to-noise ratio is significantly large enough to detect the correct direction for the next search. Hence, we may decrease the sample size of observations taken at new trial points. This idea forms the basis of a new simplex method for stochastic optimization founded on the simplex operations of the Nelder-Mead algorithm.

The theory behind our proposed algorithm is valid if the observations at each vertex are independent and identically distributed. To maintain this condition of iid samples we must discard all observations at each vertex following every iteration of the simplex algorithm. Of course, this could be expensive in a practical situation. Now, the condition of iid samples is violated by retaining observations from previous iterations because the simplex operations cause vertices with low means to be retained while vertices with high means are discarded. Assume that the simplex undergoes a series of reflections and observations at the retained vertex are not resampled. Then under the null hypothesis the expected range of the two means will decrease with each successive reflection. Other simplex operations are more complex to analyze, but they all have the same effect of reducing the expected range of the means if resampling is not employed. Therefore, the test would be more likely to fail to reject the null hypothesis leading to an increase in the sample size at each vertex. So, it appears that it will cost more observations in practice whether we resample the vertices at each iteration or not. One possible way to avoid higher costs would be to include an adjustable critical value for the test. That is, suppose we resampled the low vertex only after a shrinkage. Then for the first iteration following a shrinkage we would have iid samples at the two vertices. For each iteration following the shrinkage the critical value could be reduced to account for the phenomenon we have just described when resampling is not employed. How this would be accomplished is not exactly certain, and such a rule may be an unnecessary complication if the algorithm is fairly robust with respect to the choice of a critical value.

4.4.2 Sample Size Increase Factor

Another issue to be determined is the factor for increasing the sample size should the hypothesis test fail. Earlier in Section 4.3.3 we suggested increasing the sample size by a multiple of the current sample size. That is, if $m^k = \min\{m_1, m_2\}$ following iteration k and we fail to reject the null hypothesis, then we suggest letting $m^{k+1} = bm^k$ where $b > 1$. For any reasonable choice of b the test could still fail to reject the null hypothesis with new samples of size bm^k. Therefore, we might be tempted to repeat the test until we reject the null hypothesis before continuing with the simplex operation. Alternatively, the minimum sample size required to satisfy rejection at a particular α-level could be determined *a priori*. However, we do not recommend either of these procedures, because the difference in the true means might not be that different from zero. This could happen on a fairly level portion of the curve, near the optimum, or if the current simplex was straddling the optimum. Instead, we recommend performing the hypothesis test once each iteration, and should the test fail to reject H_0 we recommend increasing the sample size geometrically. Additionally, we recommend setting b large enough so that the test would not fail to reject H_0 forever—a situation that would probably result in simplex convergence prior to reaching a local optimum. For a small simplex we can assume that the function is linear across the simplex. The worst case scenario that could result from having too small a sample size is an inappropriate shrinkage of the simplex. To improve our chances of rejecting H_0 following a shrinkage we should select $b > \delta^{-2}$ where δ is the shrinkage coefficient. If b satisfies this condition, then the signal-to-noise ratio is guaranteed to increase on a linear function following a failed test no matter what simplex operation followed.

In Chapter 5 we report the results of a test of our proposed algorithm using Monte Carlo simulation on a set of one-dimensional test functions. Also, we examine the sensitivity of the algorithm to the choice of the factor b and the choice of the critical value for the hypothesis test.

Chapter 5

Testing of Proposed Algorithm on Univariate Functions

5.1 Purpose

We ran the following experiment to investigate the performance of our proposed algorithm for minimizing a univariate stochastic function. For our test we assume that the random response $Y(x)$ is the sum of an unobservable continuous real function g having a unique minimum and an additive iid noise component:

$$Y(x) = g(x) + \epsilon, \tag{5.1}$$

where $\epsilon \sim N(0, \sigma^2)$. We compare the performance of our new algorithm to the original Nelder-Mead algorithm, the best performing simplex algorithm tested by Barton and Ivey (1993) and the Kiefer-Wolfowitz stochastic approximation method which is known to converge *w.p.1* (see Kiefer & Wolfowitz, 1952; Venter, 1967).

5.2 The Design Frame

In this section we define the various algorithms we tested, the set of test functions for the real continuous function g, the measure of performance and the starting values for the algorithms.

5.2.1 Optimization Codes

The following algorithms were implemented in FORTRAN and run on a Sun SPARCstation for the comparative evaluation. The original Nelder-Mead simplex algorithm and each of its variants employ the strict acceptance rule for an expansion as well as the standard coefficients for simplex operations unless otherwise stated.

NM: The Nelder-Mead simplex algorithm.

RS9: The Nelder-Mead simplex algorithm except that the shrinkage coefficient is set to 0.9 instead of 0.5; and furthermore, the best vertex is resampled upon a shrinkage. This was the best performing Nelder-Mead variant among those tested by Barton and Ivey (1993).

NMSN: Our proposed Nelder-Mead algorithm employing a signal-to-noise ratio. Additionally, we use the higher shrinkage coefficient and resampling of the best vertex upon shrinkage as in algorithm RS9. Let m^k be the minimum number of observations taken at each new trial point during the k^{th} iteration. Then the number of observations taken at each new trial point during the $(k+1)^{st}$ iteration is determined by the following rule:

$$m^{k+1} = \begin{cases} \lfloor b m^k \rfloor & \text{if } \dfrac{\overline{Y}_{.2} - \overline{Y}_{.1}}{\sigma\sqrt{1/m_2 + 1/m_1}} \leq z^{\alpha/2} \\ \lfloor m^k / b \rfloor & \text{if } \dfrac{\overline{Y}_{.2} - \overline{Y}_{.1}}{\sigma\sqrt{1/m_2 + 1/m_1}} > z^{\alpha/2} \end{cases} \tag{5.2}$$

where the operation $\lfloor u \rfloor$ yields the smallest integer greater than or equal to u. Additionally, if the rule yields $m^{k+1} > m^k$, then new observations are taken at each vertex x_i if $m_i < m^{k+1}$. This is done so that all vertices have at least m^{k+1} observations before performing the $(k+1)^{st}$ iteration. Note that we do not need to take the absolute value of the test statistic because the simplex algorithm sorts the vertices in ascending order according to the observed function values (sample means). Comparisons with the other algorithms are made using $b =$

1.25 which is slightly greater than 0.9^{-2} and $z^{\alpha/2} = 1.96$ corresponding to $\alpha = 0.05$. Additional runs were made to test the sensitivity of these two parameters.

KW: The Kiefer-Wolfowitz stochastic approximation algorithm for minimizing a function. A starting value x_0 must be provided by the user. Then the recursive relationship for sampling a new point in the parameter space is given by Equation 5.3.

$$x_{i+1} = x_i - a_i \left(\frac{Y(x_i + c_i) - Y(x_i - c_i)}{2c_i} \right), \tag{5.3}$$

where $a_i = i^{-1}$ and $c_i = i^{-1/3}$.

5.2.2 Test Functions

The selection of a set of test functions is a critical step in a comparative analysis of minimization algorithms (see Barton, 1987, 1984; Jackson, Boggs, Nash, & Powell, 1991). For the real deterministic function g we selected 12 functions representative of the class of continuous real functions with a unique minimum. Some of the selected function types have appeared in previous studies. For example, Barton (1980) tested an absolute value function and a simple quadratic function. Test functions G7 and G8 were inspired by a cost minimization example found in Nash (1979). Furthermore, the set includes the reflection of each of the nonsymmetric functions as the starting values were selected to the right of the origin for convenience. Each of the test functions has the unique minimum $g(0) = 0$.

G1: $g(x) = 2|x|$.

G2: $g(x) = [1 - cos(3\pi x)]/6 + 2|x|$.

G3: $g(x) = 0.5x^2$.

G4: $g(x) = \exp^{|x|-3} - \exp^{-3}$.

G5: $g(x) = 0.1[|x| + x^2 - 1/(x^2 + 0.2)] + 5$.

G6: $g(x) = 20x^2/(x^2 + 1)$.

G7: $g(x) = 10[1.3^x - 1] - 2.6x$.

G8: $g(x) = 10[1.3^{-x} - 1] + 2.6x$.

G9: $g(x) = [exp^{1.2x} + 2(x - 0.3)^2 - 1.18]/25$.

G10: $g(x) = [exp^{-1.2x} + 2(x + 0.3)^2 - 1.18]/25$.

G11: $g(x) = \begin{cases} 0.4x^2 & x < 0 \\ 10x^2/(x^2 + 1) & x \geq 0. \end{cases}$

G12: $g(x) = \begin{cases} 10x^2/(x^2 + 1) & x > 0 \\ 0.4x^2 & x \leq 0. \end{cases}$

These functions are illustrated using the same axis scales in Figures 5.1 - 5.3.

5.2.3 Measure of Performance

Several measures of performance involving distance to the optimum, observed function values and expected function values were considered. Barton (1987) argues using the percentage gap remaining in the expected function value after iteration k, where $GAP_k = g(x_k) - g(x^*)$ and x^* is the minimum. We define x_k to be the current point after the k^{th} iteration of the Kiefer-Wolfowitz method, or similarly

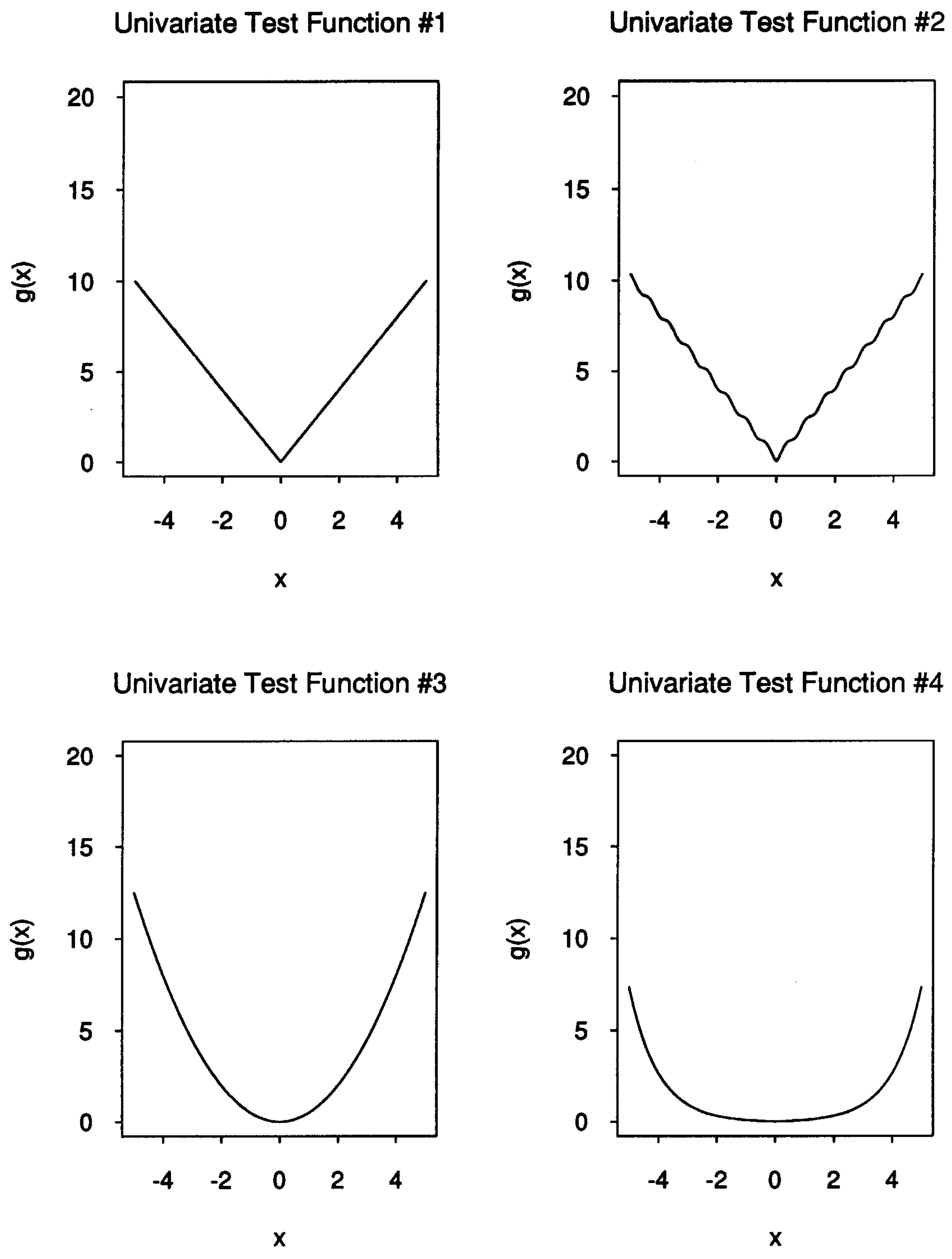


Figure 5.1: Univariate Test Functions #1-4

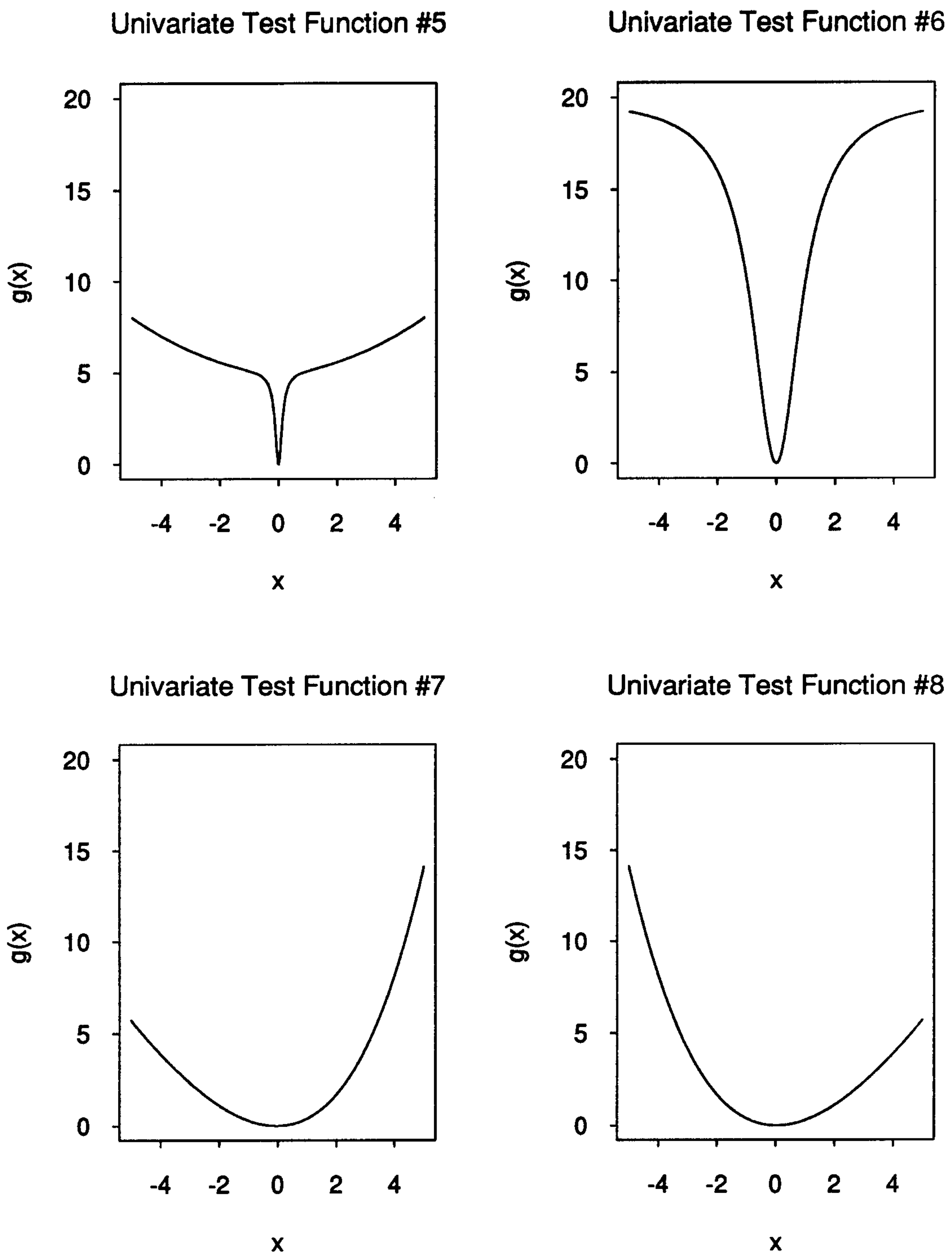


Figure 5.2: Univariate Test Functions #5-8

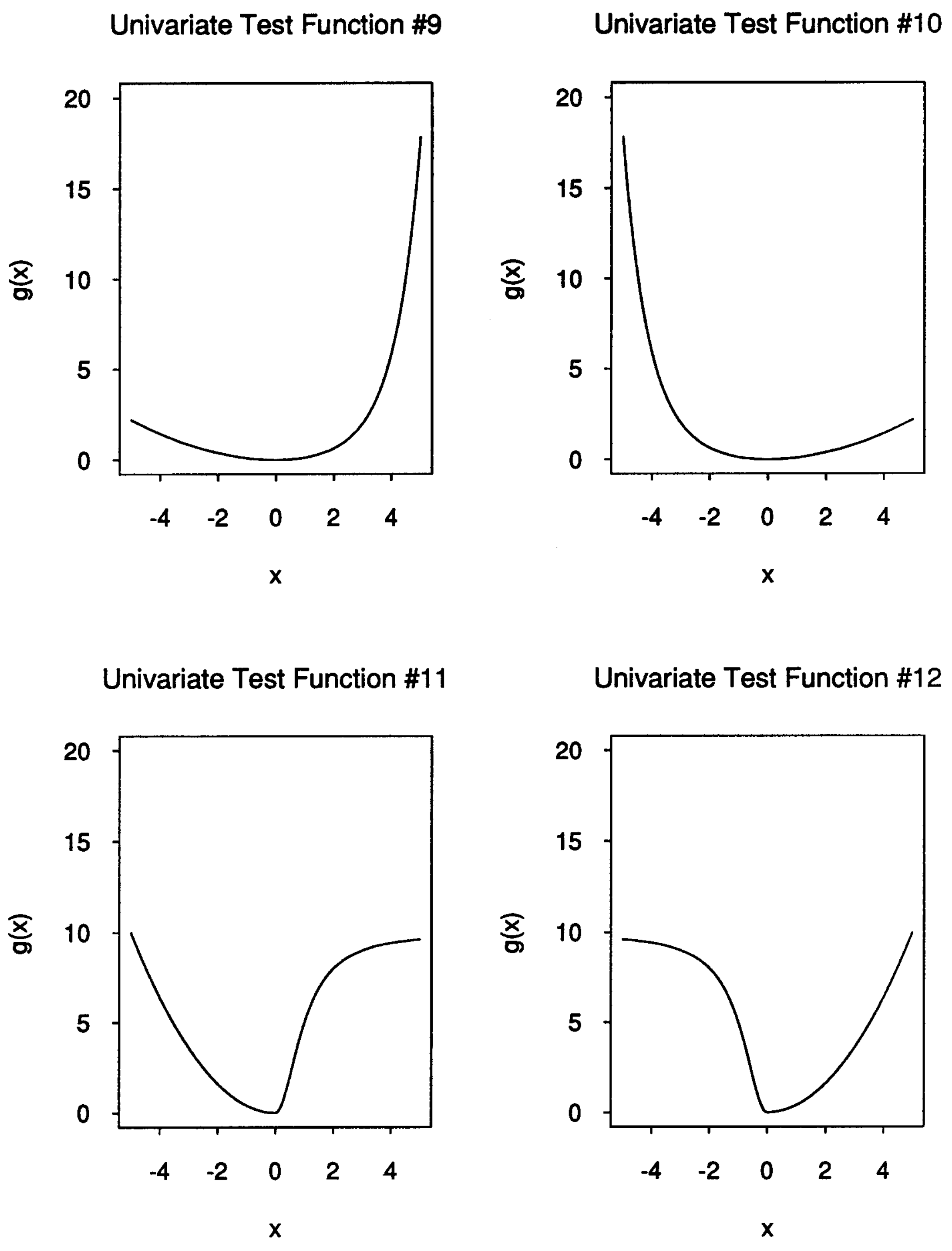


Figure 5.3: Univariate Test Functions #9-12

the center of mass of the simplex after the k^{th} iteration of a simplex method. The measure of performance is given in Equation 5.4 below.

$$PERGAP_k = 100\left(\frac{g(x_k)-g(x^*)}{g(x_0)-g(x^*)}\right), \tag{5.4}$$

where x_0 is the starting point in the Kiefer-Wolfowitz method or the center of mass of the starting simplex in the simplex methods. All of the algorithms tested are compared solely using the number of objective function evaluations required to achieve some level of $PERGAP$. Such a comparison assumes that all costs except the cost of objective function evaluations are insignificant as in the case where each evaluation may involve a lengthy computer simulation or a costly field experiment.

5.2.4 Starting Values

Hillstrom (1977) recommends using multiple starting values to reduce the likelihood of biased results and Barton (1987) further adds that random starting values allows for probabilistic assertions about the relative performance of the different algorithms. Combining these two ideas we decided upon three ranges of uniformly distributed starting values centered around an initial GAP/σ ratio. A large initial GAP/σ value should make it a relatively easy task for all of the methods to reduce $PERGAP$ significantly; whereas, reducing $PERGAP$ becomes more difficult as the initial GAP/σ value is decreased. What exactly a "small" value versus a "large" value would be is open to debate. However, we chose three different orders of magnitude for our initial ranges each of which occur to the right of the origin: (1) 1.0 ± 0.1; (2) 10.0 ± 0.1; and (3) 100.0 ± 0.1. By perturbing the starting values we reduce the likelihood of problems associated with starting at a particular value that might bias a single method because of the geometry of its step sizes.

Upon specification of an initial GAP/σ value, the corresponding value of x is determined using a simple search procedure employing the bisection method. The scale of the Gaussian errors was chosen to be 1 except in the following scenarios: (1) $\sigma = 0.19$ for the 6th function when $GAP/\sigma = 100$ as $\sup_x f_6(x) = 20.0$; and (2) for the 11th function $\sigma = 1.0, 0.2, 0.09$ for the three starting ranges of GAP/σ respectively as $\sup_{x>0} f_{11}(x) = 10$.

5.2.5 Experimental Design

A full factorial design of all combinations of the algorithms, test functions and ranges of starting values was performed. Therefore, our experiment consisted of 4 algorithms $\times$ 12 functions $\times$ 3 ranges of starting values for a total of 144 design points. Each of these design points were replicated 40 times using common random numbers from 40 separate streams (1 stream for each replication). The variables containing the cumulative number of objective function evaluations and the level of $PERGAP$ were recorded after each iteration and averaged across the 40 replications. A single replication was halted when one of the following stopping criteria was satisfied: (1) the number of iterations reached 10,000; (2) the number of function evaluations exceeded 50,000; or (3) the simplex size fell below 10^{-10}.

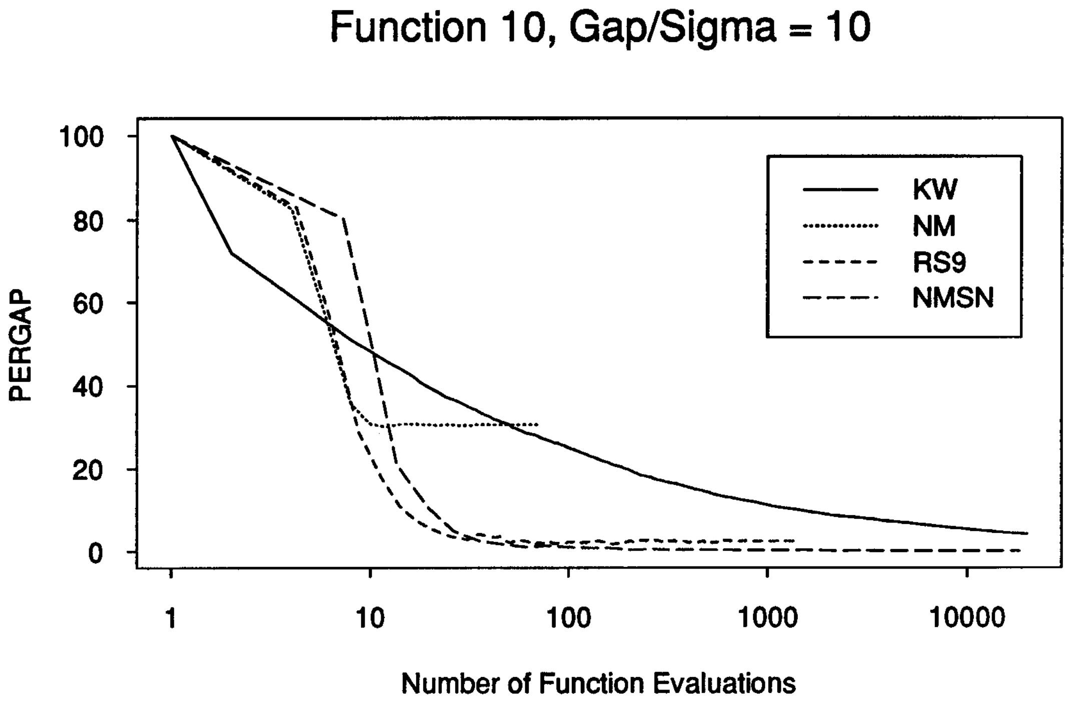
5.3 Results

The experiment generated a large volume of data and we quickly discovered that the most effective means of examining the data was through the use of graphs. For every combination of test function and starting value we plotted the average level of $PERGAP$ versus the average number of objective function evaluations for the four algorithms tested. These averages are by iteration number; and therefore, slightly different results would have been obtained if we had selected another means for averaging across the 40 replications (*e.g.* averaging *PERGAP* for every 10 function evaluations). We plotted these averages at each iteration until the average number of function evaluations reached 20,000. This is the point at which algorithm KW was terminated having reached the stopping criterion of 10,000 iterations at a cost of two function evaluations per iteration. Algorithm NMSN reached 20,000 function evaluations in fewer iterations than KW and was terminated upon exceeding 50,000 function evaluations. Methods NM and RS9 required fewer function evaluations per iteration than NMSN, and always terminated as a result of the simplex size falling below 10^{-10} before reaching 10,000 iterations; and therefore, their lines end well short of the range of the graphs.

Additionally, we plotted each function/starting value combination twice. A linear scale for the variable *PERGAP* is more useful for practical considerations; whereas, a logarithmic scale is more useful for theoretical considerations. Because of the large number of graphs, we have included only a sample pair of graphs here, Figure 5.4, along with our discussion of the results. However, the complete set of graphs are offered in Appendix A for further reference and average values at termination are tabulated for several variables in Appendix B.

5.3.1 Discussion of Algorithmic Performance

On most of our test functions the Kiefer-Wolfowitz algorithm performed well when started near the minimum (*i.e.* $GAP/\sigma = 1.0 \pm 0.1$). As the range of the starting point was moved farther from the minimum, the slow convergence rate of this algorithm is clearly exhibited. In addition to the decreasing gain parameter, the



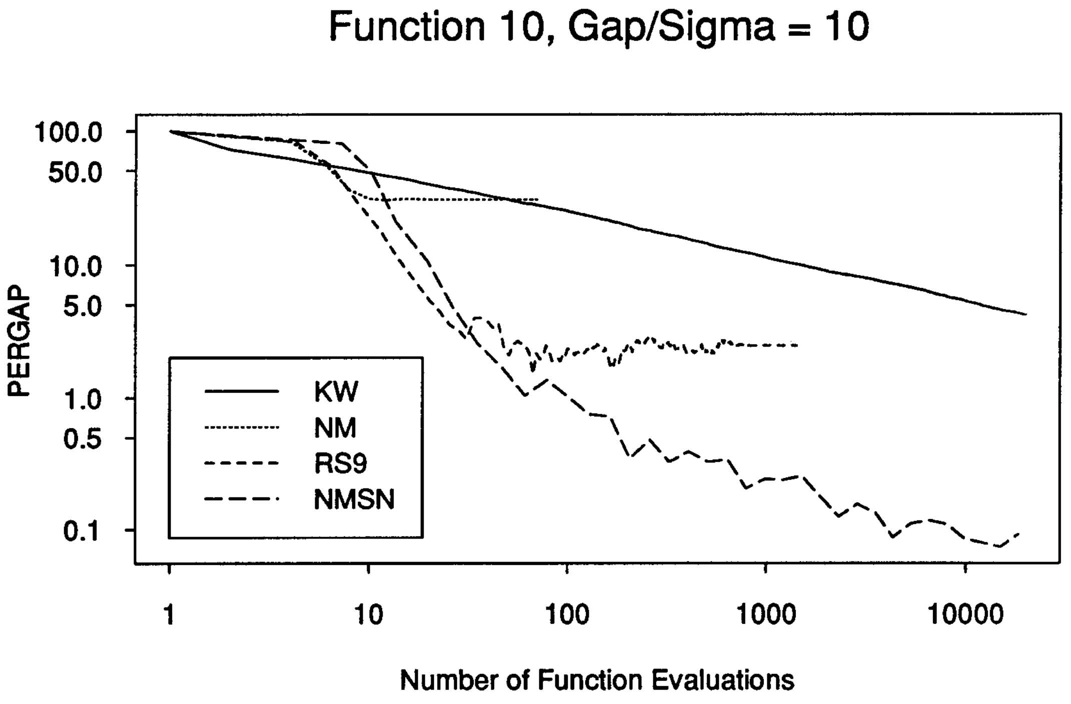


Figure 5.4: Sample Univariate Test Results

algorithm's step size is influenced by the magnitude of the gradient of the expected function at the current point. This influence is especially great at the start of the optimization while the gain parameter is fairly large. If the magnitude of the gradient is large, then the algorithm's next step size will be large. And, if the magnitude of the gradient at the current point is small, then the algorithm's next step size will be small. Hence, if the current point is a long distance from the minimum, and the magnitude of the gradient is small over that distance, then the rate of convergence will be very slow. On the other hand, if the minimum of the function lies inside a steep valley, then it is highly likely that the Kiefer-Wolfowitz method will overshoot the minimum by a large amount when it is started near or inside the valley. This phenomenon of overshooting the minimum is exhibited on Function 6 which caused a slow rate of convergence. This problem of overshooting the minimum early in the optimization was so severe on Function 4 that the $PERGAP$ computation resulted in a numerical overflow condition on the computer.

We are well aware of the fact that our implementation of the Kiefer-Wolfowitz algorithm is a straight forward interpretation of their original paper (1952), and for that reason it may be considered a naive implementation. Others have examined ways of improving stochastic approximation methods, *e.g.* Kesten (1958); however, we were not able to locate any implementations of stochastic approximation algorithms in the public domain (Andradottir, 1994; Ruppert, 1994). Besides, our intention of including the Kiefer-Wolfowitz algorithm was to have a method that is known to converge to the true optimum with which to compare our NMSN simplex algorithm.

The false convergence of algorithm NM is clearly exhibited in most of the graphs even when the algorithm is started far from the minimum as when $GAP/\sigma =$

100 ± 0.1. Obviously, when algorithm NM does terminate, additional improvement may not be of practical importance. But, clearly the Nelder-Mead simplex algorithm is not the best algorithm to choose for optimization under noisy conditions. Many modifications have been offered to improve the performance of this simplex algorithm on stochastic functions and several were tested by Barton and Ivey (1993). The best performing modifications from their study were incorporated in algorithm RS9, but it also exhibits the behavior of false convergence even though its onset is delayed. Particularly on a logarithmic scale, the convergence of these two algorithms before reaching the minimum can be clearly seen.

Our proposed algorithm, NMSN, does not appear to exhibit this problem of false convergence. This is clearly seen in the graphs using a logarithmic scale for $PERGAP$. In fact, in all but two of the graphs algorithm NMSN is doing as well or even better at minimizing the function over the first 20,000 function evaluations. The two minor exceptions occur on Function 11 at $GAP/\sigma = 10$ and Function 6 at $GAP/\sigma = 100$. Furthermore, algorithm NMSN never exhibited the kind of severely slow convergence rate as exhibited by algorithm KW on some functions, particularly when started far from the minimum.

5.3.2 Sensitivity of NMSN Parameters

Additional simulations were performed with algorithm NMSN to test the sensitivity of the critical value and the multiplication factor for increasing the sample size of observations. First, while maintaining a constant multiplier of $b = 1.25$ we set the critical value to correspond to the following values of α: 0.5, 0.4, 0.3, 0.2, and 0.1 in addition to the runs made for $\alpha = 0.05$. These values were tested on all 12 functions and at starting ranges $GAP/\sigma = 1.0 \pm 0.1$ and 10.0 ± 0.1.

Graphs of $PERGAP$ versus number of function evaluations did not indicate a large sensitivity to changes in the critical value; however, tabular summaries indicate that larger critical values are more efficient for obtaining lower levels of $PERGAP$. In Tables 5.1 and 5.2 we report the average number of function evaluations required to reach the point where $PERGAP$ remains below the specified level.

Next, we held the critical value corresponding to $\alpha = 0.10$ constant and set the multiplication factor b to the following values: 1.10, 1.25, 1.50, and 2.00. Once again, the graphs of the results were not very conclusive. In general they indicate that higher values are more expensive early in the optimization, but $b = 1.10$ was generally more expensive near the end of the optimization. This supports our rationale for choosing b to be only slightly larger than δ^{-2}. If b is not at least as large as the square of the inverse of the shrinkage coefficient, then false convergence is certain to occur. However, larger values of b seem to be of no benefit at best. Table 5.3 reports the number of function evaluations required to reach the point where $PERGAP$ remains below 1% on the 12 test functions started at $GAP/\sigma = 10.0 \pm 0.1$.

5.4 Conclusion

Our test results demonstrate the superiority of algorithm NMSN over the original Nelder-Mead algorithm and the previously known best performing derivative RS9. Although algorithm RS9 performed much better than the original Nelder-Mead algorithm, it too exhibited the problem of false convergence although its onset was delayed. It appears that progress towards the true minimum cannot be sustained without increasing the sample size of observations taken at each vertex. All of the

Table 5.1: Effect of Critical Value on Average Number of Function Evaluations Needed by NMSN to Reduce $PERGAP$ to 10% when starting at $GAP/\sigma = 1.0 \pm 0.1$

Test function	Level of Significance (α)					
	0.50	0.40	0.30	0.20	0.10	0.05
1	3373	3122	2076	1578	545	618
2	3566	2005	1037	611	440	420
3	2101	287	89	72	49	14
4	3118	153	76	72	47	69
5	705	1017	901	562	1862	1372
6	681	4	4	4	4	4
7	615	307	66	97	132	40
8	602	144	51	251	40	29
9	724	268	35	52	32	54
10	321	377	332	257	78	121
11	1166	796	121	63	105	86
12	565	327	57	118	121	77
Average	1461	733	403	311	287	242

Table 5.2: Effect of Critical Value on Average Number of Function Evaluations Needed by NMSN to Reduce *PERGAP* to 2% when starting at $GAP/\sigma = 10.0 \pm 0.1$

Test function	Level of Significance (α)					
	0.50	0.40	0.30	0.20	0.10	0.05
1	2382	664	800	420	303	188
2	558	642	554	499	607	507
3	9	9	9	9	10	11
4	9	10	10	10	11	12
5	287	558	514	652	1287	1093
6	427	23	25	29	31	34
7	9	9	9	10	9	11
8	14	20	15	17	18	18
9	9	10	11	11	12	12
10	35	30	39	36	41	46
11	28	22	31	26	27	30
12	57	79	79	76	131	123
Average	318	173	174	149	207	173

Table 5.3: Effect of Multiplication Factor on Average Number of Function Evaluations Needed by NMSN to Reduce *PERGAP* to 1% when starting at $GAP/\sigma = 10.0 \pm 0.1$

Test function	Multiplication Factor (b)			
	1.10	1.25	1.50	2.00
1	2471	815	1803	3786
2	1156	837	1781	1082
3	10	18	18	19
4	45	11	50	22
5	771	1158	2409	7125
6	33	42	53	70
7	10	73	10	10
8	38	37	40	28
9	12	11	21	12
10	101	87	84	126
11	50	82	78	203
12	133	159	281	708
Average	402	277	552	1099

variants tested by Barton and Ivey (1993) involved modifications similar to those in RS9 to delay the onset of false convergence, but none of them attack the problem by averaging repeated observations at each vertex.

Given our set of test conditions, the algorithm NMSN performed the best overall in terms of reducing the expected function value. The algorithm reduces to the original Nelder-Mead simplex method when noise is not present or when it is negligible. And, under noisy conditions the algorithm sensibly increases the sample size of observations taken at each new vertex in order to obtain more precise estimates to guarantee further progress towards the minimum. When the Kiefer-Wolfowitz algorithm performed well, algorithm NMSN performed just as well. And, method NMSN appears to be more robust than the Kiefer-Wolfowitz stochastic approximation method, which demonstrated a very slow rate of convergence on some functions.

Although one could probably create a situation in which method NMSN performed much worse than any of the other methods we tested, we believe that our set of test functions represents the types of topologies one would expect to encounter in the class of continuous functions with a unique minimum. And on these test functions, algorithm NMSN shows the best overall performance among the four algorithms tested.

Chapter 6

Extension of Proposed Algorithm to N Dimensions

Given the success of our proposed algorithm for univariate stochastic minimization we next considered extending the procedure for determining when to increase the sample size per vertex to n-dimensional problems. In this chapter we present some candidate procedures for such an extension of our proposed algorithm. When $n = 1$ the procedures in Sections 6.1 and 6.2 reduce to the univariate procedure discussed in Section 4.4.

The model for observations of the function g at $\mathbf{x}_j$ remains the same; that is, assume Y_{ij} independent with

$$Y_{ij} \sim N(\mu_j, \sigma^2), \tag{6.1}$$

where $\mu_j = g(\mathbf{x}_j)$ for some continuous real $g : \Re^n \longrightarrow \Re$ with a unique minimum. Given m_j iid observations of g at $\mathbf{x}_j$, we are able to determine critical values for the hypothesis test under the procedures described in Sections 6.1 and 6.2. Each of the procedures test the following set of hypotheses:

$$\begin{aligned} H_0 &: \; \mu_1 = \mu_2 = \cdots = \mu_{n+1} \\ H_1 &: \; \text{not all } \mu_j \text{ are equal} \end{aligned} \tag{6.2}$$

As in our univariate procedure, we increase the minimum sample size needed per vertex if we fail to reject the null hypothesis; otherwise, we may decrease the minimum sample size required.

6.1 Range of Vertex Means

Noting that our univariate procedure of Section 4.4 uses the difference between the two vertex means, one natural extension of this procedure to higher dimensions would be to use the range of the vertex means; that is, the difference between the highest and lowest vertex means. One restriction we need to make for theoretical reasons is the requirement that the sample size at all the vertices are equal (*i.e.* $m_j = m$). Then assuming an iid sample of size m from our model in Equation 6.1 we have

$$\overline{Y}_{.j} - \mu_j \sim N\left(0, \frac{\sigma^2}{m}\right). \tag{6.3}$$

Assuming the Y_{ij} are independent, the $(\overline{Y}_{.j} - \mu_j)$ are independent and identically distributed. Therefore, a test could be based on the standardized range distribution. The statistic T_3 is distributed as the standardized range under the null hypothesis, where

$$T_3 = \frac{\overline{Y}_{.[n+1]} - \overline{Y}_{.[1]}}{\sigma/\sqrt{m}} \tag{6.4}$$

and where $\overline{Y}_{.[n+1]} = \max(\overline{Y}_{.j})$ and $\overline{Y}_{.[1]} = \min(\overline{Y}_{.j})$. Harter (1960) tabulated critical values for the standardized range distribution. In the event σ^2 is unknown, one may use the studentized range distribution which uses the mean squared error (MSE) to estimate σ^2. The sum of squares for error is defined in Equation 6.5.

$$\begin{aligned} SSE &= \sum_{j=1}^{n+1}\sum_{i=1}^{m_j}\left(Y_{ij} - \overline{Y}_{.j}\right)^2 \\ &= \sum_{j=1}^{n+1}\sum_{i=1}^{m_j} Y_{ij}^2 - \sum_{j=1}^{n+1} m_j \overline{Y}_{.j}^2 . \end{aligned} \tag{6.5}$$

Then $MSE = SSE/(M - (n+1))$, where $M = \sum_{j=1}^{n+1} m_j$, is an unbiased estimator for σ^2. It follows from the definition of the studentized range distribution q (see pages 180-182 Arnold, 1981) that

$$\frac{\max(\overline{Y}_{.j} - \mu_j) - \min(\overline{Y}_{.j} - \mu_j)}{\sqrt{MSE/m}} \sim q_{n+1,M-(n+1)}\,, \tag{6.6}$$

where $n + 1$ is the number of treatments (vertices for our usage) and $M - (n + 1)$ is the number of degrees of freedom associated with SSE. Hence the statistic T_4 of Equation 6.7 has a studentized range distribution under the null hypothesis.

$$T_4 = \frac{\overline{Y}_{.[n+1]} - \overline{Y}_{.[1]}}{\sqrt{MSE/m}}. \tag{6.7}$$

The appropriate test rejects the null hypothesis whenever $T_4 > q^{\alpha}_{n+1,M-(n+1)}$, where the critical value is the $100(1 - \alpha)$ percentile of the studentized range distribution for $n + 1$ treatments and $M - (n + 1)$ degrees of freedom for SSE. Tables of these critical values can be found in Harter (1960) as well as most textbooks on applied linear models (*e.g.* Neter, Wasserman, & Kutner, 1985).

6.2 Variance of Vertex Means

Suppose we define the signal to be the variance of the μ_j, which are the expected function values at the current simplex vertices. We can estimate this variance using the numerator mean squares from the one-way analysis of variance (ANOVA) model. The numerator sum of squares is given in Equation 6.8.

$$\begin{aligned} S^2 &= \sum_{j=1}^{n+1} m_j \left(\overline{Y}_{.j} - \overline{Y}_{..}\right)^2 \\ &= \sum_{j=1}^{n+1} m_j \overline{Y}^2_{.j} - M\overline{Y}^2_{..}\,. \end{aligned} \tag{6.8}$$

where $M = \sum_{j=1}^{n+1} m_j$. Next, we define the test statistic T_5.

$$T_5 = \frac{S^2/n}{\sigma^2}\,. \tag{6.9}$$

Under the null hypothesis $T_5 \sim \chi^2_n$, a chi-squared distribution on n degrees of freedom. And, we reject the null hypothesis if $T_5 > \chi^{2\alpha}_n$. Note the expected value of the mean squares (see p. 447 Arnold, 1990) given in Equation 6.10.

$$\frac{1}{n}ES^2 = \sigma^2 + \frac{1}{n}\sum_{j=1}^{n+1} m_j(\mu_j - \overline{\mu})^2, \tag{6.10}$$

where $\overline{\mu} = (1/M)\sum_{j=1}^{n+1} m_j\mu_j$. Therefore, our statistic is an estimate of the ratio of (signal plus noise) to noise. And, if the μ_j are all equal, then $ES^2/n = \sigma^2$. Furthermore, when $n = 1$ this test is the same test we derived in Section 4.4.

Since Equation 6.8 is the formula for the numerator sum of squares in a one-way ANOVA test, the natural extension to the situation where σ^2 is unknown is the familiar F-test used in ANOVA. The denominator sum of squares for the test statistic is given by Equation 6.5, and the appropriate test statistic is given in Equation 6.11.

$$T_6 = \frac{S^2/n}{SSE/(M-(n+1))}. \tag{6.11}$$

Therefore, under the null hypothesis $T_6 \sim F_{n,M-(n+1)}$, an F-distribution on n and $M-(n+1)$ degrees of freedom; and, we reject the null hypothesis if $T_6 > F^{\alpha}_{n,M-(n+1)}$.

6.3 Other Possible Candidate Procedures

Other procedures for testing the hypothesis that all vertex means are equal may be developed. In this section we discuss the use of order statistics for the testing procedure. First, we present how the critical value of the largest order statistic from an iid sample of $N(0,1)$ random variables can be found. Let $Z_1, \ldots, Z_n$ be independently distributed $N(0,1)$ random variables and let $Z_{[n]} = \max\{Z_1, \ldots, Z_n\}$. Then probability statements involving $Z_{[n]}$ are easily made as in Equation 6.12.

$$P(Z_{[n]} > z) = 1 - \Phi(z)^n, \tag{6.12}$$

Table 6.1: Upper $\alpha = 0.05$ Percentage Points of the Distribution of $Z_{[n]}$

n	$\Phi(z)$	z	n	$\Phi(z)$	z
1	0.9500	1.645	6	0.9915	2.386
2	0.9747	1.955	7	0.9927	2.442
3	0.9830	2.121	8	0.9936	2.490
4	0.9873	2.234	9	0.9943	2.531
5	0.9898	2.319	10	0.9949	2.568

where Φ is the cumulative distribution function for the standard normal distribution. By setting the right side of Equation 6.12 equal to α, a critical value for a size α test based on $Z_{[n]}$ can be found by first solving for $\Phi(z)$ (see Equation 6.13).

$$\Phi(z) = \sqrt[n]{1-\alpha}. \tag{6.13}$$

Then the appropriate critical value can be located using a table or computer program that yields percentiles of the standard normal distribution. For example, Table 6.1 lists the upper $\alpha = 0.05$ percentage points for the distribution of $Z_{[n]}$ and $n = 1, \ldots, 10$.

One criticism of using the range of the vertex means (Section 6.1) is that the search direction for $n > 1$ is not from the worst vertex through the best vertex but rather from the worst vertex through the centroid of the remaining vertices. This suggests a procedure similar to the one in Section 6.1, but instead we use the mean at the worst vertex (the largest mean) minus the average of the remaining vertex means (which is an estimate of the mean at the centroid). Let $\overline{Y}^{(i)}$ be the weighted average of the vertex sample means excluding vertex i as in Equation 6.14.

$$\overline{Y}^{(i)} = \frac{1}{M^{(i)}} \sum_{j \neq i} m_j \overline{Y}_{.j}, \tag{6.14}$$

where $M^{(i)} = \sum_{j \neq i} m_j$. And let $\bar{\mu}^{(i)}$ be the weighted average of the true vertex means. To obtain identically distributed random variables we must have equal sample sizes at the vertices (*i.e.* let $m_j = m \ \ \forall j$); and therefore, $M^{(i)} = nm$. Now, let W_j be defined by Equation 6.15.

$$W_j = \sqrt{\frac{nm}{n+1}} \left(\frac{(\overline{Y}_{.j} - \mu_j) - (\overline{Y}^{(j)} - \bar{\mu}^{(j)})}{\sigma} \right). \tag{6.15}$$

Although the W_j are identically distributed $N(0,1)$ random variables, they are not independent. To achieve independence we would have to resample all vertices for each W_j. Therefore, this procedure would be impractical to implement.

With a slight modification, however, we can develop another procedure that does not require resampling the vertices $(n+1)$ times to achieve independence for the $(n+1)$ random variables. Instead, suppose we partition the total number of observations at a vertex into two groups, m and m'. Let $\overline{Y}_{.j}$ be the sample mean of the first m observations at vertex j. Next, let $\overline{Y}$ be the grand mean of the remaining $m'(n+1)$ observations (m' per vertex) where we could let $m' \approx m/(n+1)$. Therefore, $\overline{Y}_{.1}, \ldots, \overline{Y}_{.(n+1)}, \overline{Y}$ are independent estimators for $\mu_1, \ldots, \mu_{n+1}, \bar{\mu}$ respectively. Now, define U_j as follows:

$$U_j = \sqrt{\frac{mm'(n+1)}{m + m'(n+1)}} \left(\frac{(\overline{Y}_{.j} - \mu_j) - (\overline{Y} - \bar{\mu})}{\sigma} \right). \tag{6.16}$$

Then $U_1, \ldots, U_{n+1}$ are independent $N(0,1)$ random variables. Next, define the test statistic T_7 as follows:

$$T_7 = \max_j \left\{ \sqrt{\frac{mm'(n+1)}{m + m'(n+1)}} \left(\frac{\overline{Y}_{.j} - \overline{Y}}{\sigma} \right) \right\}. \tag{6.17}$$

Then under the null hypothesis T_7 has the same distribution as $Z_{[n+1]}$, and the critical values in Table 6.1 may be used for an $\alpha = 0.05$ test of equal vertex means.

6.4 Summary

We have described several procedures for determining whether to increase or decrease the minimum sample size of observations taken at each vertex. In Chapter 7 we implement the standardized range procedure and the variance of the means procedure in a multivariate experiment. We chose not to estimate σ^2, because implementation of the studentized range or ANOVA procedures would require a large table of critical values or the coding of an algorithm to derive them as needed. Also, our implementations do not include resampling the retained vertices at every iteration except following a shrinkage. Therefore, as was the case for our univariate procedure, our implementations are heuristic rules for changing the vertex sample size. Since the hypothesis test will not have the correct size, except following a shrinkage, we surmise this will cause the test to fail more often; and therefore, the sample size may be increased unnecessarily. However, additional observations averaged with those from previous iterations will reduce the dependency between the vertex means at the next iteration. For this reason and because of the impracticality of discarding observations from previous iterations, our implementations do not include resampling after every iteration.

Chapter 7

Testing of Proposed Algorithm on Multivariate Functions

7.1 Purpose

We ran a similar experiment to the one described in Chapter 5 to investigate the performance of our proposed algorithm for minimizing a stochastic function of more than one variable. Our model of the minimization problem is similar to the one described in Chapter 5 except that the domain of the function g is now $\Re^n$.

7.2 The Design Frame

In this section we define the various algorithms we tested, the set of test functions for the real continuous function g, the measure of performance and the starting values for the algorithms.

7.2.1 Optimization Codes

Algorithms NM, RS9 and KW were included in our multivariate experiment for comparison with our proposed algorithm. In addition to these three algorithms the following two algorithms were implemented in FORTRAN and run on a Sun SPARCstation.

NMSNR: Our proposed algorithm using the range of the vertex means to estimate the signal assuming σ is known. Once again we use a shrinkage coefficient of 0.9 and a contraction coefficient of 0.9. Let $m^k = min(m_1, \ldots, m_{n+1})$ following the k^{th} iteration of the algorithm. The rule for determining the sample size at the next iteration is

$$m^{k+1} = \begin{cases} \lfloor b m^k \rfloor & \text{if } \dfrac{\overline{Y}_{\cdot[n+1]} - \overline{Y}_{\cdot[1]}}{\sigma/\sqrt{m^k}} \leq r^{\alpha}_{n+1} \\ \lfloor m^k/b \rfloor & \text{if } \dfrac{\overline{Y}_{\cdot[n+1]} - \overline{Y}_{\cdot[1]}}{\sigma/\sqrt{m^k}} > r^{\alpha}_{n+1} \end{cases} \qquad (7.1)$$

where r^{α}_{n+1} is the $100(1-\alpha)$ percentile of the standardized range distribution with $n+1$ levels (simplex vertices). Theory requires all vertex sample sizes to be equal; however, we allow different sample sizes in our implementation since we do not have independent means anyway except following a shrinkage at which time the sample means are all equal, too.

NMSNV: Our proposed algorithm using the numerator mean squares due to differences in the expected function values at the vertices to estimate the signal assuming σ^2 is known. The shrinkage and contraction coefficients are set to 0.9 as in NMSNR and the rule for changing the minimum sample size of observations to take at new vertices is given in Equation 7.2.

$$m^{k+1} = \begin{cases} \lfloor b m^k \rfloor & \text{if } \dfrac{S^2/n}{\sigma^2} \leq \chi^{2\,\alpha}_{n} \\ \lfloor m^k/b \rfloor & \text{if } \dfrac{S^2/n}{\sigma^2} > \chi^{2\,\alpha}_{n} \end{cases} \qquad (7.2)$$

where $\chi^{2\,\alpha}_{n}$ is the $100(1-\alpha)$ percentile of the chi-squared distribution with n degrees of freedom and S^2 is the numerator sum of squares defined by Equation 6.8.

For the primary experiment, the methods NMSNR and NMSNV were normalized in the selection of critical values for their respective testing procedures using $\alpha = 0.05$. Later simulation runs were made to test the sensitivity of these two methods to changes in the critical values (*i.e.* changes in α).

7.2.2 Test Functions

The set of test functions used in this experiment are the same ones selected by Barton and Ivey (1993). They are a set of 18 deterministic functions collected by More, Garbow, and Hillstrom (1981) for the express purpose of testing the reliability and robustness of unconstrained optimization software. Each of the test functions in the collection have the form $f : \Re^n \longrightarrow \Re^m$ where n, m or both may be variable. An unconstrained minimization problem is formed using a sum of squares as in Equation 7.3.

$$g(\mathbf{x}) = \sum_{i=1}^{m} f_i^2(\mathbf{x}), \quad \mathbf{x} \in \Re^n. \tag{7.3}$$

For large values of m or badly scaled functions f_i, the value of g will greatly exceed its minimum value even within a relatively small neighborhood of the optimum. Therefore, we divide g of Equation 7.3 by 10,000 (except for Function 13) to rescale it before adding the $N(0, 1)$ noise. The parameters n and m along with the names by which the functions are known are listed in Table 7.1.

7.2.3 Measure of Performance

Our measure of performance is the same one used in the univariate testing described in Chapter 5. That is, it is the percentage gap remaining in the expected function values, where the initial gap is the difference in the expected function values at the starting point and at the minimum. Our performance variable *PERGAP* is defined by Equation 5.4.

Table 7.1: Parameters of Multivariate Test Functions

#	Function Name	n	m	#	Function Name	n	m
1	Helical Valley	3	3	10	Brown badly scaled	2	3
2	Biggs EXP6	6	13	11	Brown & Dennis	4	20
3	Gaussian	3	15	12	Gulf R&D	3	99
4	Powell badly scaled	2	2	13	Trigonometric	8	8
5	Box 3-Dimensional	3	10	14	Extended Rosenbrock	4	4
6	Variably Dimensioned	4	6	15	Extended Powell	8	8
7	Watson	9	31	16	Beale	2	3
8	Penalty Function I	8	9	17	Wood	4	6
9	Penalty Function II	8	16	18	Chebyquad	9	9

7.2.4 Starting Values

More et al. (1981) provide starting values for each of the functions in their collection. These starting values were used by Barton and Ivey (1993) in their empirical tests. However, upon rescaling, some of these starting values are too close to the minimum. Rather, we selected starting values to control the initial gap to obtain a starting GAP/σ ratio as was done in the univariate testing. Since the results in the univariate experiment for $GAP/\sigma = 100$ proved to be not very interesting in differentiating the simplex-based algorithms we chose starting values near $GAP/\sigma = 1$ and 10 only. Once having located a particular starting value, each coordinate was perturbed by adding a uniformly distributed variate on the interval $(-0.1, 0.1)$. The starting values are listed in Table 7.2.

Table 7.2: Starting Values of Multivariate Test Functions

Test function	Starting Values	
	$GAP/\sigma \approx 1$	$GAP/\sigma \approx 10$
1	(3, 5, -7.2)	(5, 25, -17.74)
2	(10, -2, 8, -1, -2.7, -1.5)	(10, -2, 20, -4.9, -1.5, 4.9)
3	(2, -0.1, -5)	(6.28, -0.1, -5)
4	(0.01, 1)	(0.01, 3.2)
5	(-4.25, 3, -10)	(-5.5, 4, -20)
6	$x_j = (j/n - 0.1)(-1)^{j+1}$	$x_j = (4 - j/n)(-1)^{j+1}$
7	$x_j = -0.65$	$x_j = -1.32$
8	$x_j = 0.7j$	$x_j = 1.25j$
9	$x_j = 1.7$	$x_j = 3$
10	(1.0E+06, 1.05E-04)	(9.999E+05, 5.0E-06)
11	(-8.6, 12.2, -0.7, 0.3)	(-8, 11, -5, 0)
12	(-0.95, 1, 0.333)	(-0.95, 1, 0.4)
13	$x_j = 0.45j/8$	$x_j = 0.71j/8$
14	$x_j = 2.2(-1)^{j+1}$	$x_j = 4.4(-1)^{j+1}$
15	(3, -3, 1.5, 7.1, 3, -3, 1.5, 7.1)	(3, -9, 1.5, 10, 3, -9, 1.5, 10)
16	(2.6, 4.3)	(2.5, 6)
17	(-2.8, -2, 3, 7)	(-5, -2, -5, 7)
18	$x_j = 0.1j + 0.274$	$x_j = 0.1j + 0.34$

7.2.5 Experimental Design

A full factorial design of the 5 algorithms $\times$ 18 test functions $\times$ 2 starting values yielding 180 design points was performed. Each design point was replicated 40 times using common random number from 40 separate streams. The variables containing the cumulative number of objective function evaluations and the level of $PERGAP$ were recorded after each iteration and averaged across the 40 replications. A single replication was halted when one of the following stopping criteria was satisfied: (1) the total number of function evaluations exceeded 10,000$(n+1)$, where n is the dimension of the domain of the test function; (2) the total number of iterations reached 10,000; or (3) the simplex size fell below 10^{-10}.

7.3 Results

Our multivariate experiment generated volumes of data as we discussed for the univariate testing in Section 5.3. The same types of graphs were made for the multivariate test functions and the complete set are offered in Appendix C. Additionally, the average value of several variables at termination are provided in tabular form in Appendix D. In Tables 7.3 - 7.8 we report the average value of $PERGAP$, when the average cost has reached 100, 1000, and 10,000 function evaluations for the 18 test functions and the 2 starting values. The lowest value of $PERGAP$ in each row of these tables is boxed providing a snapshot of the performance of the methods at these three cost figures; whereas, comparisons between the methods over the entire simulation are best made by examining the graphs in Appendix C.

Table 7.3: *PERGAP* at 100 Evaluations when starting at $GAP/\sigma \approx 1$

Test function	Method				
	NM	RS9	NMSNR	NMSNV	KW
1	0.996E+02	$\boxed{0.918E+02}$	0.941E+02	0.967E+02	0.928E+02
2	0.697E+02	0.376E+02	0.484E+02	0.472E+02	$\boxed{0.633E+01}$
3	0.147E+01	0.712E+00	0.140E+01	0.989E+00	$\boxed{0.127E+00}$
4	0.378E+02	$\boxed{0.791E+01}$	0.203E+02	0.191E+02	–
5	0.579E+02	0.208E+02	0.221E+02	0.223E+02	$\boxed{0.659E+01}$
6	0.184E+02	$\boxed{0.203E+01}$	0.252E+01	0.235E+01	–
7	0.646E+02	0.496E+02	0.697E+02	0.705E+02	$\boxed{0.893E+01}$
8	0.992E+02	0.992E+02	0.984E+02	0.978E+02	$\boxed{0.837E+02}$
9	0.927E+02	0.795E+02	0.880E+02	0.869E+02	$\boxed{0.328E+02}$
10	$\boxed{0.527E+02}$	$\boxed{0.527E+02}$	$\boxed{0.527E+02}$	$\boxed{0.527E+02}$	–
11	0.106E+03	$\boxed{0.811E+02}$	0.920E+02	0.919E+02	–
12	0.308E+00	$\boxed{0.117E+00}$	0.144E+00	0.175E+00	–
13	0.307E+02	0.442E+01	$\boxed{0.241E+01}$	0.297E+01	0.129E+06
14	0.814E+02	0.543E+02	0.576E+02	0.565E+02	$\boxed{0.246E+02}$
15	0.992E+02	0.917E+02	0.936E+02	0.941E+02	$\boxed{0.644E+02}$
16	0.524E+01	0.724E+00	0.646E+00	$\boxed{0.539E+00}$	0.591E+00
17	0.189E+01	$\boxed{0.145E+01}$	0.179E+01	0.182E+01	0.161E+01
18	0.128E+00	$\boxed{0.107E+00}$	0.129E+00	0.115E+00	–

Table 7.4: *PERGAP* at 1,000 Evaluations when starting at $GAP/\sigma \approx 1$

Test function	Method				
	NM	RS9	NMSNR	NMSNV	KW
1	0.996E+02	0.928E+02	$\boxed{0.804E+02}$	0.880E+02	0.850E+02
2	0.699E+02	0.334E+02	0.147E+02	0.150E+02	$\boxed{0.478E+01}$
3	0.147E+01	0.107E+01	0.124E+01	0.132E+01	$\boxed{0.127E+00}$
4	0.378E+02	0.111E+02	0.490E+01	$\boxed{0.460E+01}$	–
5	0.578E+02	0.194E+02	0.960E+01	0.914E+01	$\boxed{0.603E+01}$
6	0.184E+02	0.229E+01	0.264E+00	$\boxed{0.159E+00}$	–
7	0.652E+02	0.384E+02	0.298E+02	0.292E+02	$\boxed{0.492E+01}$
8	0.992E+02	0.987E+02	0.883E+02	0.876E+02	$\boxed{0.624E+02}$
9	0.926E+02	0.732E+02	0.512E+02	0.492E+02	$\boxed{0.196E+02}$
10	0.782E-03	$\boxed{0.411E-03}$	0.607E-02	0.457E-02	–
11	0.106E+03	0.808E+02	0.733E+02	$\boxed{0.673E+02}$	–
12	0.308E+00	$\boxed{0.984E-01}$	0.110E+00	0.131E+00	–
13	0.325E+02	0.123E+02	$\boxed{0.219E+01}$	0.255E+01	0.115E+06
14	0.814E+02	0.544E+02	0.198E+02	0.193E+02	$\boxed{0.157E+02}$
15	0.992E+02	0.894E+02	0.730E+02	0.736E+02	$\boxed{0.380E+02}$
16	0.524E+01	0.773E+00	0.203E+00	$\boxed{0.140E+00}$	0.439E+00
17	0.189E+01	0.147E+01	0.125E+01	$\boxed{0.107E+01}$	0.119E+01
18	0.503E+00	0.390E+00	$\boxed{0.121E+00}$	0.124E+00	–

Table 7.5: *PERGAP* at 10,000 Evaluations when starting at $GAP/\sigma \approx 1$

Test function	Method				
	NM	RS9	NMSNR	NMSNV	KW
1	0.996E+02	0.928E+02	$\boxed{0.407E+02}$	0.440E+02	0.772E+02
2	0.699E+02	0.334E+02	0.586E+01	0.529E+01	$\boxed{0.402E+01}$
3	0.147E+01	0.107E+01	0.281E+01	0.119E+01	$\boxed{0.127E+00}$
4	0.378E+02	0.111E+02	0.149E+01	$\boxed{0.110E+01}$	–
5	0.578E+02	0.194E+02	0.595E+01	0.568E+01	$\boxed{0.543E+01}$
6	0.184E+02	0.229E+01	$\boxed{0.494E-01}$	0.533E-01	–
7	0.652E+02	0.384E+02	0.295E+01	$\boxed{0.255E+01}$	0.411E+01
8	0.992E+02	0.987E+02	0.631E+02	0.615E+02	$\boxed{0.503E+02}$
9	0.926E+02	0.731E+02	0.131E+02	0.137E+02	$\boxed{0.128E+02}$
10	0.782E-03	0.412E-03	0.273E-02	$\boxed{0.153E-02}$	–
11	0.106E+03	0.808E+02	0.299E+02	$\boxed{0.238E+02}$	–
12	0.308E+00	$\boxed{0.984E-01}$	0.102E+00	0.114E+00	–
13	0.325E+02	0.123E+02	$\boxed{0.203E+01}$	0.232E+01	0.387E+01
14	0.814E+02	0.544E+02	0.705E+01	$\boxed{0.685E+01}$	0.118E+02
15	0.992E+02	0.894E+02	0.361E+02	0.382E+02	$\boxed{0.294E+02}$
16	0.524E+01	0.773E+00	0.123E+00	$\boxed{0.112E+00}$	0.290E+00
17	0.189E+01	0.147E+01	0.754E+00	$\boxed{0.665E+00}$	0.863E+00
18	0.503E+00	0.383E+00	$\boxed{0.807E-01}$	0.829E-01	–

Table 7.6: *PERGAP* at 100 Evaluations when starting at $GAP/\sigma \approx 10$

Test function	Method				
	NM	RS9	NMSNR	NMSNV	KW
1	0.985E+02	0.890E+02	0.925E+02	0.925E+02	$\boxed{0.882E+02}$
2	0.534E+01	0.338E+01	0.453E+01	0.606E+01	$\boxed{0.841E-01}$
3	0.963E+00	0.541E+00	0.104E+01	0.117E+01	$\boxed{0.123E-01}$
4	$\boxed{0.369E+00}$	0.727E+00	0.578E+01	0.651E+01	–
5	0.544E+01	0.277E+01	0.259E+01	0.239E+01	$\boxed{0.121E+01}$
6	0.460E+01	0.615E+00	0.774E+00	$\boxed{0.504E+00}$	–
7	0.220E+01	0.108E+01	0.286E+01	0.337E+02	$\boxed{0.877E+00}$
8	0.908E+02	0.776E+02	0.944E+02	0.944E+02	$\boxed{0.367E+02}$
9	0.232E+02	0.151E+02	0.683E+02	0.828E+02	$\boxed{0.341E+01}$
10	$\boxed{0.548E+02}$	$\boxed{0.548E+02}$	$\boxed{0.548E+02}$	$\boxed{0.548E+02}$	–
11	0.399E+02	$\boxed{0.213E+02}$	0.338E+02	0.400E+02	0.822E+87
12	0.132E+00	0.336E+04	0.434E-01	$\boxed{0.304E-01}$	–
13	0.147E+01	0.414E+00	$\boxed{0.210E+00}$	0.266E+00	0.152E+05
14	0.632E+01	$\boxed{0.338E+01}$	0.395E+01	0.421E+01	0.347E+01
15	0.559E+02	0.400E+02	0.806E+02	0.827E+02	$\boxed{0.192E+02}$
16	0.137E+00	0.806E-01	0.267E-01	$\boxed{0.577E-02}$	–
17	0.372E+01	0.323E+01	0.372E+01	0.338E+01	$\boxed{0.259E+01}$
18	0.138E+00	$\boxed{0.388E-02}$	0.414E-02	0.389E-02	–

Table 7.7: *PERGAP* at 1,000 Evaluations when starting at $GAP/\sigma \approx 10$

Test function	Method				
	NM	RS9	NMSNR	NMSNV	KW
1	0.985E+02	0.863E+02	0.427E+02	$\boxed{0.148E+02}$	0.806E+02
2	0.536E+01	0.299E+01	0.165E+01	0.135E+01	$\boxed{0.850E-01}$
3	0.963E+00	0.587E+00	0.873E+00	0.720E+00	$\boxed{0.123E-01}$
4	0.356E+00	0.404E+00	0.140E+00	$\boxed{0.131E+00}$	–
5	0.544E+01	0.272E+01	0.180E+01	0.170E+01	$\boxed{0.121E+01}$
6	0.462E+01	0.630E+00	0.147E+00	$\boxed{0.140E+00}$	–
7	0.217E+01	0.568E+00	$\boxed{0.153E+00}$	0.251E+00	0.513E+00
8	0.908E+02	0.618E+02	0.696E+02	$\boxed{0.115E+02}$	0.201E+02
9	0.232E+02	0.103E+02	0.834E+01	0.622E+01	$\boxed{0.223E+01}$
10	0.245E-01	$\boxed{0.224E-01}$	0.294E-01	0.255E-01	–
11	0.399E+02	0.175E+02	0.160E+02	$\boxed{0.121E+02}$	0.822E+87
12	0.132E+00	0.536E-01	$\boxed{0.271E-01}$	0.363E-01	–
13	0.163E+01	0.108E+01	0.176E+00	$\boxed{0.172E+00}$	0.137E+05
14	0.632E+01	0.312E+01	$\boxed{0.239E+01}$	$\boxed{0.239E+01}$	0.296E+01
15	0.559E+02	0.298E+02	0.256E+02	$\boxed{0.110E+02}$	0.125E+02
16	0.137E+00	0.720E-01	$\boxed{0.178E-01}$	0.212E-01	–
17	0.373E+01	0.277E+01	0.225E+01	0.204E+01	$\boxed{0.175E+01}$
18	0.153E+00	0.420E-02	$\boxed{0.391E-02}$	0.405E-02	–

Table 7.8: *PERGAP* at 10,000 Evaluations when starting at $GAP/\sigma \approx 10$

Test function	Method				
	NM	RS9	NMSNR	NMSNV	KW
1	0.985E+02	0.863E+02	0.154E+02	$\boxed{0.560E+01}$	0.734E+02
2	0.536E+01	0.299E+01	0.900E+00	0.772E+00	$\boxed{0.839E-01}$
3	0.963E+00	0.587E+00	0.910E+00	0.896E+00	$\boxed{0.123E-01}$
4	0.356E+00	0.404E+00	0.292E-01	$\boxed{0.271E-01}$	–
5	0.544E+01	0.272E+01	0.146E+01	0.138E+01	$\boxed{0.121E+01}$
6	0.462E+01	0.630E+00	0.936E-01	$\boxed{0.885E-01}$	–
7	0.217E+01	0.569E+00	0.844E-01	$\boxed{0.728E-01}$	0.426E+00
8	0.908E+02	0.618E+02	0.256E+02	$\boxed{0.431E+01}$	0.131E+02
9	0.232E+02	0.103E+02	0.342E+01	0.339E+01	$\boxed{0.162E+01}$
10	0.245E-01	0.223E-01	0.232E-01	$\boxed{0.221E-01}$	–
11	0.399E+02	0.175E+02	0.675E+01	$\boxed{0.385E+01}$	0.822E+87
12	0.132E+00	0.536E-01	0.348E-01	$\boxed{0.298E-01}$	–
13	0.163E+01	0.109E+01	$\boxed{0.157E+00}$	0.161E+00	0.405E+00
14	0.632E+01	0.312E+01	$\boxed{0.180E+01}$	0.183E+01	0.253E+01
15	0.559E+02	0.298E+02	0.818E+01	$\boxed{0.756E+01}$	0.102E+02
16	0.137E+00	0.720E-01	0.869E-02	$\boxed{0.848E-02}$	–
17	0.373E+01	0.277E+01	0.133E+01	$\boxed{0.115E+01}$	0.133E+01
18	0.153E+00	$\boxed{0.419E-02}$	0.420E-02	0.423E-02	–

7.3.1 Discussion of Algorithmic Performance

As in the univariate case, algorithm KW performed quite well on some functions and poorly on others. In those graphs where an asterisk appears next to an algorithm's label in the legend, the performance is not graphed either due to a numerical overflow condition or extremely high values of $PERGAP$. A hyphen appears in the tables for the same reason.

Early during the minimization, method KW is performing the most efficiently when it is doing well. Otherwise, method RS9 is performing the most efficiently. However, method RS9 eventually converges falsely and usually by 1000 function evaluations methods NMSNR and NMSNV are outperforming it. Smaller critical values could make methods NMSNR and NMSNV more efficient early during the minimization (see Section 7.3.2), but are less efficient as the minimization continues.

And, as we observed in the univariate experiment, our proposed algorithm eventually outperforms KW on almost all of the functions by the time the simulation is terminated. Among the two implementations of our algorithm, method NMSNV (the one which uses the variance of the vertex means) is a more efficient implementation than method NMSNR (the one which uses the range of the vertex means).

7.3.2 Sensitivity of NMSNR, NMSNV to Critical Values

When considering the sensitivity of these two implementations to the choice of critical values we included some higher dimensional problems; that is, additional simulations using the Variably Dimensioned function (19-21) and the Extended Rosenbrock function (22-24) with $n = 10, 20$, and 50 were made. These six functions along with the other 18 test functions were used against both methods using critical values corresponding to $\alpha = 0.3, 0.2, 0.1, 0.05$, and 0.01. The average cost required to reduce $PERGAP$ to 10% is reported in Tables 7.9 and 7.10, where the functions are grouped by dimension.

As we saw in the univariate testing, the sensitivity of methods NMSNR and NMSNV to changes in the significance level are relatively small. One could make an argument for smaller critical values for method NMSNV, but there does not appear to be a trend in the sensitivity of method NMSNR. From a theoretical perspective the higher critical values ($\alpha \leq 0.10$) are more appealing; and, a better means towards lowering the number of function evaluations, particularly during the early stages of optimization, might be achieved by a procedure that adapts the critical values for the dependency that is introduced between shrinkages.

7.4 Conclusion

Once again, the two variants of our proposed algorithm, NMSNR and NMSNV, have performed extremely well in comparison to the other algorithms tested. However, our NMSN implementations were not as dominant in higher dimensions as in the univariate experiment. The added complexity of minimizing in several dimensions in addition to the complexity of the multivariate functions we expect contributed to this end.

The difference in the performance of the two implementations, NMSNR and NMSNV, demonstrates the possibility of an even better implementation using a different testing procedure. In particular, a testing procedure that adjusts the critical values to account for the dependency introduced between simplex shrinkages

Table 7.9: Effect of Critical Value on Average Number of Function Evaluations Needed by NMSNR to Reduce *PERGAP* to 10% when starting at $GAP/\sigma \approx 1$

Dimension	Test function	Level of Significance (α)				
		0.30	0.20	0.10	0.05	0.01
2	4	545	489	419	362	449
	10	126	126	126	126	126
	16	15	13	12	12	10
3	3	7	7	7	7	7
	5	713	690	860	811	1278
	12	612	7	7	548	7
4	6	56	50	46	39	38
	14	3638	4285	5300	5253	5230
	17	18	18	18	18	17
6	2	2301	3683	2799	2641	2499
8	9	17418	16900	13486	16074	18708
	13	21	20	20	20	20
9	7	3827	3766	4576	4360	4053
	18	20	20	20	20	20
10	19	81	62	62	56	52
20	20	3310	4462	4283	3935	3455
50	21	8363	6674	5570	4899	2816
Average		2415	2427	2212	2304	2281

Note: PERGAP was not reduced to 10% on functions 1, 8, 11, 15, or 22-24 (Extended Rosenbrock with $n = 10, 20, 50$ respectively) prior to termination.

Table 7.10: Effect of Critical Value on Average Number of Function Evaluations Needed by NMSNV to Reduce *PERGAP* to 10% when starting at $GAP/\sigma \approx 1$

Dimension	Test function	Level of Significance (α)				
		0.30	0.20	0.10	0.05	0.01
2	4	538	456	625	415	413
	10	126	126	126	126	126
	16	17	16	15	13	12
3	3	7	7	7	7	7
	5	898	729	896	1166	1155
	12	602	624	3267	3208	3267
4	6	67	47	46	63	58
	14	2873	3498	3564	3588	3604
	17	18	20	17	18	17
6	2	2955	2971	2366	2935	3907
8	9	17252	17243	17220	16867	16801
	13	47	42	41	37	34
9	7	3758	3759	3749	3779	3829
	18	20	20	20	20	20
10	19	65	65	65	65	65
20	20	4956	5029	4855	4946	5060
50	21	10844	10844	10844	10856	10832
Average		2649	2676	2807	2829	3971

Note: PERGAP was not reduced to 10% on functions 1, 8, 11, 15, or 22-24 (Extended Rosenbrock with $n = 10, 20, 50$ respectively) prior to termination.

may be a promising direction. As Barton and Ivey (1993) have demonstrated, the probability of a shrinkage on a constant function decreases as the dimension of the parameter space increases. This fact is apparent from the probabilities given in Table 3.3 as well. Since our heuristic procedure does not have the correct size except following a shrinkage, the test may be increasing the sample size more often than it should. Therefore, more work is needed with respect to tuning the algorithm to improve its performance even more.

Chapter 8

Conclusion

8.1 Summary

In Chapter 1 we introduced the popular nonlinear optimization method of Nelder and Mead (1965). The algorithm is popular because of its conceptual simplicity and it is an efficient method for many practical problems. Although much work has been done to improve its performance for deterministic optimization, a general proof of deterministic convergence has not been discovered.

In Chapter 2 we reviewed several deterministic and stochastic applications of the Nelder-Mead simplex algorithm. It is with respect to the latter that our research has been focused. Because of the robustness of the algorithm to small perturbations in the values of the objective function, this method has found widespread use in stochastic applications. However, this algorithm's performance degrades as the noise level is increased.

In Chapter 3 we proved that the simplex converges to a point *w.p. 1* on a constant function with additive noise for one and two-dimensional problems. A proof for the general n-dimensional problem proved elusive, but we offer strong empirical evidence of this result. A successful general proof may require a different approach for measuring the size of the simplex because of the geometry of the reflection operation when $n > 3$.

In Chapter 4 we demonstrated that the simplex can still shrink to a point on an unbounded univariate linear function obscured by noise. We provided evidence for this result both analytically using Markov chain theory and empirically using

Monte Carlo simulation. Our Markov chain analysis led us to a new modification of the Nelder-Mead simplex algorithm which increases the number of observations taken at each vertex to reduce the noise level. The rule for changing the simplex size is based on familiar statistical hypothesis tests for equal means.

In Chapter 5 we implement the proposed algorithm, which is only a heuristic rule by nature, and test its performance against the original Nelder-Mead algorithm, the previously known best-performing modified simplex algorithm and the Kiefer-Wolfowitz stochastic approximation algorithm. Our test results demonstrate the superiority of our approach for stochastic optimization.

In Chapter 6 we discussed the extension of our proposed algorithm to the general n-dimensional problem. A number of possible candidate procedures are discussed and two of them were implemented in a multivariate test which we reported in Chapter 7. Use of the variance of the vertex means yielded better results than the use of the range of these means even though these two implementations were normalized to have the same size test following a shrinkage operation.

8.2 Contributions

Our proof of convergence of the Nelder-Mead simplex on stochastic functions with constant expectation is a new addition to the limited convergence results that are known to exist. Additionally, our analytical research into the convergence of the simplex on a one-dimensional linear function contaminated with noise demonstrates that Nelder-Mead is clearly the wrong algorithm to use for stochastic optimization despite its robustness to small perturbations in the function's values. Furthermore, this analysis led us to the development of a new simplex method, NMSN, for stochastic optimization, which adjusts the number of observations taken at each vertex by a statistical test for equal means.

Our empirical studies of NMSN yielded a significant improvement in the quality of the minimum expected values obtained as compared to Nelder-Mead and the previously known best-performing variant RS9. And furthermore, NMSN was as efficient as Kiefer-Wolfowitz for most of our test functions. Therefore, NMSN has the potential to provide better answers than other known methods used for stochastic optimization.

8.3 Future Research

There are three general areas for future research implied by our results. First, a general proof of convergence of the simplex on constant functions with noise remains to be discovered. Secondly, a more interesting proof would be that our proposed algorithm converges to the optimum of the expected function. We are encouraged that such a result may be true given the empirical comparisons with the Kiefer-Wolfowitz algorithm which is known to converge to the optimizer *w.p. 1.* Finally, other modifications to tune the algorithm's performance should be pursued. A few possibilities for tuning include: (1) adjustment of the algorithm's parameters to increase the rate of convergence when the simplex is near the optimum (*e.g.* restoring the shrinkage and contraction coefficients to their standard values of 0.5); (2) apply the testing procedure only after a shrinkage, which would eliminate the dependency problem and perhaps reduce the number of function evaluations required to reach a certain level of *PERGAP*; (3) develop better heuristic rules for changing the vertex sample size, particularly rules that are more robust to the lack of independence (*e.g.* sequential procedures that incorporate information from iterations since the last shrinkage); and (4) incorporation of some of the better modifications discussed in Section 1.6, *e.g.* the weighted centroid approach.

Bibliography

Agapiou, J. S. (1992). The Optimization of Machining Operations Based on a Combined Criterion, Part 1: The Use of Combined Objectives in Single-Pass Operations. *Transactions of the ASME*, *114*, 500–507.

Andradottir, S. (1994). Private communication via electronic mail. Univeristy of Wisconsin.

Arnold, S. F. (1981). *The Theory of Linear Models and Multivariate Analysis.* John Wiley & Sons, New York.

Arnold, S. F. (1990). *Mathematical Statistics.* Prentice-Hall, Englewood Cliffs, New Jersey.

Barton, R. R. (1980). Comparing Optimization Techniques for Experimental Design. RCA Laboratories, Princeton, New Jersey.

Barton, R. R. (1984). Minimization Algorithms for Functions with Random Noise. *American Journal of Mathematical and Management Sciences*, *4*, 109–138.

Barton, R. R. (1987). Testing Strategies for Simulation Optimization. In Thesen, A., Grant, H., & Kelton, W. D. (Eds.), *Proceedings of the 1987 Winter Simulation Conference*, pp. 391–401 Piscataway, New Jersey. Institute of Electrical and Electronics Engineers.

Barton, R. R., & Ivey, Jr., J. S. (1993). Nelder-Mead Simplex Modifications for Simulation Optimization. Submitted for publication in Management Science.

Benyon, P. R. (1976). Remark AS R15. Function Minimization Using a Simplex Procedure. *Applied Statistics*, *25*, 97.

Betteridge, D., Wade, A. P., & Howard, A. G. (1985a). Reflections on the Modified Simplex-I. *Talanta*, *32*, 709–722.

Betteridge, D., Wade, A. P., & Howard, A. G. (1985b). Reflections on the Modified Simplex-II. *Talanta*, *32*, 723–734.

Billingsley, P. (1986). *Probability and Measure* (second edition). John Wiley & Sons, New York.

Box, G. E. P. (1957). Evolutionary Operation: A Method for Increasing Industrial Productivity. *Applied Statistics*, *6*, 81–101.

Box, G. E. P., & Wilson, K. B. (1951). On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society, Series B*, *13*, 1–45.

Brooks, S. H., & Mickey, M. R. (1961). Optimum Estimation of Gradient Direction in Steepest Ascent Experiments. *Biometrics*, *17*, 48–56.

Burgess, D. D. (1985). Optimization of Multielement Instrumental Neutron Activation Analysis. *Analytical Chemistry*, *57*, 1433–1436.

Burgess, D. D., & Hayumbu, P. (1984). Simplex Optimization by Advance Prediction for Single-Element Instrumental Neutron Activation Analysis. *Analytical Chemistry*, *56*, 1440–1443.

Busing, W. R., & Matsui, M. (1984). The Application of External Forces to Computational Models of Crystals. *Acta Crystallographica. Section A*, *40*, 532.

Caceci, M. S., & Cacheris, W. P. (1984). Fitting Curves to Data: The Simplex Algorithm is the Answer. *Byte*, *9*, 340–362.

Chambers, J. M., & Ertel, J. E. (1974). Remark AS R11. Function Minimization Using a Simplex Procedure. *Applied Statistics*, *23*, 250–251.

Cheok, K. C., Hu, H., & Loh, N. K. (1988). Modeling and Identification of a Class of Servomechanism Systems with Slip-Stick Friction. *Transactions of the ASME*, *110*, 324–328.

Dantzig, G. B. (1963). *Linear Progamming and Extensions.* Princeton University Press, Princeton, New Jersey.

De Smet, D. J., & Ord, J. L. (1989). Analysis of Ellipsometric Data in Electrochemical Systems Using the Simplex Algorithm. *Journal of the Electrochemical Society*, *136*, 2841–2845.

Deming, S. N., & Morgan, S. L. (1973). Simplex Optimization of Variables in Analytical Chemistry. *Analytical Chemistry*, *45*, 278A–283A.

Deming, S. N., & Parker, Jr., L. R. (1978). A Review of Simplex Optimization in Analytical Chemistry. *CRC Critical Review of Analytical Chemistry*, *7*, 187–202.

Dennis, Jr., J. E. (1994). Private communication via electronic mail. Rice Univeristy.

Dennis, Jr., J. E., & Schnabel, R. E. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice Hall, Inc., Englewood Cliffs, New Jersey.

Dennis, Jr., J. E., & Torczon, V. (1991). Direct Search Methods on Parallel Machines. *SIAM Journal of Optimization*, *1*, 448–474.

Dennis, Jr., J. E., & Woods, D. J. (1987). Optimization on Microcomputers: the Nelder-Mead Simplex Algorithm. In Wouk, A. (Ed.), *New Computing Environments: Microcomputers in Large Scale Computing*, pp. 116–122. SIAM, Philadelphia.

Ebdon, L., Cave, M. R., & Mowthorpe, D. J. (1980). Simplex Optimisation of Inductively Coupled Plasmas. *Analytica Chimica Acta*, *115*, 179–187.

Elling, J. W., de Koning, L. J., Pinkse, F. A., & Nibbering, N. M. M. (1989). Computer-Controlled Simplex Optimization on a Fourier Transform Ion Cyclotron Resonance Mass Spectrometer. *Analytical Chemistry*, *61*, 330–334.

Fletcher, R. (1981). *Practical Methods of Optimization.* John Wiley & Sons, New York.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical Optimization.* Academic Press, New York.

Harter, H. L. (1960). Tables of Range and Studentized Range. *Annals of Mathematical Statistics*, *31*, 1122–1147.

Hensley, D., Smith, P., & Woods, D. (1988). Simplex Distortions in Nelder-Mead Reflections. Tech. rep. 8801, IMSL, Inc., Houston, TX.

Hill, I. D. (1978). Remark AS R28. Function Minimization Using a Simplex Procedure. *Applied Statistics*, *27*, 380–382.

Hillstrom, K. E. (1977). A Simulation Test Approach to the Evaluation of Nonlinear Optimization Algorithms. *ACM Transactions on Mathematical Software*, *3*, 305–315.

Hooke, R., & Jeeves, T. A. (1961). Direct Search Solution of Numerical and Statistical Problems. *Journal of the Association for Computing Machinery*, *8*, 212–229.

Ibarra, J. V., & Lazaro, J. J. (1985). Lignite Sulfonation Optimized by a Modified Simplex Method. *Industrial & Engineering Chemistry Product Research and Development*, *24*, 604–607.

Jackson, R. H. F., Boggs, P. T., Nash, S. G., & Powell, S. (1991). Guidelines for Reporting Results of Computational Experiments. Report of the ad hoc committee. *Mathematical Programming*, *49*, 413–425.

Jang, G.-W., & Rajeshwar, K. (1988). Computer Simulation of Differential Scanning Calorimetry: Influence of Thermal Resistance Factors and Simplex Optimization of Peak Resolution. *Analytical Chemistry*, *60*, 1003–1009.

Karim, M. R., & Mal, A. K. (1990). Inversion of Leaky Lamb Wave Data by Simplex Algorithm. *Journal of the Acoustical Society of America*, *88*, 482–491.

Kesten, H. (1958). Accelerated Stochastic Approximation. *Annals of Mathematical Statistics*, *29*, 41–59.

Khuri, A. (1987). *Response Surfaces*. Marcel Dekker, New York.

Kiefer, J., & Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *Annals of Mathematical Statistics*, *23*, 462–466.

Lawitts, J. A., & Biggers, J. D. (1991). Optimization of Mouse Embryo Culture Media Using Simplex Methods. *Journal of Reproduction & Fertility, 91*, 543–556.

Luenberger, D. G. (1984). *Linear and Nonlinear Programming* (second edition). Addison-Wesley, Reading, Massachusettes.

Marsili-Libelli, S., & Castelli, M. (1987). An Adaptive Search Algorithm for Numerical Optimization. *Applied Mathematics and Computation, 23*, 341–357.

Mehta, N. C., & Allen, C. W. (1993). Segmented Mirror Alignment with Far-Field Optimization in the Presence of Atmospheric Turbulence. *Applied Optics, 32*, 2664–2673.

More, J. J., Garbow, B. S., & Hillstrom, K. E. (1981). Testing Unconstrained Optimization Software. *ACM Transactions on Mathematical Software, 7*, 17–41.

Morgan, S. L., & Deming, S. N. (1974). Simplex Optimization of Analytical Chemical Methods. *Analytical Chemistry, 46*, 1170–1181.

Nash, J. C. (1979). *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*. John Wiley & Sons, New York.

Nelder, J. A., & Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal, 7*, 308–313.

Neter, J., Wasserman, W., & Kutner, M. H. (1985). *Applied Linear Statistical Models*. Richard D. Irwin, Inc., Homewood, Illinois.

Olansky, A. S., Parker, Jr., L. R., Morgan, S. L., & Deming, S. N. (1977). Automated Development of Analytical Chemical Methods. *Analytica Chimica Acta*, *95*, 107–133.

Olsson, D. M. (1974). A Sequential Simplex Program for Solving Minimization Problems. *Journal of Quality Technology*, *6*, 53–57.

O'Neill, R. (1971). Algorithm AS 47. Function Minimization Using a Simplex Procedure. *Applied Statistics*, *20*, 338–345.

Parker, Jr., L. R., Cave, M. R., & Barnes, R. M. (1985). Comparison of Simplex Algorithms. *Analytica Chimica Acta*, *175*, 231–237.

Parkinson, J. M., & Hutchinson, D. (1972). An Investigation into the Efficiency of Variants of the Simplex Method. In Lootsma, F. A. (Ed.), *Numerical Methods for Non-linear Optimization*, pp. 115–135. Academic Press, New York.

Phillips, G. R., & Eyring, E. M. (1988). Error Estimation Using the Sequential Simplex Method in Nonlinear Least Squares Data Analysis. *Analytical Chemistry*, *60*, 738–741.

Powell, M. J. D. (1964). An Efficient Algorithm for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives. *The Computer Journal*, *7*, 155–162.

Prothero, W. A., Taylor, W. J., & Eickemeyer, J. A. (1988). A Fast, Two-Point, Three-Dimensional Raytracing Algorithm Using a Simple Step Search Method. *Bulletin of the Seismological Society of America*, *78*, 1190–1198.

Prugger, A. F., & Gendzwill, D. J. (1988). Microearthquake Location: A Nonlinear Approach that makes use of a Simplex Stepping Procedure. *Bulletin of the Seismological Society of America, 78*, 799–815.

Rabinowitz, N. (1988). Microearthquake Location by means of Nonlinear Simplex Procedures. *Bulletin of the Seismological Society of America, 78*, 380–384.

Rainey, L. M., & Purdy, W. C. (1977). Simplex Optimization of the Separation of Phospholipids by High-Pressure Liquid Chromatography. *Analytica Chimica Acta, 93*, 211–219.

Reklaitis, G. V., Ravindran, A., & Ragsdell, K. M. (1983). *Engineering Optimization.* John Wiley & Sons, New York.

Rhines, T. D., & Arnold, M. A. (1988). Simplex Optimization of a Fiber-Optic Ammonia Sensor Based on Multiple Indicators. *Analytical Chemistry, 60*, 76–81.

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *Annals of Mathematical Statistics, 22*, 400–407.

Routh, M. W., Swartz, P. A., & Denton, M. B. (1977). Performance of the Super Modified Simplex. *Analytical Chemistry, 49*, 1422–1428.

Ruppert, D. (1994). Private communication via electronic mail. Cornell University.

Ryan, B. P., Barr, R. L., & Todd, H. D. (1980). Simplex Techniques for Nonlinear Optimization. *Analytical Chemistry, 52*, 1460–1467.

Silver, G. L. (1981). Space Modification: An Alternative Approach to Chemistry Problems Involving Geometry. *Journal of Computational Chemistry*, *2*, 478–482.

Smith, D. K. (1986). A Minimization Solver for Many Parameters. *Journal of Quality Technology*, *18*, 248–254.

Spendley, W., Hext, G. R., & Himsworth, F. R. (1962). Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. *Technometrics*, *4*, 441–461.

Steck, L., & Prothero, Jr., W. A. (1989). Seismic Calibration Using the Simplex Algorithm. *Bulletin of the Seismological Society of America*, *79*, 1618–1628.

Subrahmanyam, M. B. (1989). An Extension of the Simplex Method to Constrained Nonlinear Optimization. *Journal of Optimization Theory and Applications*, *62*, 311–319.

Taylor, H. M., & Karlin, S. (1984). *An Introduction to Stochastic Modeling*. Academic Press, Inc., New York.

Thompson, J. R. (1989). *Empirical Model Building*. John Wiley & Sons, New York.

Torczon, V. (1991). On the Convergence of the Multidirectional Search Algorithm. *SIAM Journal of Optimization*, *1*, 123–145.

Torczon, V. (1993). On the Convergence of Pattern Search Algorithms. Tech. rep. TR93-10, Rice University, Dept. of Mathematical Sciences, Houston, TX 77251.

Van Der Wiel, P. F. A. (1980). Improvement of the Super-Modified Simplex Optimization Procedure. *Analytica Chimica Acta*, *122*, 421–433.

Van Der Wiel, P. F. A., Maassen, R., & Kateman, G. (1983). The Symmetry-Controlled Simplex Optimization Procedure. *Analytica Chimica Acta*, *153*, 83–92.

Vance, J. B., Hassani, F. P., & Mottahed, P. (1988). Improved Determination of Microseismic Source Location Using a Simplex Technique. *IEEE Transactions on Industry Applications*, *24*, 666–671.

Venter, J. H. (1967). On Convergence of the Kiefer-Wolfowitz Approximation Procedure. *Annals of Mathematical Statistics*, *38*, 1031–1036.

Wood, J. R. (1993). Calculation of Fluid-Mineral Equilibria Using the Simplex Algorithm. *Computers and Geosciences*, *19*, 23–39.

Woods, D. J. (1985). An Interactive Approach for Solving Multi-Objective Optimization Problems. Tech. rep. TR85-5, Rice University, Dept. of Mathematical Sciences, Houston, TX 77251.

Young, P. (1976). Optimization in the Presence of Noise. In Dixon, L. C. W. (Ed.), *Optimization in Action*, pp. 517–573. Academic Press, London.

Zangwill, W. I. (1969). *Nonlinear Programming: A Unified Approach*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Appendix A

Graphs of Univariate Test Results

The complete set of graphs of the results from the univariate experiment described in Chapter 5 are presented here for further reference. The graphs illustrate the average level of percent gap remaining between the expected function value at the current point and the expected function value at the minimum (all of which are zero) versus the average cost in terms of the number of function evaluations required for each of the four algorithms tested. Each test function and starting point combination are plotted twice. The graphs on the left side of the page use a linear scale for *PERGAP* while those on the right use a logarithmic scale for *PERGAP*. The linear scale is more illustrative for practical considerations; whereas, the logarithmic scale is more illustrative for theoretical considerations. Note the logarithmic scaling of the horizontal axis and that the lines for methods NM and RS9 typically stop well short of the lines for methods KW and NMSN. This is a result of the different stopping criteria used in the experiment. In general, algorithms NM and RS9 stopped as a result of the simplex size falling below 10^{-10}, algorithm NMSN stopped as a result of exceeding 50,000 total function evaluations, and algorithm KW stopped upon reaching 10,000 iterations. All of the graphs display the performance of each algorithm up through the first 20,000 function evaluations.

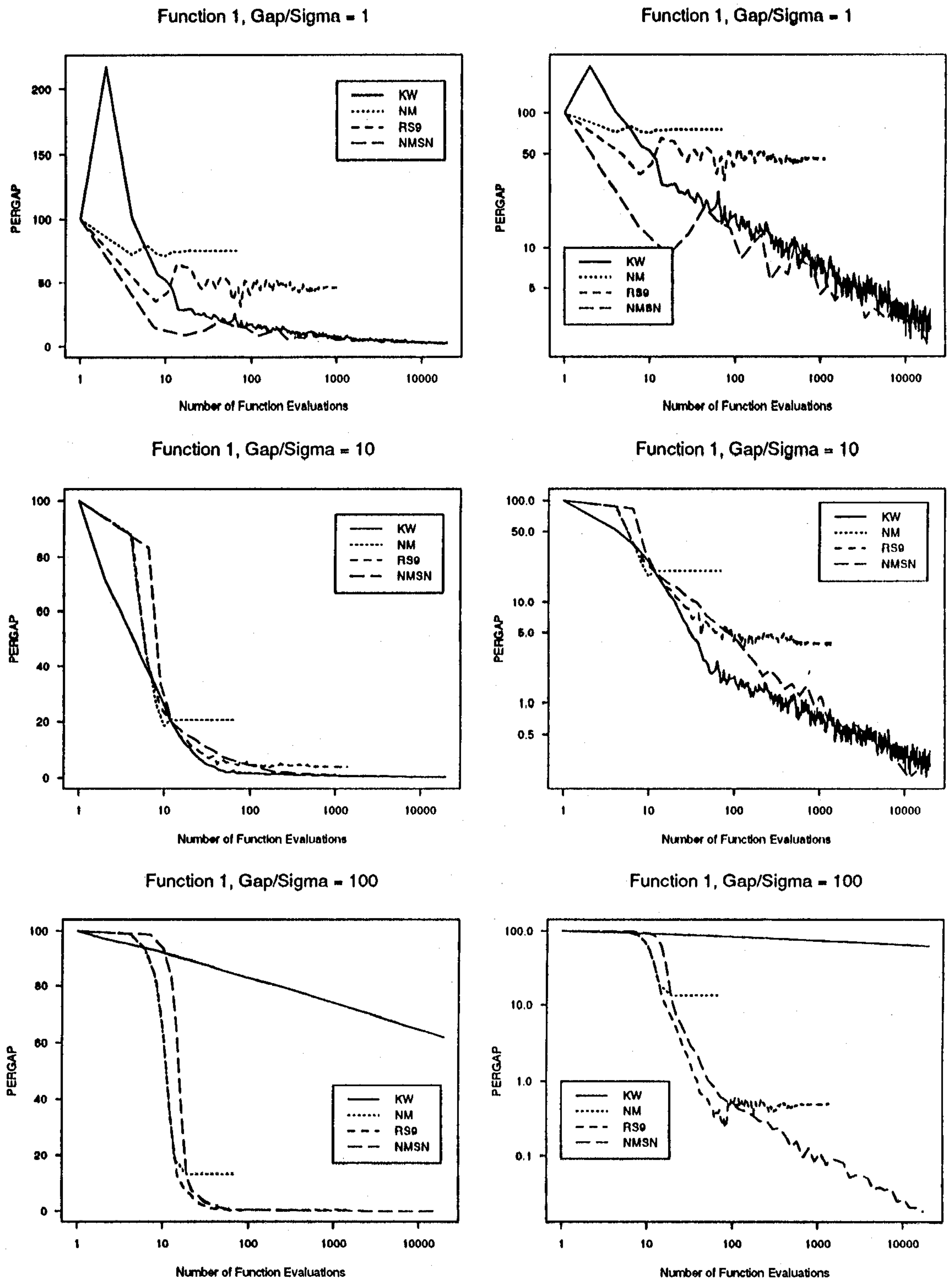


Figure A.1: Test Results for Univariate Function 1

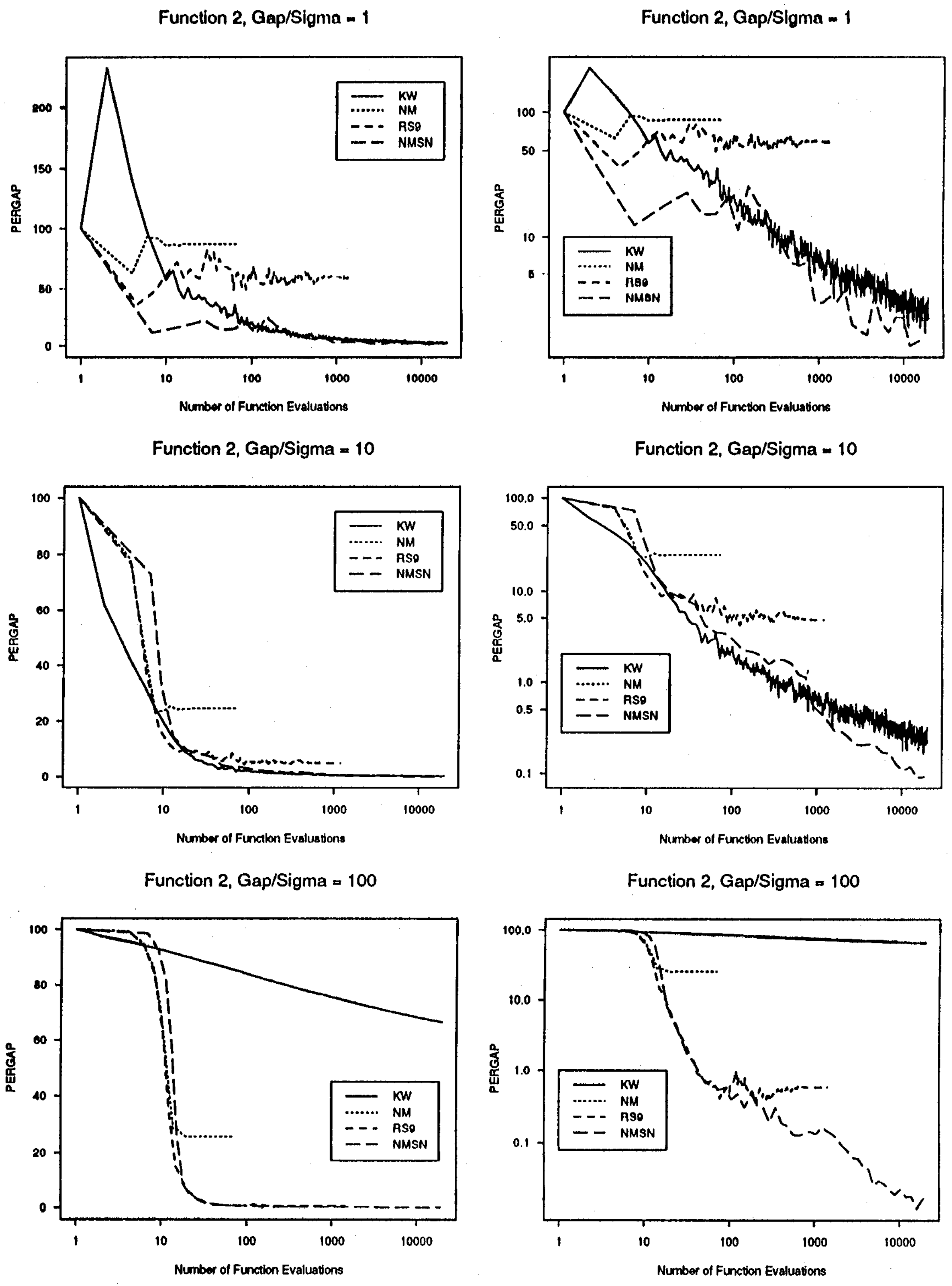


Figure A.2: Test Results for Univariate Function 2

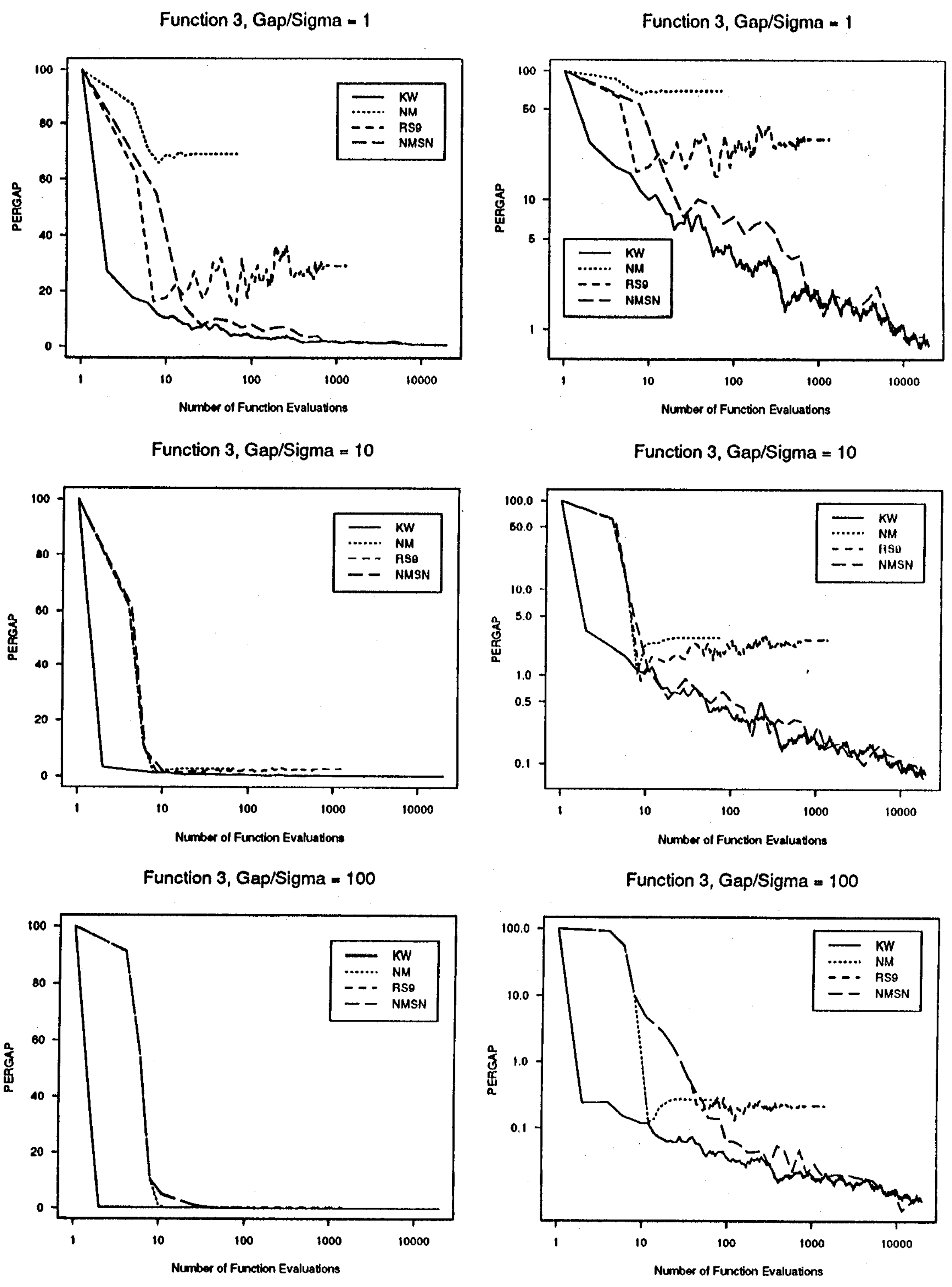


Figure A.3: Test Results for Univariate Function 3

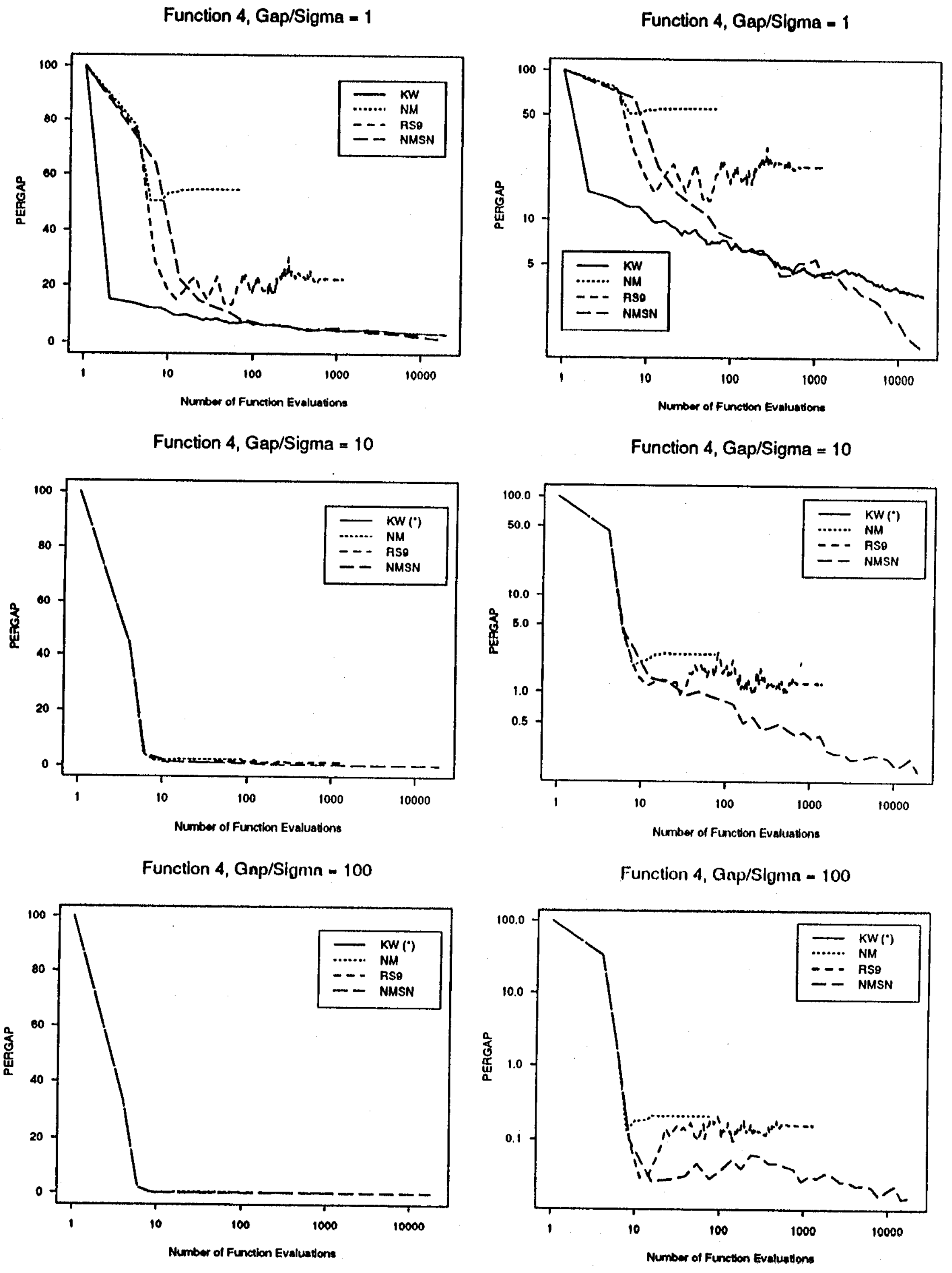


Figure A.4: Test Results for Univariate Function 4

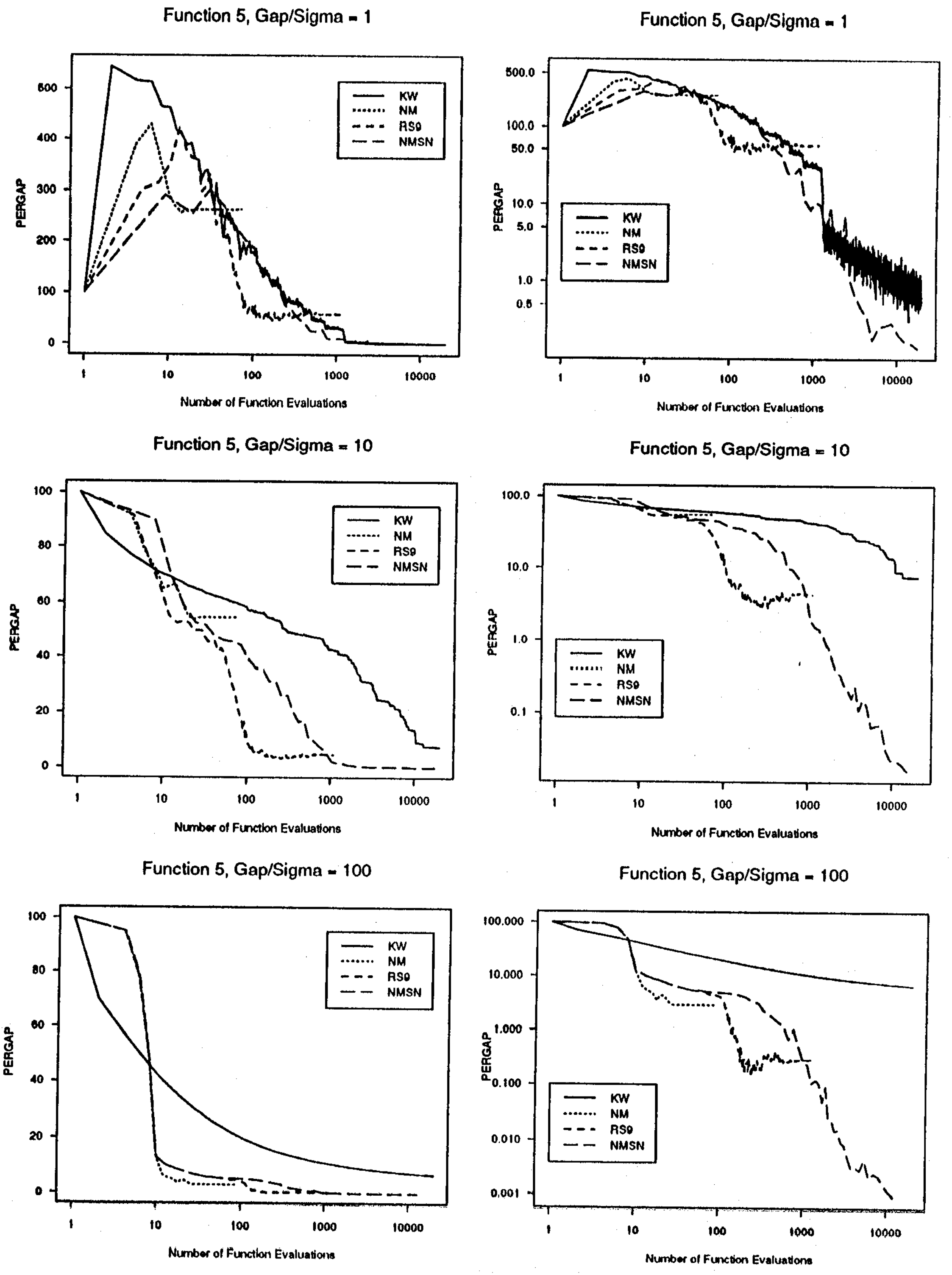


Figure A.5: Test Results for Univariate Function 5

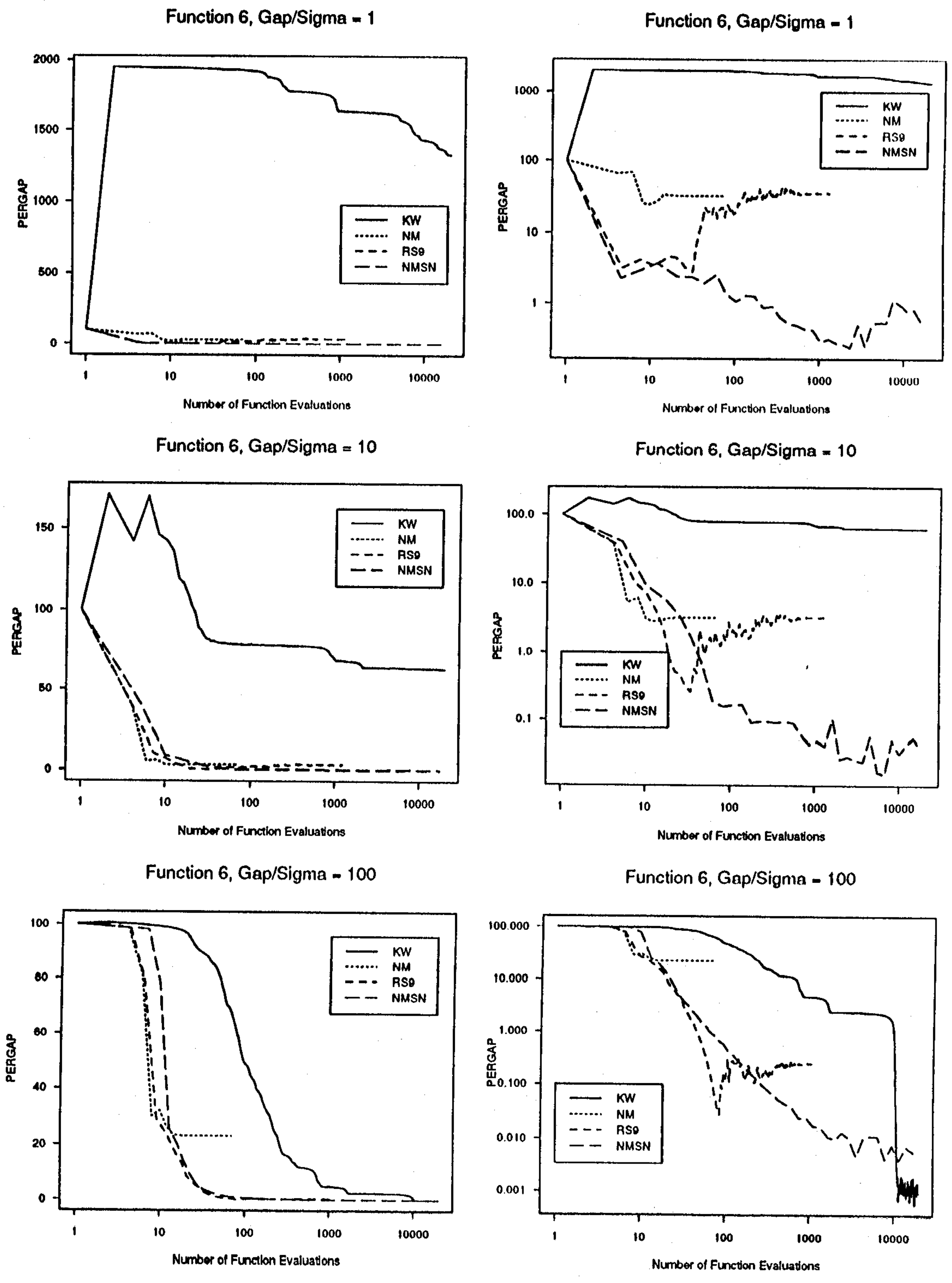


Figure A.6: Test Results for Univariate Function 6

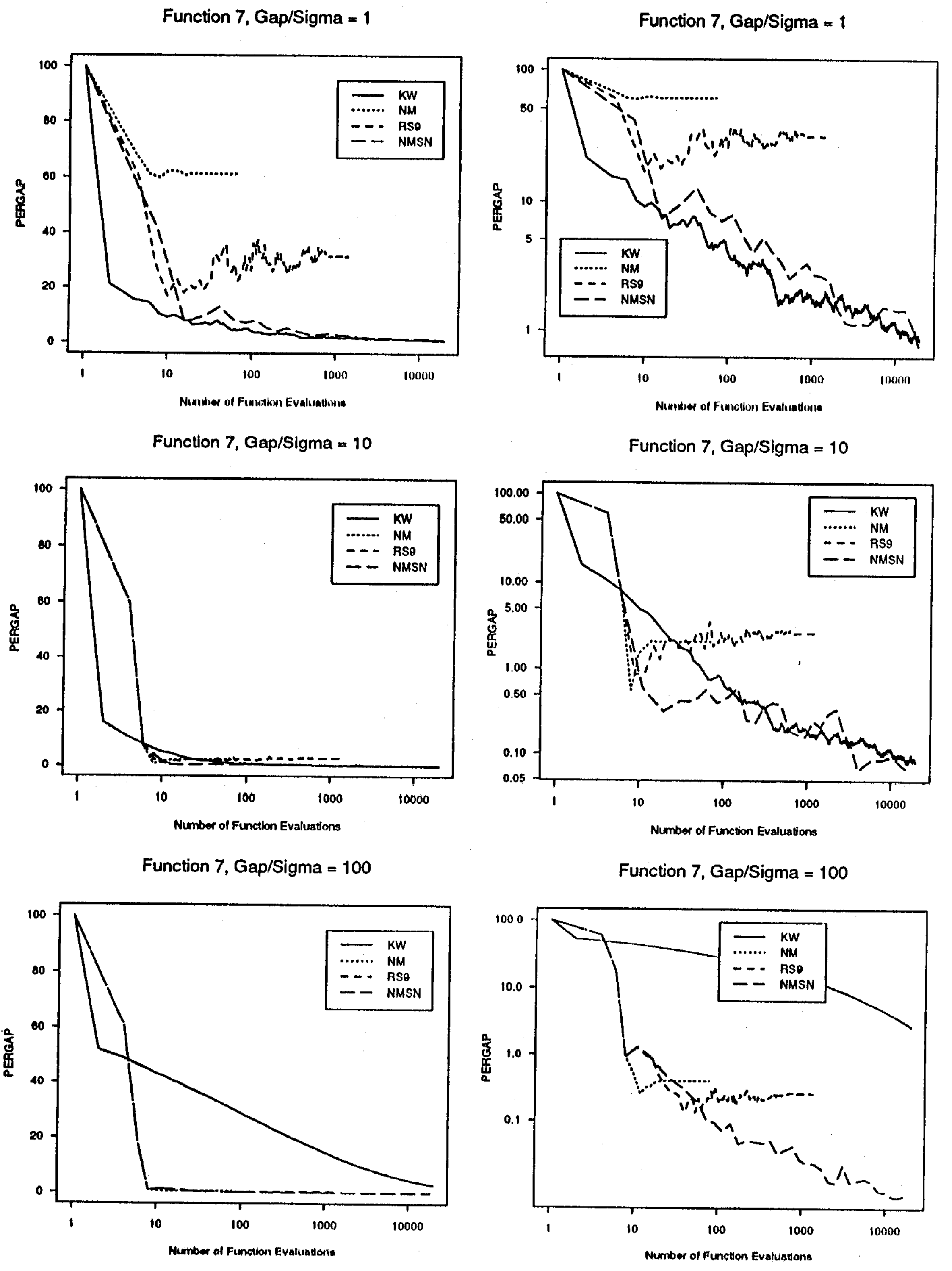


Figure A.7: Test Results for Univariate Function 7

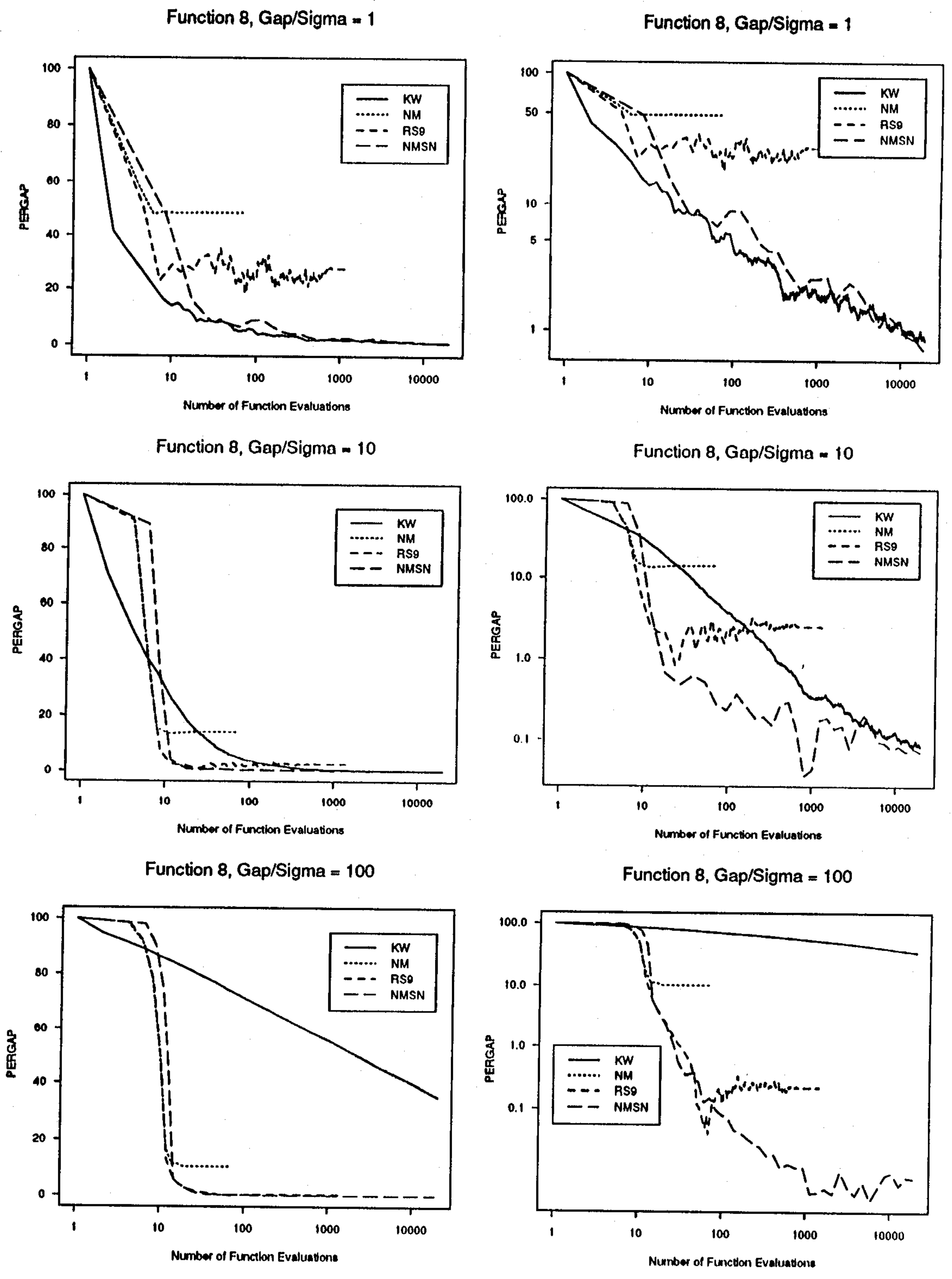


Figure A.8: Test Results for Univariate Function 8

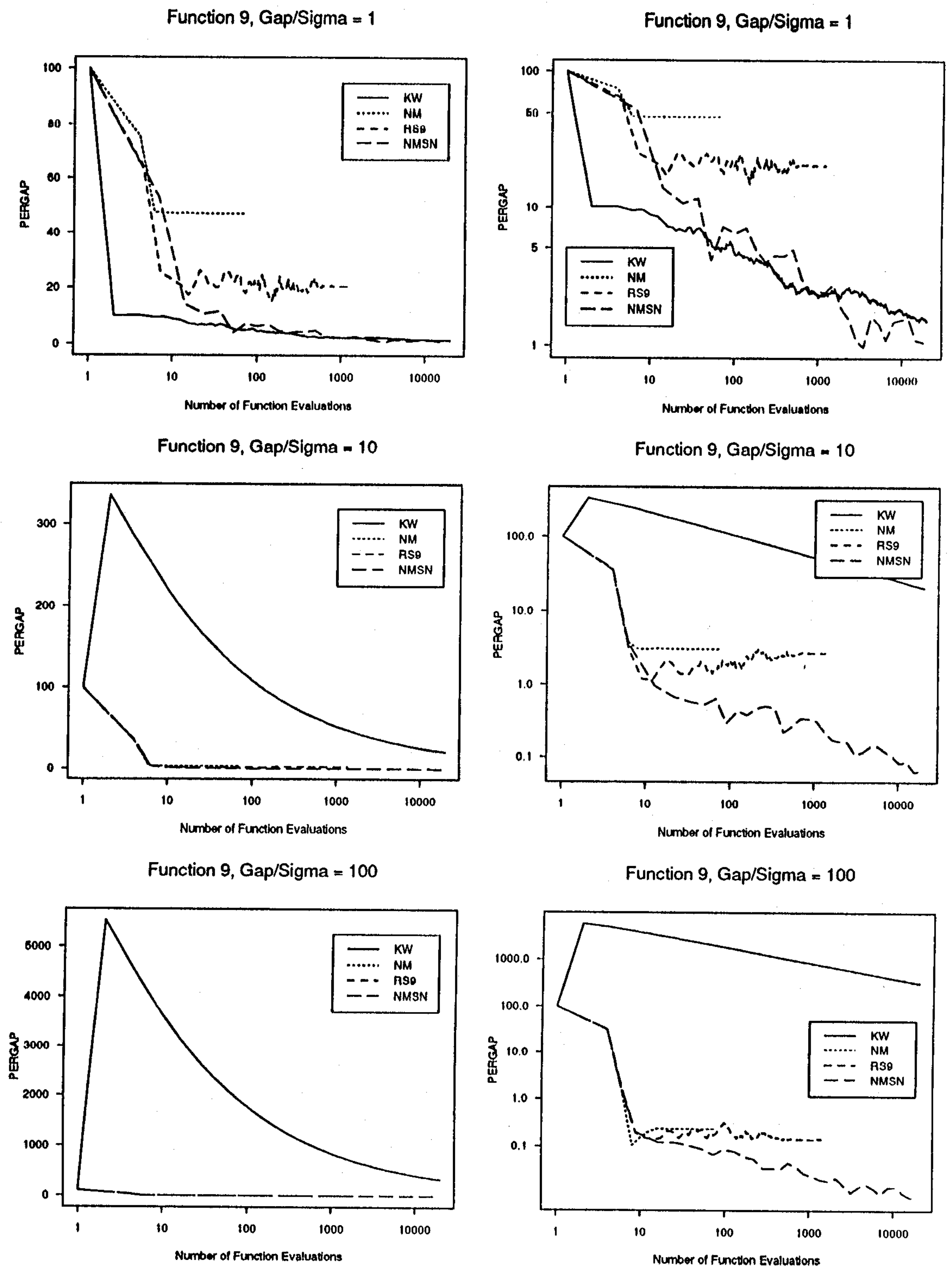


Figure A.9: Test Results for Univariate Function 9

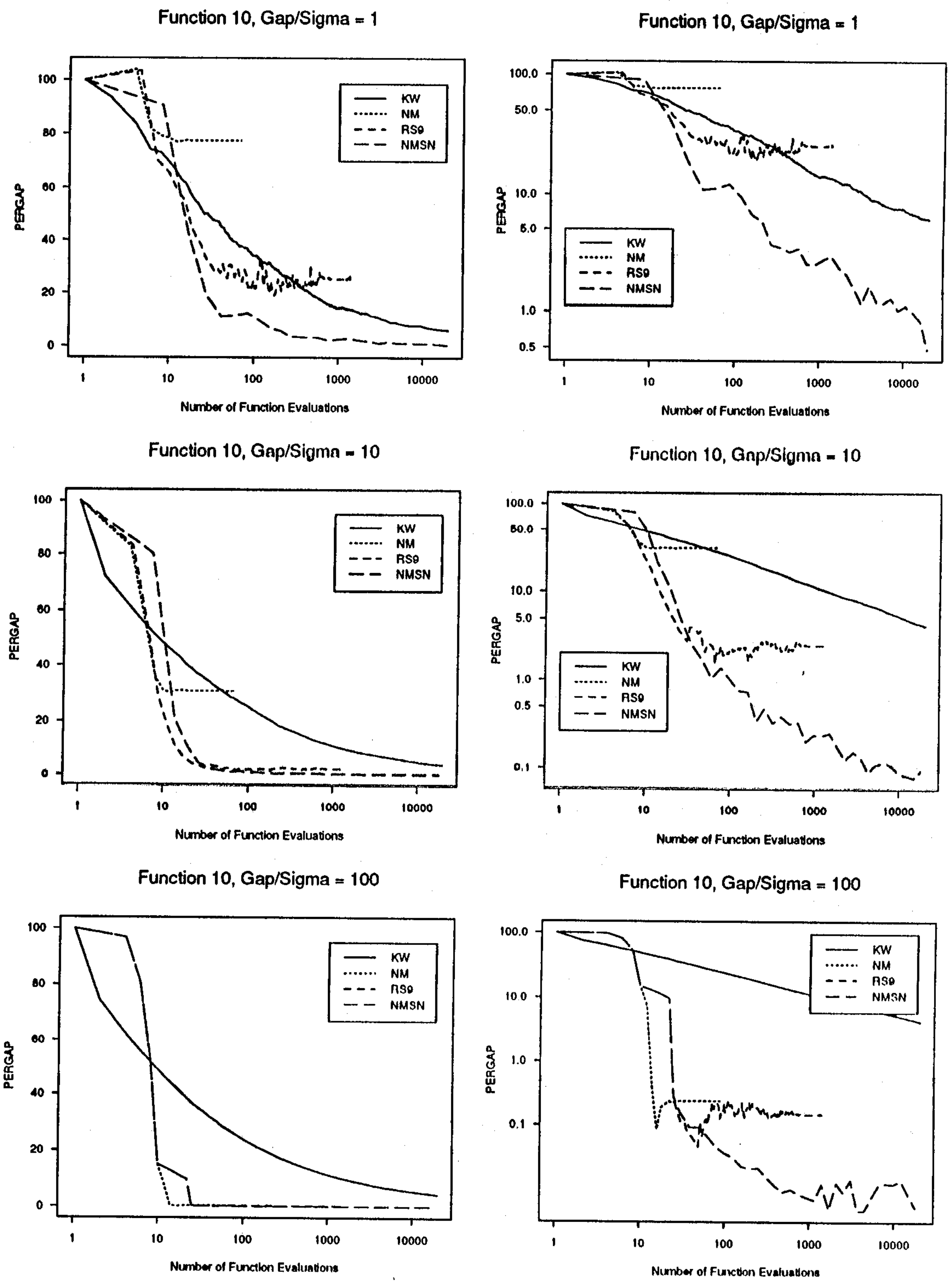
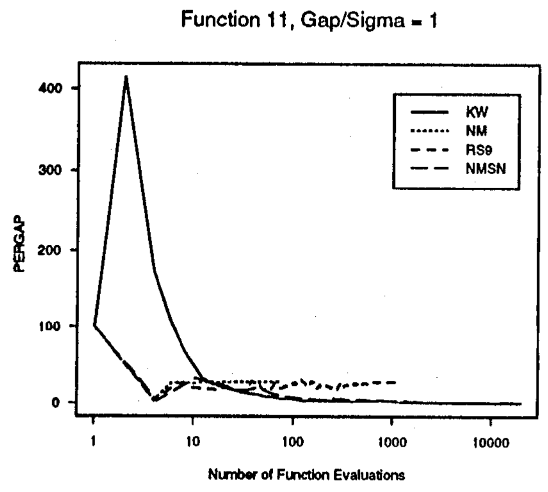
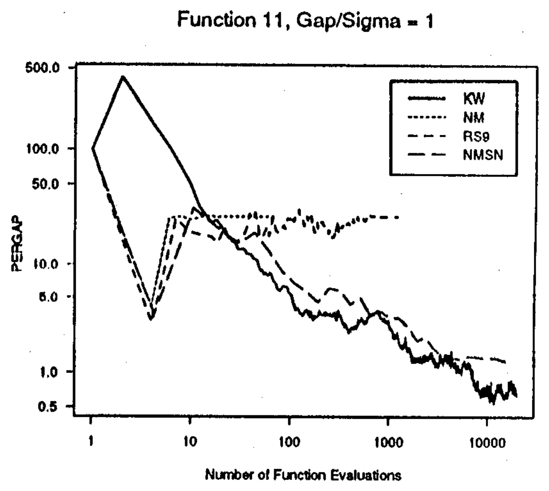
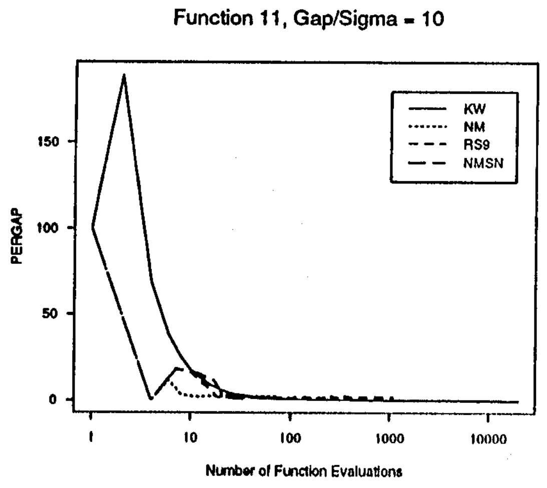
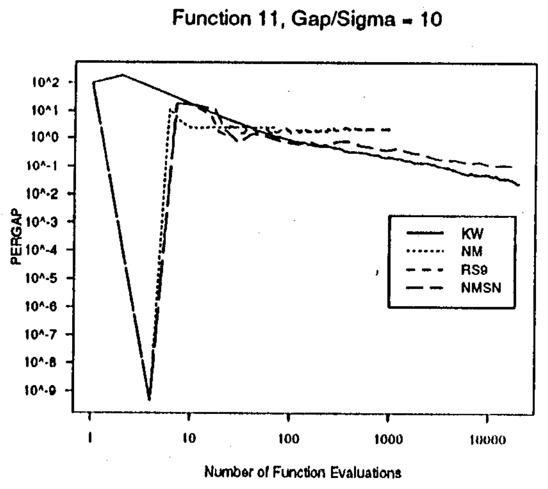


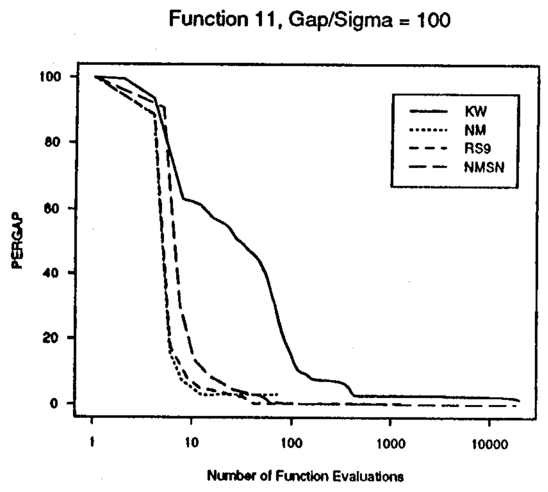
Figure A.10: Test Results for Univariate Function 10











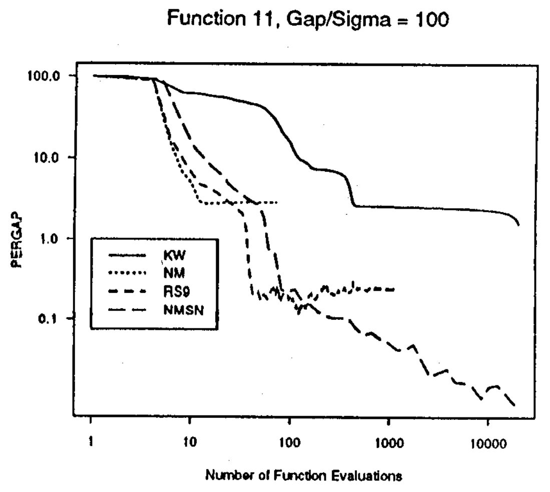


Figure A.11: Test Results for Univariate Function 11

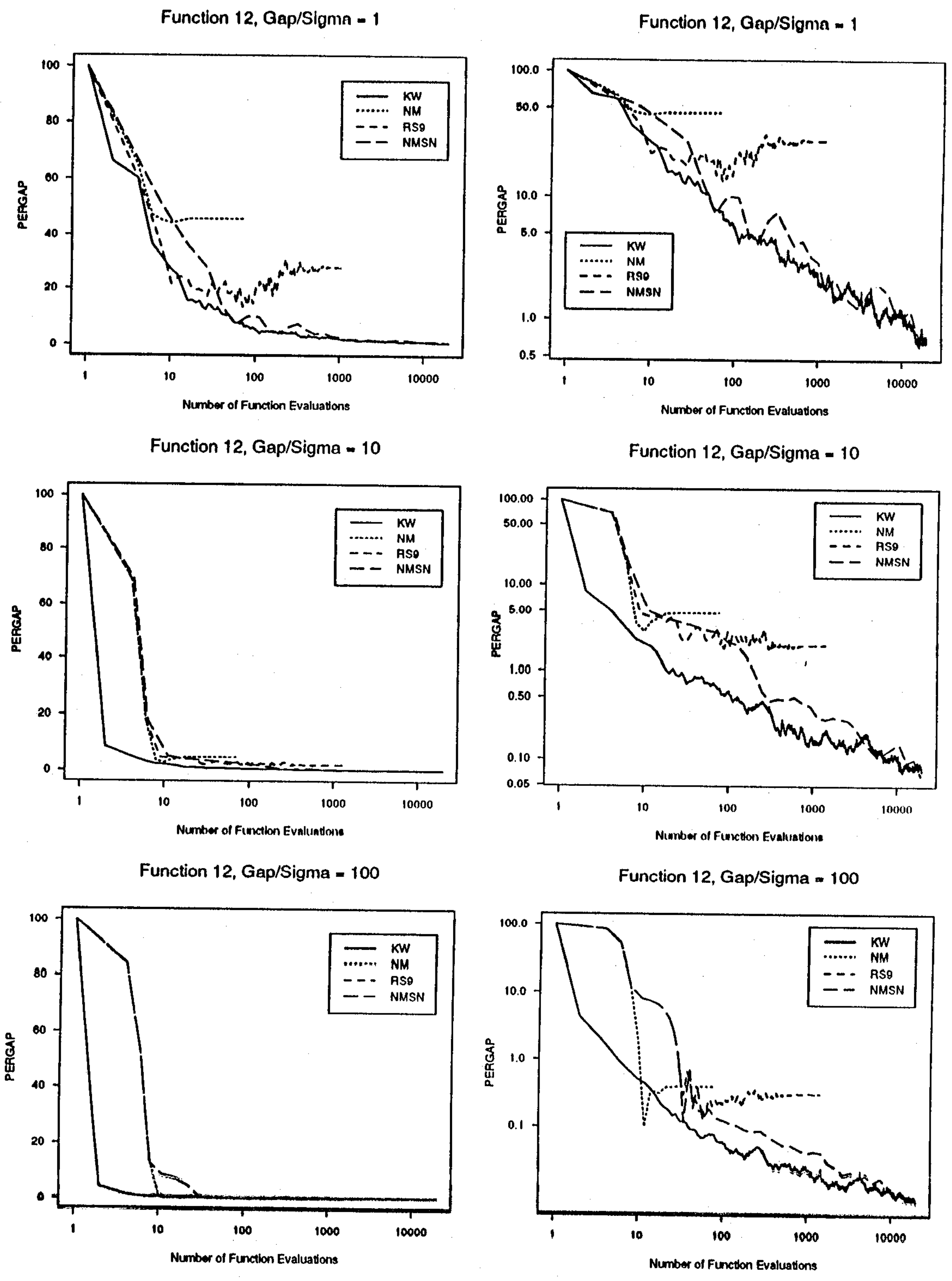


Figure A.12: Test Results for Univariate Function 12

Appendix B

Tabular Output of Univariate Testing

The average value of several variables at termination for each design point in the experiment described in Chapter 5 are provided here in tabular form. The values appearing in the tables are the average of 40 simulation runs. The variables that appear in the tables are defined as follows:

NumIter: number of iterations at termination.

NumEval: cumulative number of objective function evaluations.

StepSize: size of simplex for simplex methods, or distance between two most recent points in Kiefer-Wolfowitz method.

Error: expected function value at point of termination minus minimum expected function value. The point of termination is the centroid of the final simplex, or the last point in the Kiefer-Wolfowitz method.

PerGap: percentage gap remaining in expected function value. Pergap = 100 $\times$ Error / Initial Gap, where the initial gap is the expected function value at the starting point minus the minimum expected function value.

Note that methods NM and RS9 terminated as a result of falling below the minimum simplex size of 10^{-10}, method NMSN terminated as a result of exceeding 50,000 cumulative function evaluations and method KW terminated as a result of reaching a maximum of 10,000 iterations.

A hyphen appears in entries for method KW in Table B.4. When a hyphen appears under StepSize, our implementation of the Kiefer-Wolfowitz procedure took so large a step that the computer output read "Inf," resulting in overflow conditions for the variables Error and PerGap.

Table B.1: Output at Termination for Univariate Function 1

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.358E+02	0.736E+02	0.582E-10	0.764E+00	0.753E+02
RS9	0.747E+03	0.203E+04	0.943E-10	0.453E+00	0.455E+02
NMSN	0.380E+02	0.557E+05	0.234E-01	0.111E-01	0.109E+01
KW	0.100E+05	0.200E+05	0.101E-02	0.233E-01	0.230E+01

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.386E+02	0.792E+02	0.582E-10	0.204E+01	0.204E+02
RS9	0.759E+03	0.207E+04	0.952E-10	0.461E+00	0.462E+01
NMSN	0.460E+02	0.554E+05	0.331E-01	0.725E-02	0.724E-01
KW	0.100E+05	0.200E+05	0.115E-02	0.231E-01	0.230E+00

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.433E+02	0.886E+02	0.582E-10	0.132E+02	0.132E+02
RS9	0.778E+03	0.212E+04	0.947E-10	0.556E+00	0.556E+00
NMSN	0.556E+02	0.548E+05	0.871E-01	0.779E-02	0.779E-02
KW	0.100E+05	0.200E+05	0.112E-02	0.619E+02	0.619E+02

Table B.2: Output at Termination for Univariate Function 2

		$GAP/\sigma = 1.0 \pm 0.1$			
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.360E+02	0.739E+02	0.582E-10	0.944E+00	0.945E+02
RS9	0.754E+03	0.205E+04	0.950E-10	0.495E+00	0.494E+02
NMSN	0.394E+02	0.547E+05	0.211E-01	0.101E-01	0.999E+00
KW	0.100E+05	0.200E+05	0.131E-02	0.207E-01	0.204E+01
		$GAP/\sigma = 10.0 \pm 0.1$			
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.382E+02	0.785E+02	0.582E-10	0.222E+01	0.222E+02
RS9	0.753E+03	0.205E+04	0.948E-10	0.560E+00	0.560E+01
NMSN	0.431E+02	0.560E+05	0.317E-01	0.526E-02	0.526E-01
KW	0.100E+05	0.200E+05	0.134E-02	0.211E-01	0.210E+00
		$GAP/\sigma = 100.0 \pm 0.1$			
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.433E+02	0.886E+02	0.582E-10	0.156E+02	0.156E+02
RS9	0.793E+03	0.216E+04	0.949E-10	0.665E+00	0.665E+00
NMSN	0.592E+02	0.549E+05	0.642E-01	0.615E-02	0.615E-02
KW	0.100E+05	0.200E+05	0.118E-02	0.666E+02	0.666E+02

Table B.3: Output at Termination for Univariate Function 3

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.370E+02	0.761E+02	0.582E-10	0.430E+00	0.421E+02
RS9	0.783E+03	0.213E+04	0.947E-10	0.281E+00	0.285E+02
NMSN	0.364E+02	0.552E+05	0.943E-01	0.629E-02	0.643E+00
KW	0.100E+05	0.200E+05	0.113E-02	0.772E-02	0.767E+00

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.387E+02	0.793E+02	0.582E-10	0.323E+00	0.323E+01
RS9	0.795E+03	0.216E+04	0.949E-10	0.250E+00	0.250E+01
NMSN	0.394E+02	0.552E+05	0.109E+00	0.334E-02	0.335E-01
KW	0.100E+05	0.200E+05	0.127E-02	0.771E-02	0.770E-01

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.414E+02	0.849E+02	0.582E-10	0.329E+00	0.329E+00
RS9	0.796E+03	0.217E+04	0.937E-10	0.217E+00	0.217E+00
NMSN	0.521E+02	0.542E+05	0.113E+00	0.365E-02	0.365E-02
KW	0.100E+05	0.200E+05	0.118E-02	0.773E-02	0.773E-02

Table B.4: Output at Termination for Univariate Function 4

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.371E+02	0.763E+02	0.582E-10	0.627E+00	0.614E+02
RS9	0.807E+03	0.219E+04	0.948E-10	0.254E+00	0.253E+02
NMSN	0.355E+02	0.547E+05	0.248E+00	0.797E-02	0.795E+00
KW	0.100E+05	0.200E+05	0.119E-02	0.250E-01	0.247E+01

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.391E+02	0.802E+02	0.582E-10	0.176E+00	0.176E+01
RS9	0.805E+03	0.219E+04	0.953E-10	0.225E+00	0.225E+01
NMSN	0.384E+02	0.554E+05	0.205E+00	0.862E-02	0.862E-01
KW	0.100E+05	0.200E+05	–	–	–

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.400E+02	0.820E+02	0.582E-10	0.296E+00	0.296E+00
RS9	0.816E+03	0.222E+04	0.947E-10	0.233E+00	0.233E+00
NMSN	0.384E+02	0.549E+05	0.242E+00	0.724E-02	0.724E-02
KW	0.100E+05	0.200E+05	–	–	–

Table B.5: Output at Termination for Univariate Function 5

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.364E+02	0.748E+02	0.582E-10	0.205E+01	0.203E+03
RS9	0.719E+03	0.196E+04	0.946E-10	0.506E+00	0.494E+02
NMSN	0.458E+02	0.551E+05	0.111E-01	0.147E-02	0.150E+00
KW	0.100E+05	0.200E+05	0.127E-02	0.522E-02	0.522E+00

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.379E+02	0.778E+02	0.582E-10	0.582E+01	0.581E+02
RS9	0.716E+03	0.195E+04	0.947E-10	0.279E+00	0.279E+01
NMSN	0.568E+02	0.555E+05	0.131E-01	0.258E-02	0.259E-01
KW	0.100E+05	0.200E+05	0.117E-02	0.268E+00	0.267E+01

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.442E+02	0.904E+02	0.582E-10	0.273E+01	0.273E+01
RS9	0.734E+03	0.200E+04	0.940E-10	0.487E+00	0.487E+00
NMSN	0.714E+02	0.550E+05	0.149E-01	0.160E-02	0.160E-02
KW	0.100E+05	0.200E+05	0.110E-02	0.679E+01	0.679E+01

Table B.6: Output at Termination for Univariate Function 6

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.360E+02	0.739E+02	0.582E-10	0.373E+00	0.373E+02
RS9	0.731E+03	0.199E+04	0.948E-10	0.318E+00	0.320E+02
NMSN	0.394E+02	0.547E+05	0.213E-01	0.326E-02	0.323E+00
KW	0.100E+05	0.200E+05	0.118E-02	0.147E+02	0.146E+04

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.369E+02	0.758E+02	0.582E-10	0.461E+00	0.460E+01
RS9	0.722E+03	0.197E+04	0.953E-10	0.290E+00	0.291E+01
NMSN	0.410E+02	0.556E+05	0.241E-01	0.297E-02	0.296E-01
KW	0.100E+05	0.200E+05	0.122E-02	0.629E+01	0.632E+02

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.376E+02	0.772E+02	0.582E-10	0.631E+01	0.332E+02
RS9	0.716E+03	0.195E+04	0.951E-10	0.491E-01	0.258E+00
NMSN	0.538E+02	0.555E+05	0.107E-01	0.652E-03	0.343E-02
KW	0.100E+05	0.200E+05	0.226E-03	0.164E-03	0.862E-03

Table B.7: Output at Termination for Univariate Function 7

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.367E+02	0.754E+02	0.582E-10	0.558E+00	0.549E+02
RS9	0.781E+03	0.212E+04	0.947E-10	0.203E+00	0.203E+02
NMSN	0.369E+02	0.554E+05	0.108E+00	0.369E-02	0.367E+00
KW	0.100E+05	0.200E+05	0.121E-02	0.852E-02	0.839E+00

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.388E+02	0.795E+02	0.582E-10	0.238E+00	0.238E+01
RS9	0.791E+03	0.215E+04	0.944E-10	0.249E+00	0.249E+01
NMSN	0.390E+02	0.550E+05	0.122E+00	0.196E-02	0.196E-01
KW	0.100E+05	0.200E+05	0.113E-02	0.847E-02	0.845E-01

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.408E+02	0.835E+02	0.582E-10	0.293E+00	0.293E+00
RS9	0.798E+03	0.217E+04	0.948E-10	0.222E+00	0.222E+00
NMSN	0.451E+02	0.554E+05	0.130E+00	0.279E-02	0.279E-02
KW	0.100E+05	0.200E+05	0.126E-02	0.258E+01	0.258E+01

Table B.8: Output at Termination for Univariate Function 8

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.361E+02	0.741E+02	0.582E-10	0.491E+00	0.491E+02
RS9	0.771E+03	0.209E+04	0.955E-10	0.349E+00	0.344E+02
NMSN	0.358E+02	0.552E+05	0.112E+00	0.317E-02	0.308E+00
KW	0.100E+05	0.200E+05	0.107E-02	0.861E-02	0.845E+00

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.400E+02	0.821E+02	0.582E-10	0.742E+00	0.738E+01
RS9	0.809E+03	0.220E+04	0.944E-10	0.202E+00	0.202E+01
NMSN	0.395E+02	0.555E+05	0.139E+00	0.255E-02	0.255E-01
KW	0.100E+05	0.200E+05	0.132E-02	0.985E-02	0.984E-01

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.449E+02	0.917E+02	0.582E-10	0.279E+01	0.278E+01
RS9	0.822E+03	0.223E+04	0.944E-10	0.258E+00	0.258E+00
NMSN	0.505E+02	0.557E+05	0.171E+00	0.264E-02	0.264E-02
KW	0.100E+05	0.200E+05	0.101E-02	0.360E+02	0.360E+02

Table B.9: Output at Termination for Univariate Function 9

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.367E+02	0.755E+02	0.582E-10	0.403E+00	0.399E+02
RS9	0.791E+03	0.215E+04	0.958E-10	0.218E+00	0.217E+02
NMSN	0.366E+02	0.543E+05	0.183E+00	0.318E-02	0.319E+00
KW	0.100E+05	0.200E+05	0.114E-02	0.154E-01	0.153E+01

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.387E+02	0.795E+02	0.582E-10	0.295E+00	0.294E+01
RS9	0.804E+03	0.218E+04	0.951E-10	0.260E+00	0.259E+01
NMSN	0.373E+02	0.548E+05	0.243E+00	0.292E-02	0.292E-01
KW	0.100E+05	0.200E+05	0.112E-02	0.191E+01	0.191E+02

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.394E+02	0.808E+02	0.582E-10	0.239E+00	0.239E+00
RS9	0.810E+03	0.220E+04	0.949E-10	0.188E+00	0.188E+00
NMSN	0.389E+02	0.550E+05	0.196E+00	0.409E-02	0.409E-02
KW	0.100E+05	0.200E+05	0.109E-02	0.327E+03	0.327E+03

Table B.10: Output at Termination for Univariate Function 10

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.361E+02	0.742E+02	0.582E-10	0.102E+01	0.101E+03
RS9	0.803E+03	0.218E+04	0.957E-10	0.234E+00	0.232E+02
NMSN	0.356E+02	0.551E+05	0.184E+00	0.324E-02	0.320E+00
KW	0.100E+05	0.200E+05	0.126E-02	0.540E-01	0.542E+01

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.397E+02	0.814E+02	0.582E-10	0.292E+01	0.291E+02
RS9	0.816E+03	0.222E+04	0.941E-10	0.261E+00	0.261E+01
NMSN	0.407E+02	0.548E+05	0.203E+00	0.413E-02	0.412E-01
KW	0.100E+05	0.200E+05	0.120E-02	0.438E+00	0.437E+01

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.445E+02	0.909E+02	0.582E-10	0.278E+00	0.278E+00
RS9	0.832E+03	0.226E+04	0.950E-10	0.169E+00	0.169E+00
NMSN	0.464E+02	0.540E+05	0.270E+00	0.283E-02	0.283E-02
KW	0.100E+05	0.200E+05	0.128E-02	0.428E+01	0.427E+01

Table B.11: Output at Termination for Univariate Function 11

$GAP/\sigma = 1.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.369E+02	0.758E+02	0.582E-10	0.345E+00	0.338E+02
RS9	0.756E+03	0.205E+04	0.953E-10	0.229E+00	0.231E+02
NMSN	0.376E+02	0.548E+05	0.684E-01	0.805E-02	0.812E+00
KW	0.100E+05	0.200E+05	0.108E-02	0.827E-02	0.822E+00

$GAP/\sigma = 10.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.367E+02	0.754E+02	0.582E-10	0.133E+00	0.663E+01
RS9	0.730E+03	0.198E+04	0.953E-10	0.653E-01	0.327E+01
NMSN	0.411E+02	0.551E+05	0.363E-01	0.821E-03	0.409E-01
KW	0.100E+05	0.200E+05	0.239E-03	0.718E-03	0.359E-01

$GAP/\sigma = 100.0 \pm 0.1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.386E+02	0.792E+02	0.582E-10	0.213E+01	0.236E+02
RS9	0.725E+03	0.197E+04	0.950E-10	0.230E-01	0.256E+00
NMSN	0.505E+02	0.546E+05	0.215E-01	0.449E-03	0.498E-02
KW	0.100E+05	0.200E+05	0.117E-03	0.558E-03	0.620E-02

Table B.12: Output at Termination for Univariate Function 12

$GAP/\sigma = 1.0 \pm 0.1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.358E+02	0.736E+02	0.582E-10	0.847E+00	0.843E+02
RS9	0.761E+03	0.207E+04	0.952E-10	0.251E+00	0.251E+02
NMSN	0.367E+02	0.548E+05	0.664E-01	0.652E-02	0.641E+00
KW	0.100E+05	0.200E+05	0.122E-02	0.850E-02	0.853E+00

$GAP/\sigma = 10.0 \pm 0.1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.388E+02	0.797E+02	0.582E-10	0.393E+00	0.393E+01
RS9	0.762E+03	0.207E+04	0.956E-10	0.253E+00	0.253E+01
NMSN	0.407E+02	0.553E+05	0.803E-01	0.478E-02	0.477E-01
KW	0.100E+05	0.200E+05	0.130E-02	0.853E-02	0.852E-01

$GAP/\sigma = 100.0 \pm 0.1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.416E+02	0.852E+02	0.582E-10	0.316E+00	0.316E+00
RS9	0.790E+03	0.215E+04	0.945E-10	0.260E+00	0.260E+00
NMSN	0.521E+02	0.542E+05	0.792E-01	0.413E-02	0.413E-02
KW	0.100E+05	0.200E+05	0.122E-02	0.103E-01	0.103E-01

Appendix C

Graphs of Multivariate Test Results

The complete set of graphs of the results from the multivariate experiment described in Chapter 7 are presented here for further reference. The graphs illustrate the average level of percent gap remaining between the expected function value at the which are zero) versus the average cost in terms of the number of function evaluations required for each of the four algorithms tested. Each test function and starting point combination are plotted twice. The graphs on the left side of the page use a linear scale for *PERGAP* while those on the right use a logarithmic scale for *PERGAP*. The linear scale is more illustrative for practical considerations; whereas, the logarithmic scale is more illustrative for theoretical considerations. Note the logarithmic scaling of the horizontal axis and that the lines for methods NM and RS9 typically stop well short of the lines for methods KW and NMSN. This is a result of the different stopping criteria used in the experiment. In general, algorithms NM and RS9 stopped as a result of the simplex size falling below 10^{-10}, and algorithms KW, NMSNR and NMSNV stopped as a result of exceeding $10,000(n+1)$ total function evaluations, where n is the dimension of the parameter space.

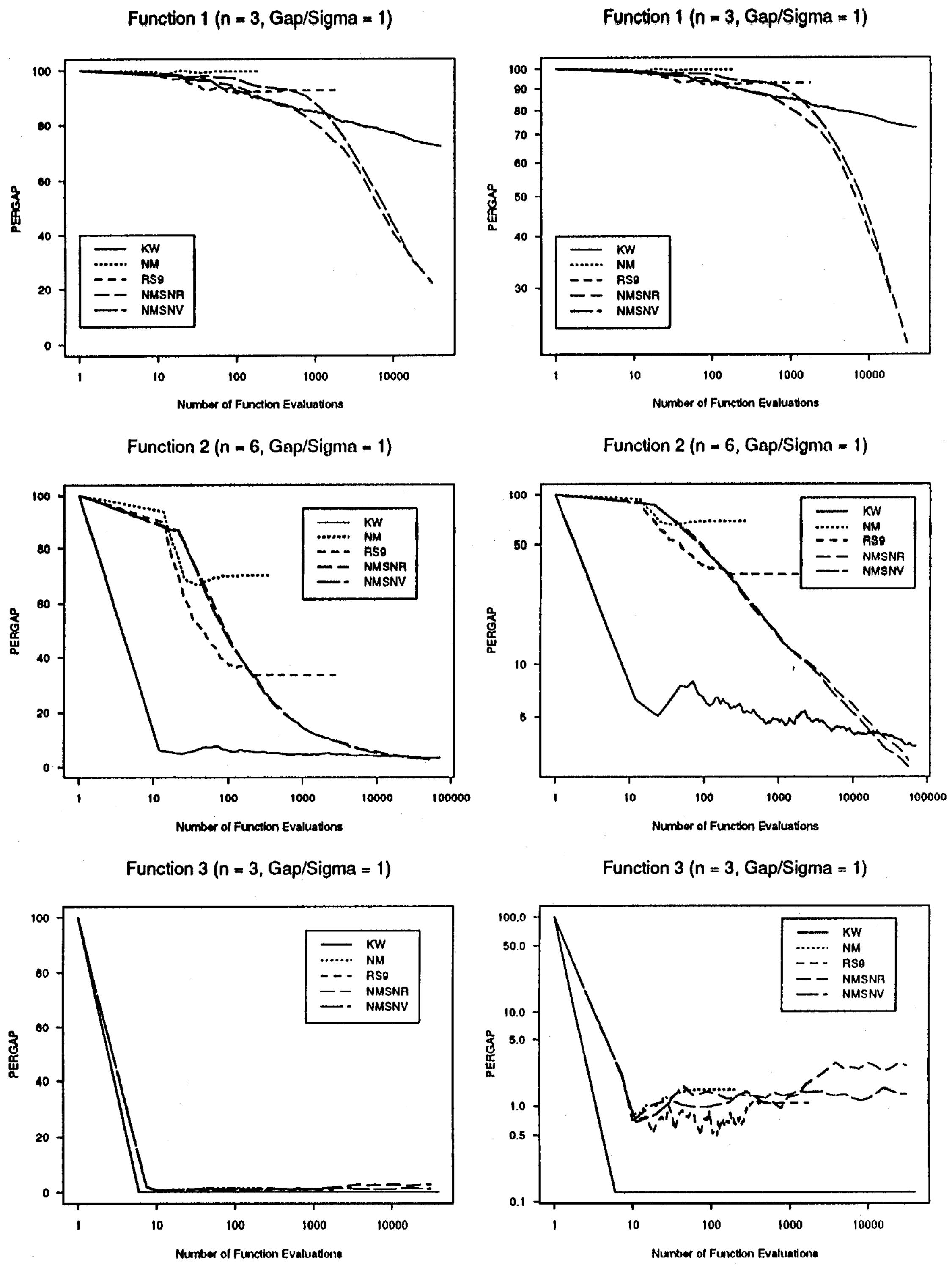


Figure C.1: Test Results for Multivariate Functions 1-3, $GAP/\sigma \approx 1$

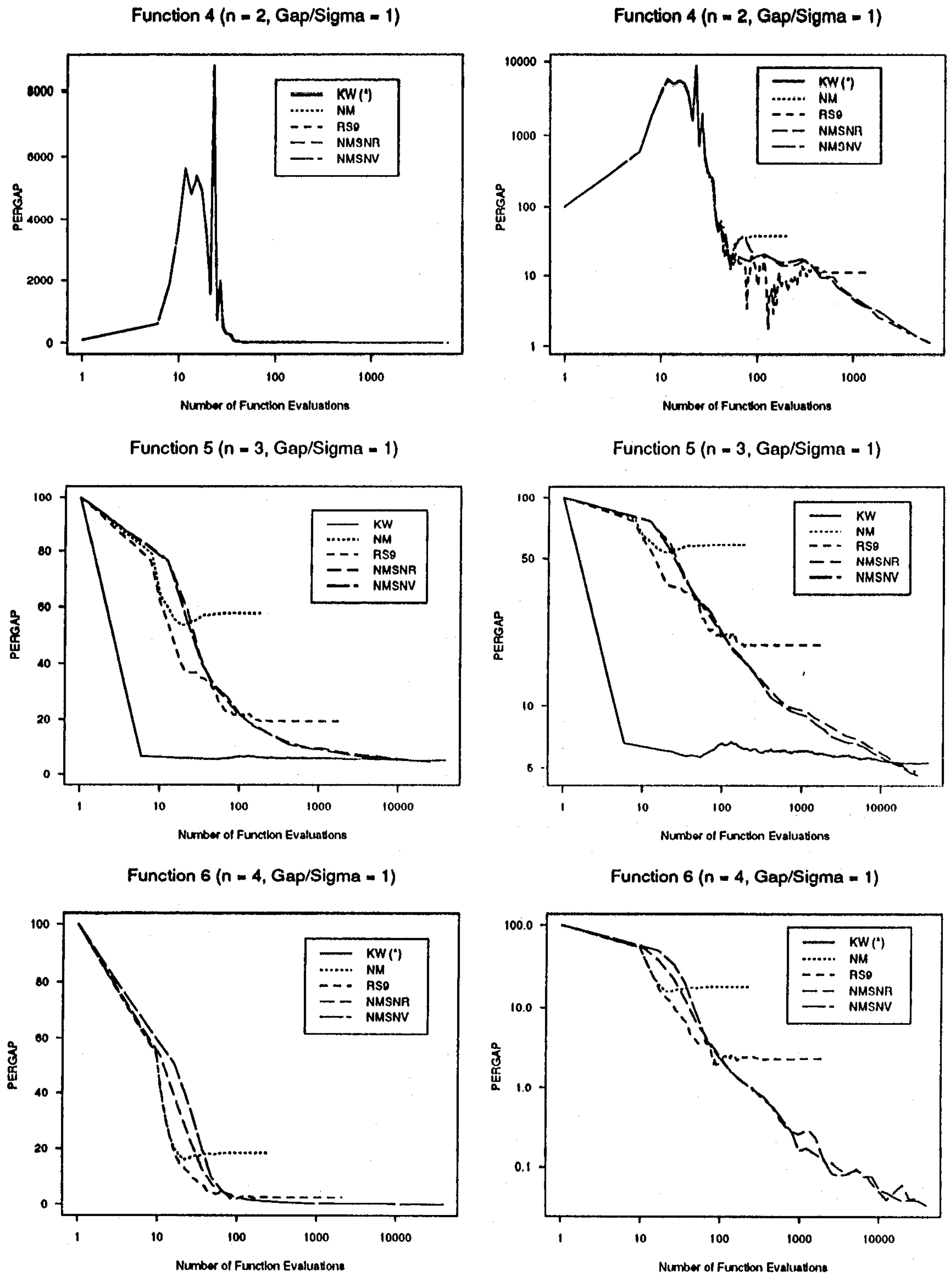


Figure C.2: Test Results for Multivariate Functions 4-6, $GAP/\sigma \approx 1$

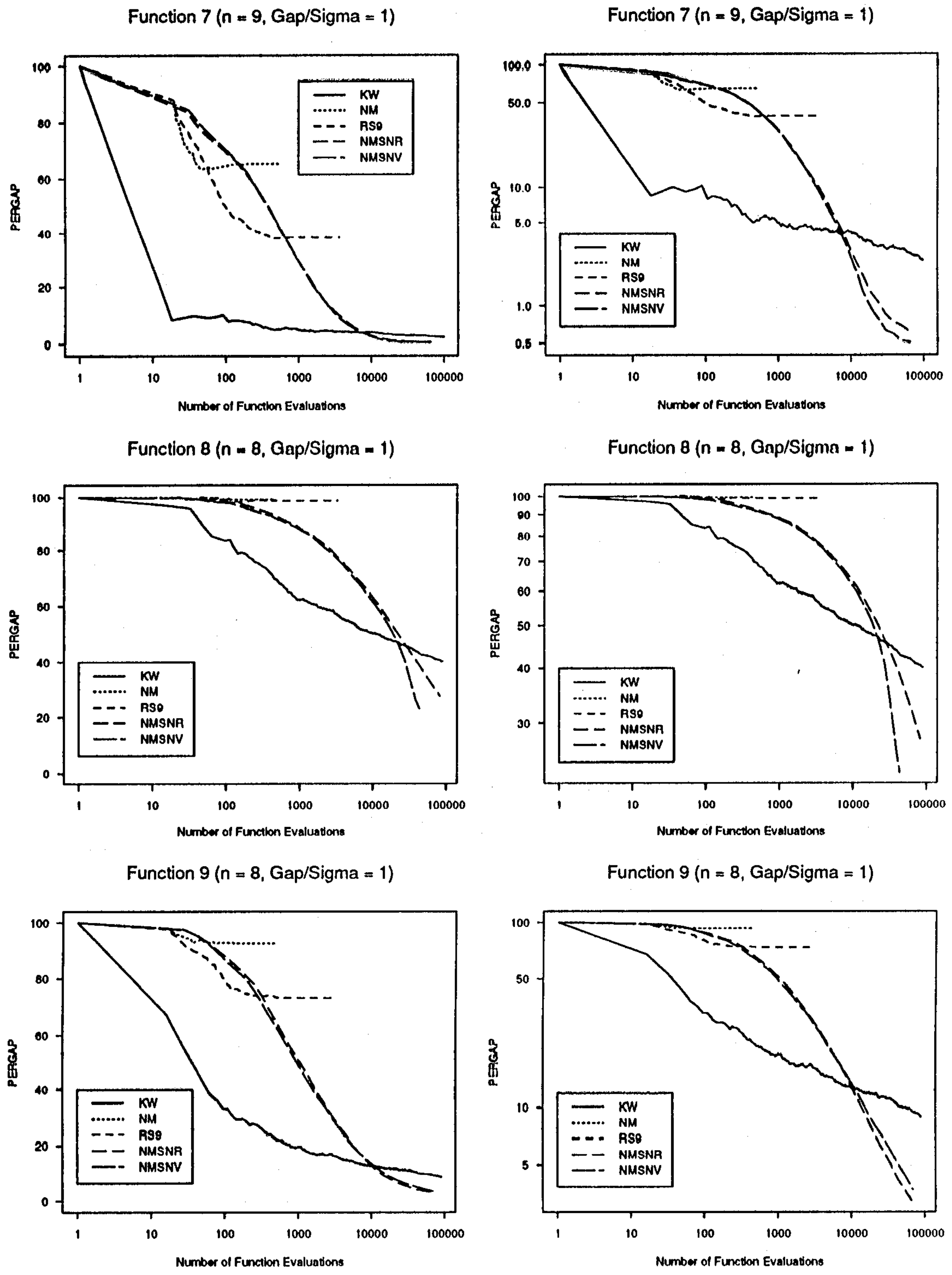


Figure C.3: Test Results for Multivariate Functions 7-9, $GAP/\sigma \approx 1$

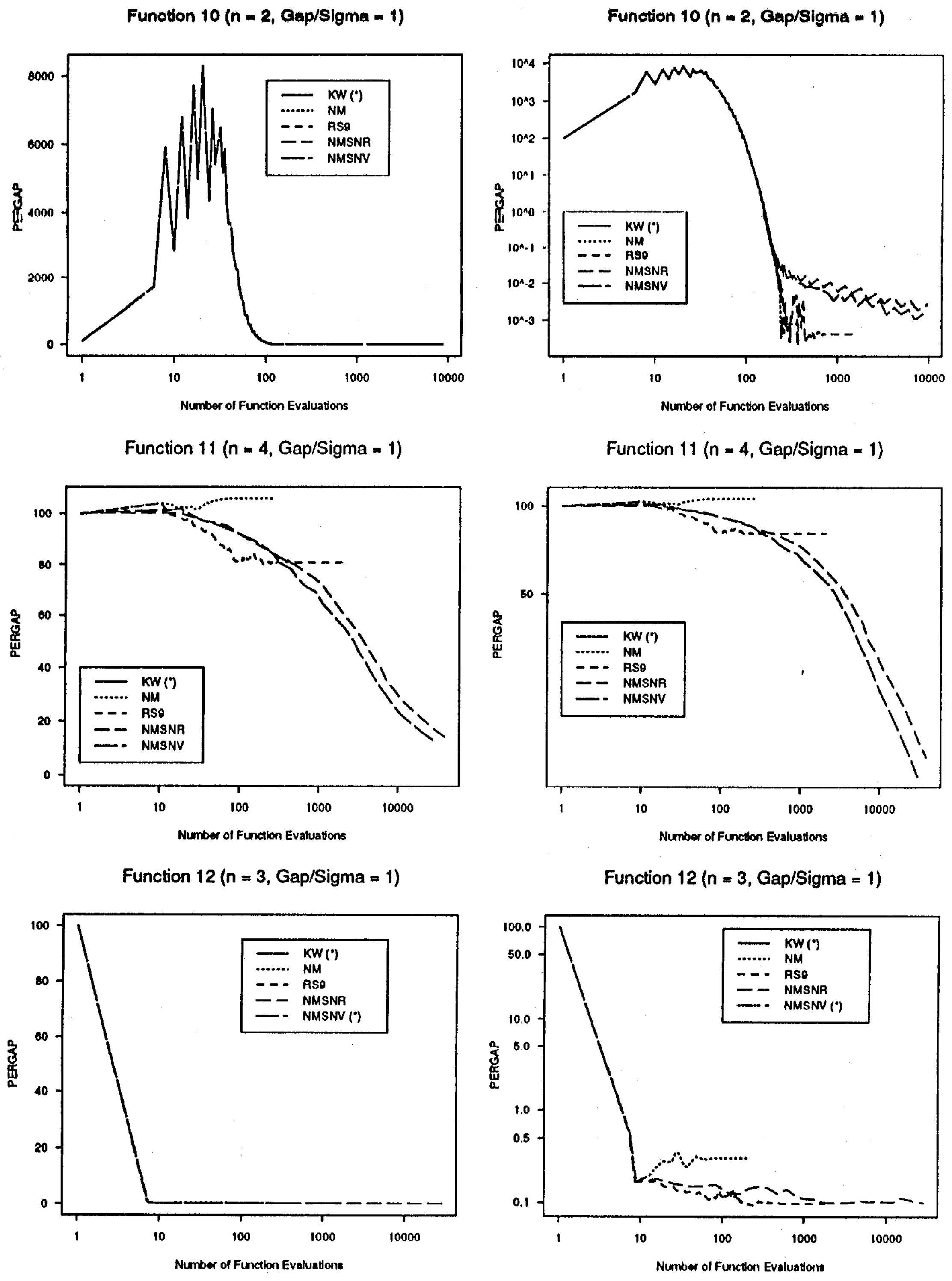


Figure C.4: Test Results for Multivariate Functions 10-12, $GAP/\sigma \approx 1$

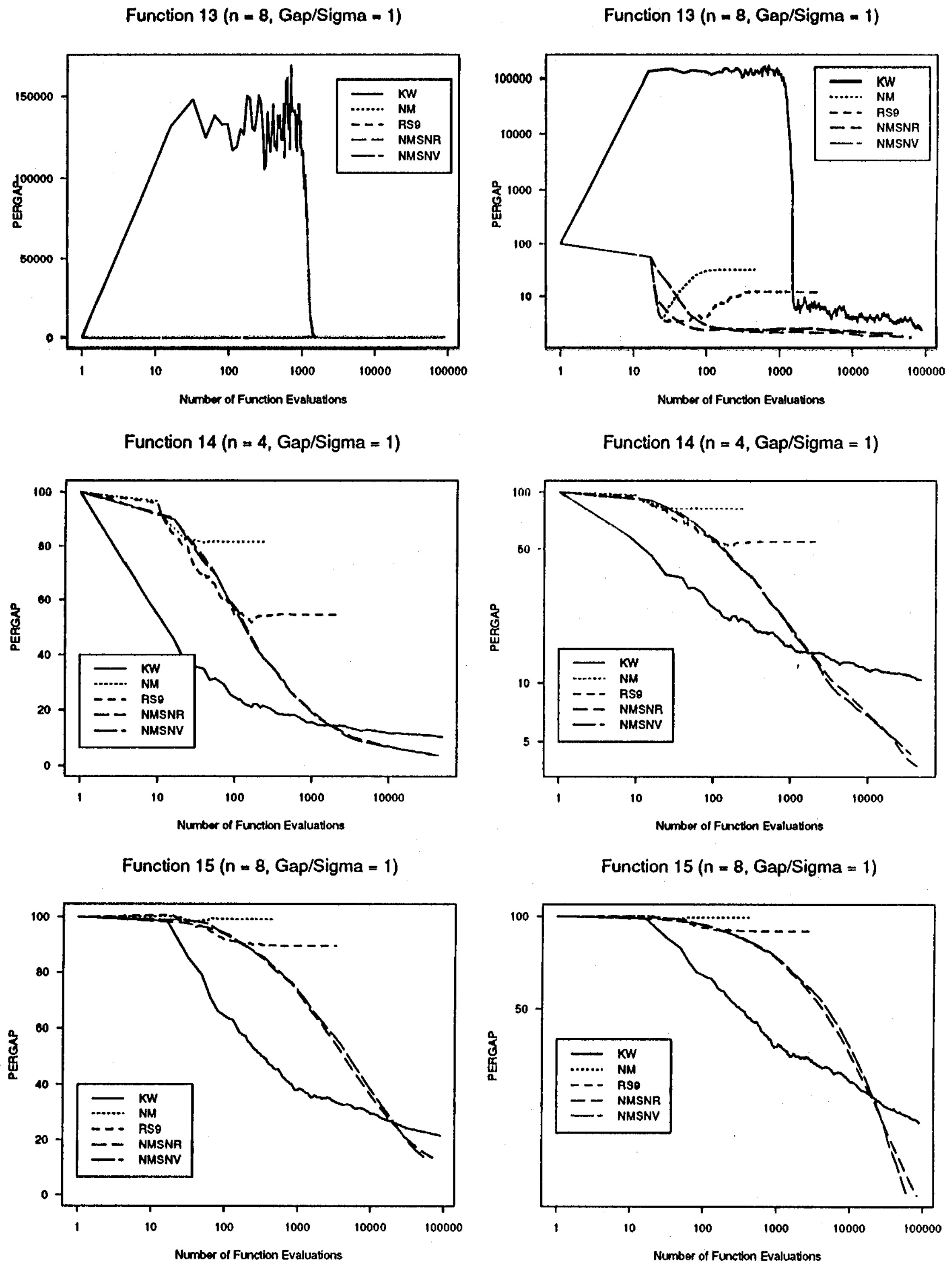


Figure C.5: Test Results for Multivariate Function 13-15, $GAP/\sigma \approx 1$

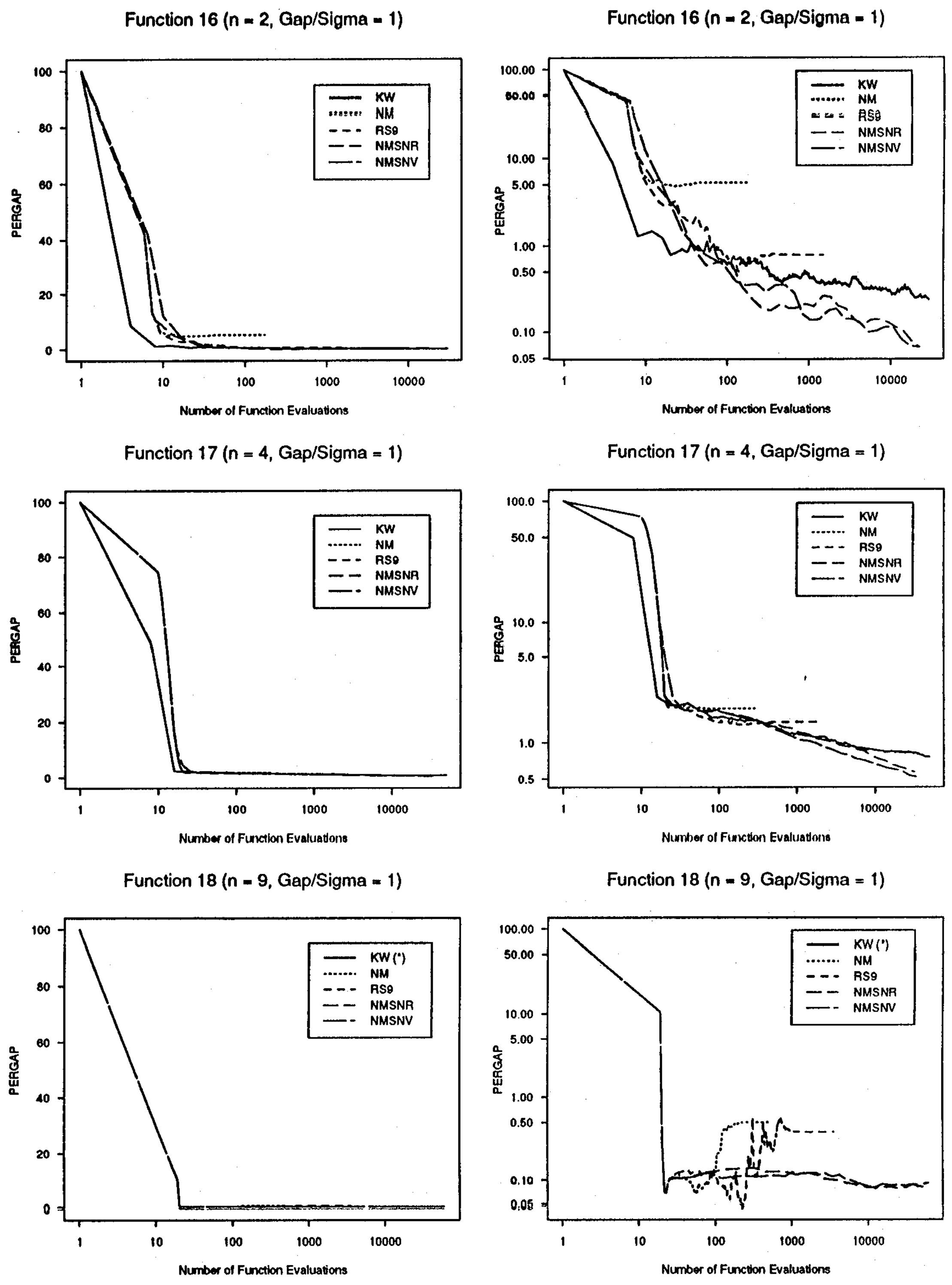


Figure C.6: Test Results for Multivariate Function 16-18, $GAP/\sigma \approx 1$

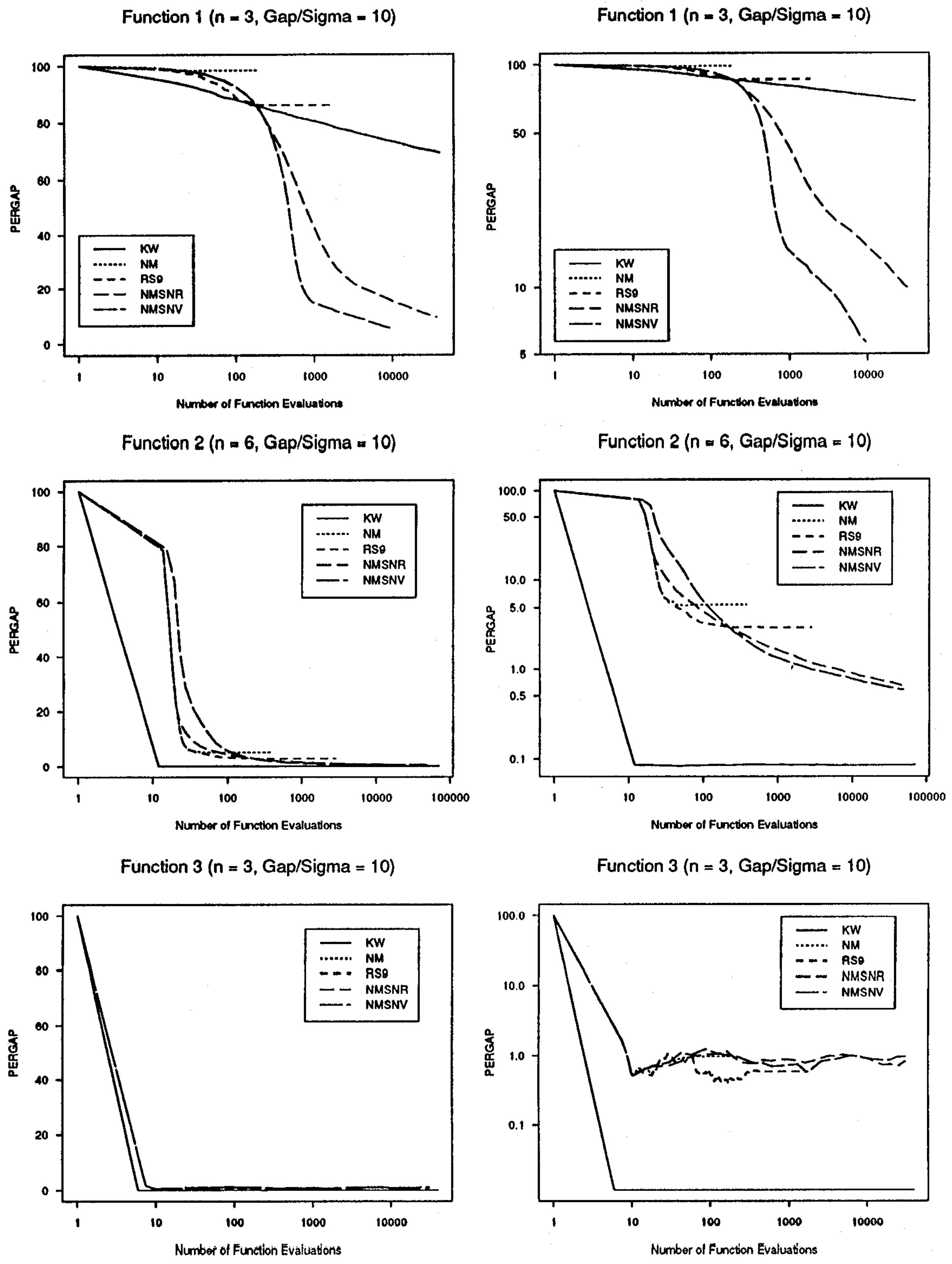


Figure C.7: Test Results for Multivariate Functions 1-3, $GAP/\sigma \approx 10$

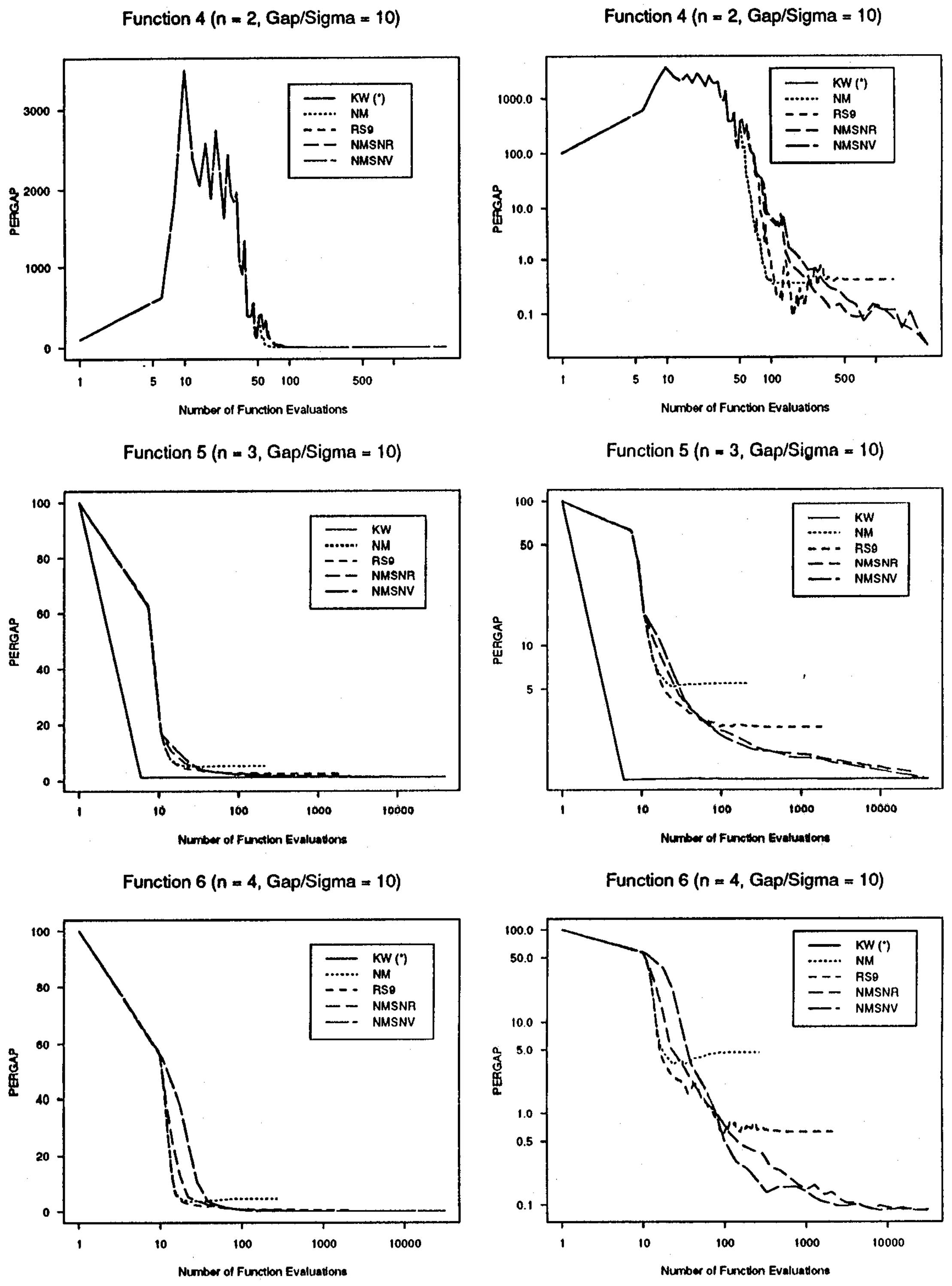


Figure C.8: Test Results for Multivariate Functions 4-6, $GAP/\sigma \approx 10$

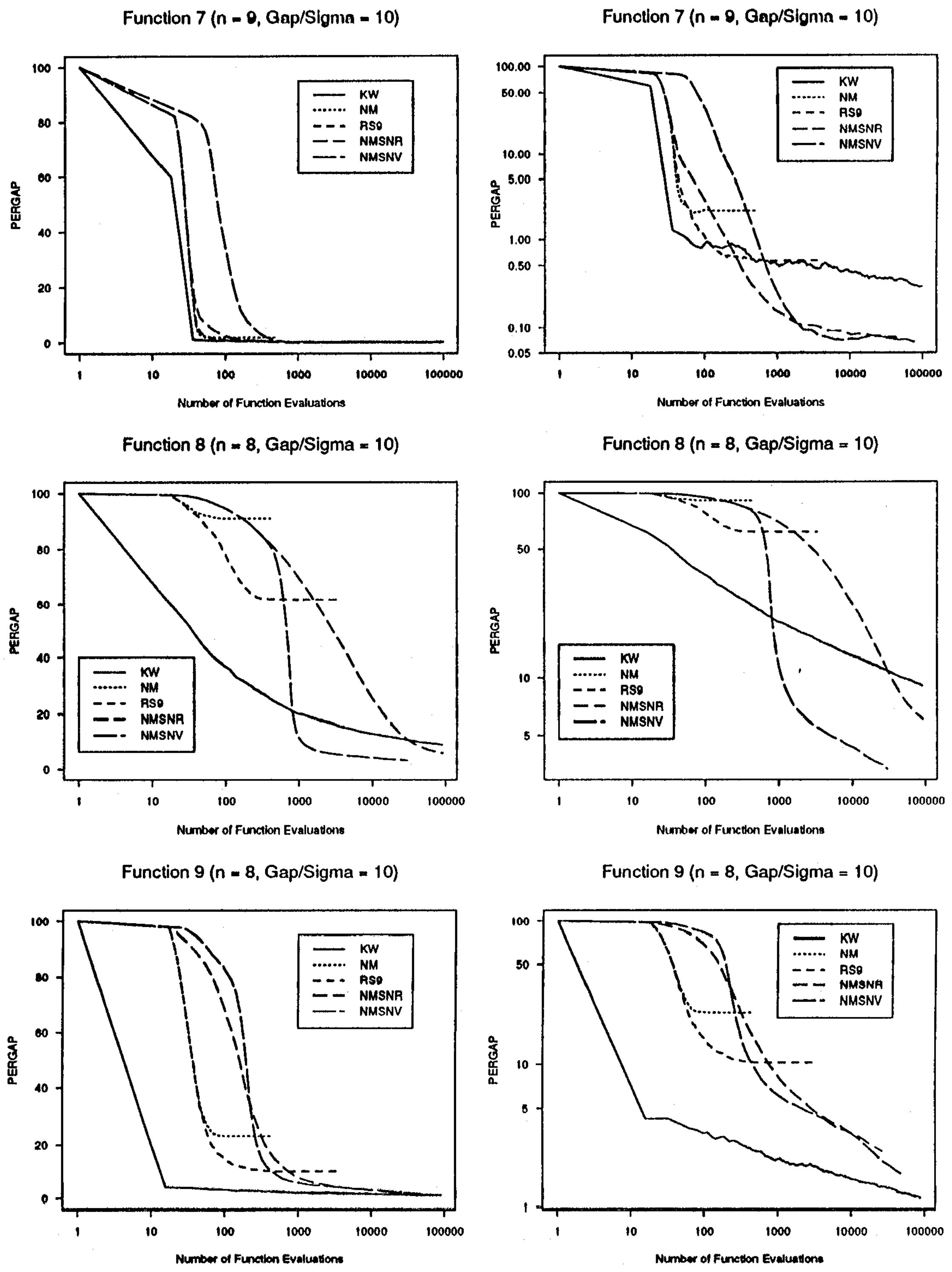


Figure C.9: Test Results for Multivariate Functions 7-9, $GAP/\sigma \approx 10$

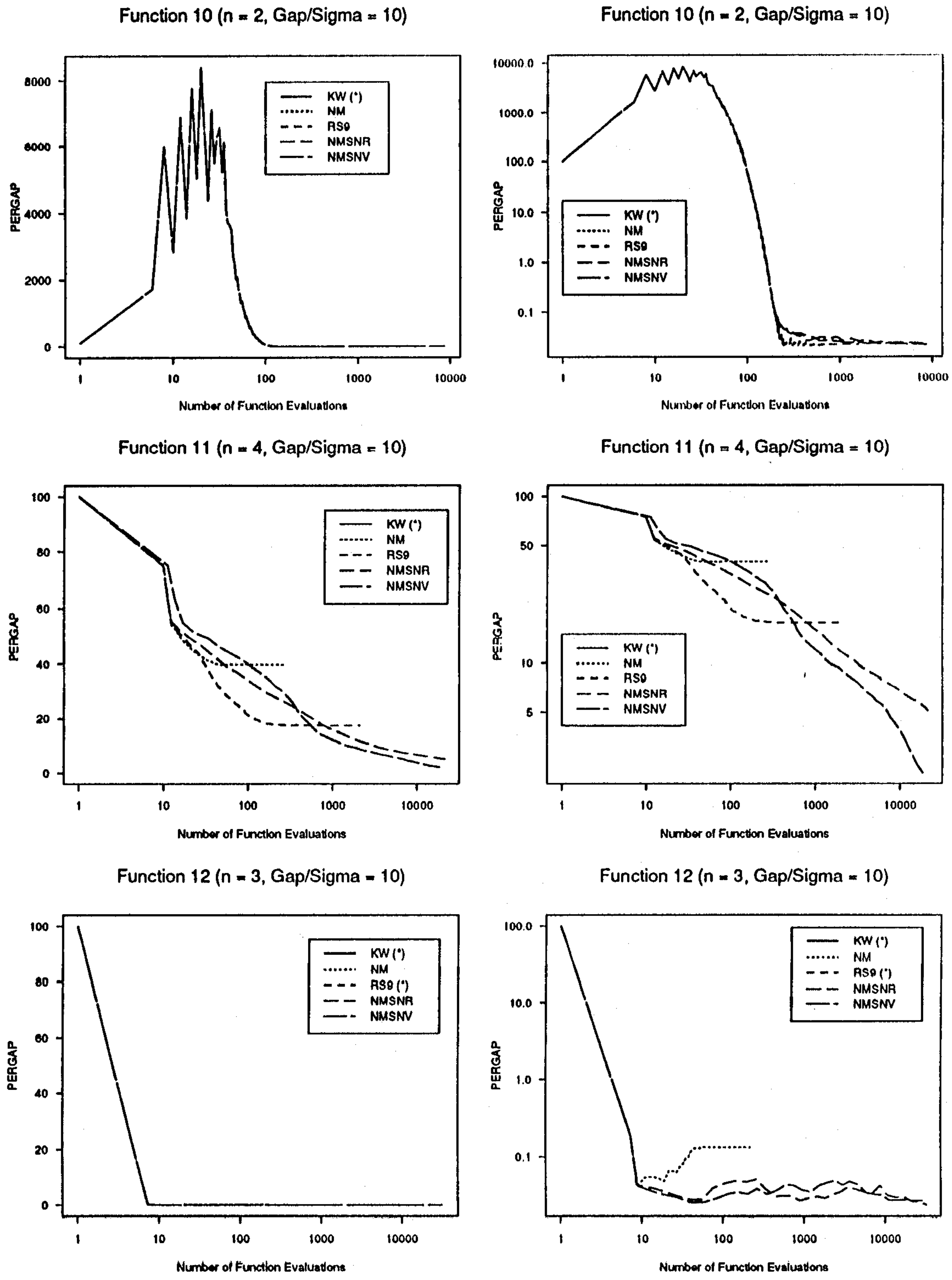


Figure C.10: Test Results for Multivariate Functions 10-12, $GAP/\sigma \approx 10$

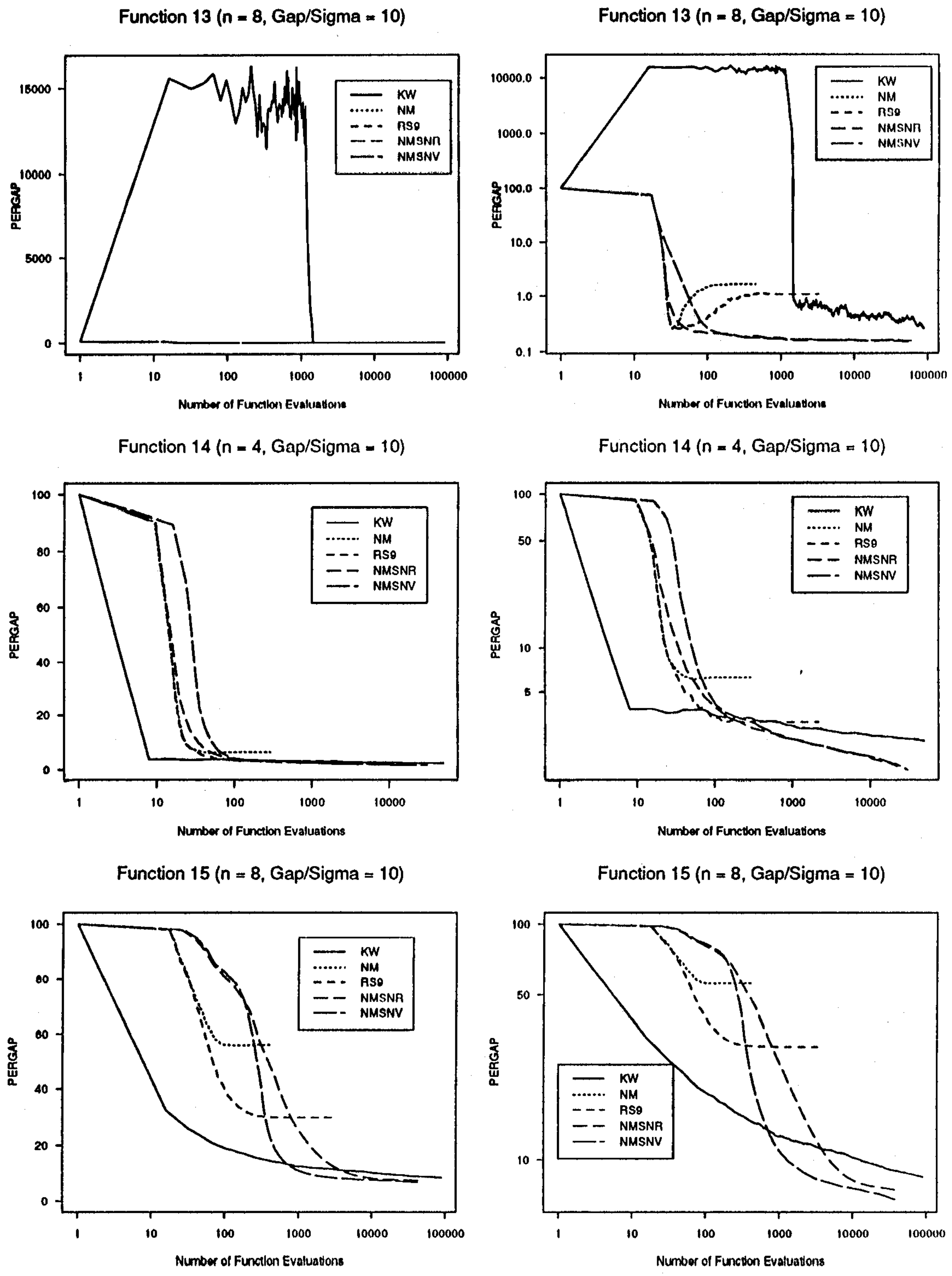


Figure C.11: Test Results for Multivariate Function 13-15, $GAP/\sigma \approx 10$

Appendix D

Tabular Output of Multivariate Testing

The average value of several variables at termination for each design point in the experiment described in Chapter 7 are provided here in tabular form. The values appearing in the tables are the average of 40 simulation runs. The variables that appear in the tables are defined as follows:

NumIter: number of iterations at termination.

NumEval: cumulative number of objective function evaluations.

StepSize: size of simplex for simplex methods, or distance between two most recent points in Kiefer-Wolfowitz method.

Error: expected function value at point of termination minus minimum expected function value. The point of termination is the centroid of the final simplex, or the last point in the Kiefer-Wolfowitz method.

PerGap: percentage gap remaining in expected function value. Pergap $= 100 \times$ Error / Initial Gap, where the initial gap is the expected function value at the starting point minus the minimum expected function value.

Note that methods NM and RS9 terminated as a result of falling below the minimum simplex size of 10^{-10}, methods NMSNR, NMSNV and KW terminated as a result of exceeding $10,000(n+1)$ cumulative function evaluations (where n is the dimension of the parameter space).

A hyphen appears in entries for method KW in Tables D.4, D.6, D.10, D.11, D.12, D.16, and D.18. When a hyphen appears under StepSize, our implementation of the Kiefer-Wolfowitz procedure took so large a step that the computer output read "Inf," resulting in overflow conditions for the variables Error and PerGap. In some cases the function is steep enough to have caused overflow conditions for Error and PerGap without necessarily taking a very large step.

Table D.1: Output at Termination for Multivariate Function 1.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.850E+02	0.232E+03	0.770E-10	0.102E+01	0.996E+02
RS9	0.718E+03	0.204E+04	0.953E-10	0.965E+00	0.945E+02
NMSNR	0.317E+02	0.463E+05	0.340E+01	0.163E+00	0.160E+02
NMSNV	0.350E+02	0.451E+05	0.309E+01	0.133E+00	0.130E+02
KW	0.667E+04	0.400E+05	0.313E-02	0.739E+00	0.724E+02

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.846E+02	0.231E+03	0.742E-10	0.985E+01	0.985E+02
RS9	0.718E+03	0.204E+04	0.943E-10	0.893E+01	0.893E+02
NMSNR	0.320E+02	0.453E+05	0.965E+01	0.852E+00	0.852E+01
NMSNV	0.500E+02	0.440E+05	0.467E+01	0.114E+00	0.114E+01
KW	0.667E+04	0.400E+05	0.312E-02	0.693E+01	0.693E+02

Table D.2: Output at Termination for Multivariate Function 2.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.156E+03	0.441E+03	0.761E-10	0.666E+00	0.634E+02
RS9	0.104E+04	0.303E+04	0.950E-10	0.342E+00	0.325E+02
NMSNR	0.335E+02	0.766E+05	0.164E+01	0.255E-01	0.245E+01
NMSNV	0.336E+02	0.801E+05	0.168E+01	0.233E-01	0.223E+01
KW	0.583E+04	0.700E+05	0.534E-02	0.353E-01	0.341E+01

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.168E+03	0.463E+03	0.711E-10	0.600E+00	0.599E+01
RS9	0.106E+04	0.307E+04	0.949E-10	0.302E+00	0.302E+01
NMSNR	0.414E+02	0.794E+05	0.238E+01	0.582E-01	0.583E+00
NMSNV	0.385E+02	0.799E+05	0.305E+01	0.502E-01	0.499E+00
KW	0.583E+04	0.700E+05	0.534E-02	0.843E-02	0.844E-01

Table D.3: Output at Termination for Multivariate Function 3.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.982E+02	0.265E+03	0.741E-10	0.456E-03	0.115E+01
RS9	0.718E+03	0.204E+04	0.951E-10	0.514E-03	0.114E+01
NMSNR	0.338E+02	0.489E+05	0.597E+00	0.764E-03	0.298E+01
NMSNV	0.336E+02	0.487E+05	0.543E+00	0.672E-03	0.133E+01
KW	0.667E+04	0.400E+05	0.249E-02	0.564E-04	0.127E+00

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.987E+02	0.266E+03	0.740E-10	0.334E-02	0.739E+00
RS9	0.720E+03	0.205E+04	0.956E-10	0.395E-02	0.969E+00
NMSNR	0.337E+02	0.500E+05	0.520E+00	0.339E-02	0.727E+00
NMSNV	0.338E+02	0.506E+05	0.526E+00	0.366E-02	0.872E+00
KW	0.667E+04	0.400E+05	0.249E-02	0.564E-04	0.123E-01

Table D.4: Output at Termination for Multivariate Function 4.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.966E+02	0.242E+03	0.713E-10	0.155E+00	0.360E+02
RS9	0.601E+03	0.169E+04	0.951E-10	0.711E-01	0.110E+02
NMSNR	0.592E+02	0.342E+05	0.807E-01	0.142E-02	0.174E+00
NMSNV	0.597E+02	0.347E+05	0.907E-01	0.204E-02	0.473E-01
KW	0.750E+04	0.300E+05	–	–	–

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.107E+03	0.263E+03	0.717E-10	0.111E+00	0.335E+00
RS9	0.623E+03	0.174E+04	0.949E-10	0.617E-01	0.409E+00
NMSNR	0.735E+02	0.335E+05	0.845E-01	0.445E-02	0.850E-02
NMSNV	0.737E+02	0.332E+05	0.819E-01	0.163E-02	0.290E-02
KW	0.750E+04	0.300E+05	–	–	–

Table D.5: Output at Termination for Multivariate Function 5.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.964E+02	0.263E+03	0.693E-10	0.560E+00	0.531E+02
RS9	0.724E+03	0.206E+04	0.950E-10	0.204E+00	0.194E+02
NMSNR	0.311E+02	0.469E+05	0.102E+01	0.474E-01	0.454E+01
NMSNV	0.309E+02	0.470E+05	0.110E+01	0.444E-01	0.432E+01
KW	0.667E+04	0.400E+05	0.313E-02	0.547E-01	0.527E+01

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.103E+03	0.274E+03	0.717E-10	0.533E+00	0.555E+01
RS9	0.733E+03	0.208E+04	0.947E-10	0.257E+00	0.261E+01
NMSNR	0.352E+02	0.497E+05	0.135E+01	0.126E+00	0.128E+01
NMSNV	0.342E+02	0.478E+05	0.131E+01	0.118E+00	0.120E+01
KW	0.667E+04	0.400E+05	0.313E-02	0.119E+00	0.120E+01

Table D.6: Output at Termination for Multivariate Function 6.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.119E+03	0.327E+03	0.736E-10	0.333E+00	0.144E+02
RS9	0.831E+03	0.238E+04	0.939E-10	0.573E-01	0.252E+01
NMSNR	0.325E+02	0.594E+05	0.729E+00	0.653E-03	0.283E-01
NMSNV	0.314E+02	0.571E+05	0.785E+00	0.677E-03	0.300E-01
KW	0.625E+04	0.500E+05	0.392E-02	–	–

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.123E+03	0.335E+03	0.743E-10	0.237E+00	0.412E+01
RS9	0.835E+03	0.238E+04	0.954E-10	0.364E-01	0.636E+00
NMSNR	0.358E+02	0.604E+05	0.864E+00	0.485E-02	0.831E-01
NMSNV	0.337E+02	0.575E+05	0.825E+00	0.477E-02	0.825E-01
KW	0.625E+04	0.500E+05	0.000E+00	–	–

Table D.7: Output at Termination for Multivariate Function 7.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.202E+03	0.598E+03	0.839E-10	0.740E+00	0.717E+02
RS9	0.131E+04	0.395E+04	0.950E-10	0.393E+00	0.384E+02
NMSNR	0.344E+02	0.116E+06	0.157E+01	0.488E-02	0.475E+00
NMSNV	0.343E+02	0.118E+06	0.163E+01	0.458E-02	0.451E+00
KW	0.556E+04	0.100E+06	0.701E-02	0.244E-01	0.236E+01

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.226E+03	0.638E+03	0.755E-10	0.190E+00	0.191E+01
RS9	0.135E+04	0.403E+04	0.951E-10	0.558E-01	0.562E+00
NMSNR	0.500E+02	0.123E+06	0.161E+01	0.637E-02	0.643E-01
NMSNV	0.438E+02	0.119E+06	0.197E+01	0.622E-02	0.619E-01
KW	0.556E+04	0.100E+06	0.701E-02	0.287E-01	0.289E+00

Table D.8: Output at Termination for Multivariate Function 8.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.184E+03	0.542E+03	0.826E-10	0.985E+00	0.988E+02
RS9	0.122E+04	0.362E+04	0.952E-10	0.972E+00	0.976E+02
NMSNR	0.350E+02	0.104E+06	0.361E+01	0.242E+00	0.243E+02
NMSNV	0.422E+02	0.101E+06	0.462E+01	0.838E-01	0.841E+01
KW	0.562E+04	0.900E+05	0.626E-02	0.400E+00	0.401E+02

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.188E+03	0.549E+03	0.805E-10	0.913E+01	0.899E+02
RS9	0.125E+04	0.367E+04	0.950E-10	0.637E+01	0.628E+02
NMSNR	0.356E+02	0.104E+06	0.876E+01	0.602E+00	0.593E+01
NMSNV	0.608E+02	0.982E+05	0.603E+01	0.146E+00	0.143E+01
KW	0.562E+04	0.900E+05	0.626E-02	0.923E+00	0.908E+01

Table D.9: Output at Termination for Multivariate Function 9.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.184E+03	0.543E+03	0.825E-10	0.986E+00	0.929E+02
RS9	0.122E+04	0.362E+04	0.952E-10	0.750E+00	0.708E+02
NMSNR	0.349E+02	0.104E+06	0.225E+01	0.265E-01	0.251E+01
NMSNV	0.348E+02	0.103E+06	0.212E+01	0.288E-01	0.272E+01
KW	0.562E+04	0.900E+05	0.626E-02	0.948E-01	0.895E+01

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.213E+03	0.589E+03	0.749E-10	0.214E+01	0.206E+02
RS9	0.127E+04	0.372E+04	0.956E-10	0.101E+01	0.972E+01
NMSNR	0.454E+02	0.100E+06	0.317E+01	0.151E+00	0.144E+01
NMSNV	0.516E+02	0.986E+05	0.302E+01	0.871E-01	0.838E+00
KW	0.562E+04	0.900E+05	0.626E-02	0.120E+00	0.115E+01

Table D.10: Output at Termination for Multivariate Function 10.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.393E+03	0.914E+03	0.595E-13	0.160E+00	0.542E-03
RS9	0.759E+03	0.200E+04	0.210E-12	0.119E+00	0.462E-03
NMSNR	0.158E+03	0.341E+05	0.296E+00	0.522E-01	0.875E-03
NMSNV	0.165E+03	0.343E+05	0.218E+00	0.217E-01	0.489E-03
KW	0.750E+04	0.300E+05	0.000E+00	–	–

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.252E+03	0.535E+03	0.556E-13	0.113E+01	0.213E-01
RS9	0.829E+03	0.220E+04	0.240E-12	0.112E+01	0.224E-01
NMSNR	0.158E+03	0.339E+05	0.292E+00	0.105E+01	0.224E-01
NMSNV	0.165E+03	0.343E+05	0.224E+00	0.102E+01	0.216E-01
KW	0.750E+04	0.300E+05	0.000E+00	–	–

Table D.11: Output at Termination for Multivariate Function 11.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.114E+03	0.321E+03	0.679E-10	0.111E+01	0.108E+03
RS9	0.821E+03	0.235E+04	0.937E-10	0.780E+00	0.755E+02
NMSNR	0.328E+02	0.586E+05	0.115E+01	0.121E+00	0.116E+02
NMSNV	0.350E+02	0.556E+05	0.981E+00	0.740E-01	0.727E+01
KW	0.625E+04	0.500E+05	0.322E-03	–	–
$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.121E+03	0.334E+03	0.715E-10	0.434E+01	0.379E+02
RS9	0.839E+03	0.239E+04	0.958E-10	0.195E+01	0.170E+02
NMSNR	0.401E+02	0.566E+05	0.203E+01	0.333E+00	0.285E+01
NMSNV	0.547E+02	0.549E+05	0.136E+01	0.885E-01	0.761E+00
KW	0.625E+04	0.500E+05	0.000E+00	0.106E+87	0.822E+87

Table D.12: Output at Termination for Multivariate Function 12.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.980E+02	0.265E+03	0.709E-10	0.281E-01	0.568E+00
RS9	0.719E+03	0.204E+04	0.936E-10	0.149E-01	0.727E+00
NMSNR	0.331E+02	0.491E+05	0.579E+00	0.391E-02	0.982E-01
NMSNV	0.329E+02	0.483E+05	0.546E+00	0.419E-02	0.100E+00
KW	0.667E+04	0.400E+05	–	–	–

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.982E+02	0.265E+03	0.723E-10	0.876E-02	0.354E-01
RS9	0.720E+03	0.205E+04	0.947E-10	0.115E-01	0.337E-01
NMSNR	0.334E+02	0.479E+05	0.560E+00	0.387E-02	0.225E-01
NMSNV	0.330E+02	0.477E+05	0.533E+00	0.404E-02	0.237E-01
KW	0.667E+04	0.400E+05	–	–	–

Table D.13: Output at Termination for Multivariate Function 13.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.187E+03	0.547E+03	0.823E-10	0.315E+00	0.332E+02
RS9	0.121E+04	0.359E+04	0.953E-10	0.127E+00	0.120E+02
NMSNR	0.392E+02	0.969E+05	0.837E+00	0.185E-01	0.171E+01
NMSNV	0.448E+02	0.986E+05	0.853E+00	0.172E-01	0.153E+01
KW	0.562E+04	0.900E+05	0.627E-02	0.278E-01	0.241E+01

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.194E+03	0.556E+03	0.810E-10	0.347E+00	0.329E+01
RS9	0.122E+04	0.361E+04	0.955E-10	0.826E-01	0.795E+00
NMSNR	0.447E+02	0.990E+05	0.764E+00	0.147E-01	0.144E+00
NMSNV	0.416E+02	0.994E+05	0.800E+00	0.156E-01	0.151E+00
KW	0.562E+04	0.900E+05	0.627E-02	0.278E-01	0.260E+00

Table D.14: Output at Termination for Multivariate Function 14.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.115E+03	0.321E+03	0.675E-10	0.780E+00	0.795E+02
RS9	0.828E+03	0.237E+04	0.940E-10	0.506E+00	0.515E+02
NMSNR	0.321E+02	0.565E+05	0.123E+01	0.317E-01	0.324E+01
NMSNV	0.317E+02	0.570E+05	0.115E+01	0.351E-01	0.358E+01
KW	0.625E+04	0.500E+05	0.400E-02	0.103E+00	0.104E+02

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.130E+03	0.348E+03	0.760E-10	0.847E+00	0.761E+01
RS9	0.854E+03	0.243E+04	0.949E-10	0.354E+00	0.317E+01
NMSNR	0.396E+02	0.585E+05	0.162E+01	0.139E+00	0.124E+01
NMSNV	0.380E+02	0.570E+05	0.156E+01	0.146E+00	0.129E+01
KW	0.625E+04	0.500E+05	0.400E-02	0.261E+00	0.232E+01

Table D.15: Output at Termination for Multivariate Function 15.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.184E+03	0.542E+03	0.824E-10	0.989E+00	0.985E+02
RS9	0.122E+04	0.361E+04	0.951E-10	0.870E+00	0.866E+02
NMSNR	0.350E+02	0.104E+06	0.318E+01	0.111E+00	0.111E+02
NMSNV	0.375E+02	0.103E+06	0.320E+01	0.788E-01	0.784E+01
KW	0.562E+04	0.900E+05	0.626E-02	0.216E+00	0.215E+02

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.203E+03	0.572E+03	0.778E-10	0.618E+01	0.585E+02
RS9	0.126E+04	0.369E+04	0.955E-10	0.332E+01	0.314E+02
NMSNR	0.397E+02	0.100E+06	0.515E+01	0.734E+00	0.696E+01
NMSNV	0.530E+02	0.982E+05	0.414E+01	0.516E+00	0.489E+01
KW	0.562E+04	0.900E+05	0.626E-02	0.892E+00	0.846E+01

Table D.16: Output at Termination for Multivariate Function 16.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.763E+02	0.204E+03	0.728E-10	0.203E+00	0.460E+01
RS9	0.636E+03	0.181E+04	0.948E-10	0.518E-01	0.113E+01
NMSNR	0.317E+02	0.343E+05	0.639E+00	0.199E-02	0.436E-01
NMSNV	0.312E+02	0.341E+05	0.613E+00	0.255E-02	0.581E-01
KW	0.750E+04	0.300E+05	0.232E-02	0.106E-01	0.231E+00

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.788E+02	0.210E+03	0.737E-10	0.680E-01	0.228E+00
RS9	0.645E+03	0.184E+04	0.957E-10	0.268E-01	0.884E-01
NMSNR	0.335E+02	0.346E+05	0.822E+00	0.176E-02	0.595E-02
NMSNV	0.333E+02	0.344E+05	0.900E+00	0.112E-02	0.377E-02
KW	0.750E+04	0.300E+05	–	–	–

Table D.17: Output at Termination for Multivariate Function 17.

$GAP/\sigma \approx 1$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.129E+03	0.347E+03	0.730E-10	0.403E+00	0.198E+01
RS9	0.855E+03	0.243E+04	0.947E-10	0.293E+00	0.144E+01
NMSNR	0.395E+02	0.576E+05	0.108E+01	0.995E-01	0.488E+00
NMSNV	0.385E+02	0.592E+05	0.996E+00	0.979E-01	0.480E+00
KW	0.625E+04	0.500E+05	0.400E-02	0.156E+00	0.761E+00

$GAP/\sigma \approx 10$

Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.129E+03	0.348E+03	0.715E-10	0.675E+00	0.350E+01
RS9	0.854E+03	0.243E+04	0.943E-10	0.523E+00	0.269E+01
NMSNR	0.399E+02	0.548E+05	0.110E+01	0.194E+00	0.100E+01
NMSNV	0.382E+02	0.566E+05	0.102E+01	0.187E+00	0.970E+00
KW	0.625E+04	0.500E+05	0.400E-02	0.234E+00	0.121E+01

Table D.18: Output at Termination for Multivariate Function 18.

$GAP/\sigma \approx 1$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.206E+03	0.604E+03	0.858E-10	0.219E-02	0.308E+00
RS9	0.131E+04	0.394E+04	0.950E-10	0.520E-03	0.421E+00
NMSNR	0.410E+02	0.119E+06	0.621E+00	0.122E-03	0.858E-01
NMSNV	0.401E+02	0.124E+06	0.611E+00	0.124E-03	0.758E-01
KW	0.556E+04	0.100E+06	–	–	–

$GAP/\sigma \approx 10$					
Method	NumIter	NumEval	StepSize	Error	PerGap
NM	0.208E+03	0.606E+03	0.861E-10	0.118E-02	0.610E-02
RS9	0.131E+04	0.394E+04	0.962E-10	0.507E-03	0.477E-02
NMSNR	0.426E+02	0.122E+06	0.632E+00	0.152E-03	0.368E-02
NMSNV	0.414E+02	0.120E+06	0.638E+00	0.125E-03	0.326E-02
KW	0.556E+04	0.100E+06	–	–	–

Vita

Name:	John James Tomick
[redacted]	[redacted]
[redacted]	[redacted]

PII Redacted

Education:

Valedictorian, Windham High School, Willimantic, Connecticut, June 1980.

Distinguished Graduate with Honors, Bachelor of Science, Mathematical Sciences, United States Air Force Academy, Colorado Springs, Colorado, May 1984.

Distinguished Graduate, Master of Science, Operations Research, Air Force Institute of Technology, Dayton, Ohio, December 1988.

Doctor of Philosophy, Statistics and Operations Research, The Pennsylvania State University, University Park, Pennsylvania.

Publications:

—, Litko J. R. & Bauer, K. W. (1989). A Comparison of Control Variates for Queueing Network Simulation. *Modeling and Simulation, 20.* Instrument Society of America, 1081-1085.

Honorary Societies: Tau Beta Pi (Engineering)