DEVELOPMENT AND APPLICATION OF MULTISTEP COMPUTATIONAL TECHNIQUES FOR CONSTRAINED AND UNCONSTRAINED MATHEMATICAL FUNCTIONS

by

Wayne C. Turner

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Industrial Engineering and Operations Research

APPROVED:

P. M. Ghare, Chairman

J. W. Schmidt

M. H. Agee

E. R. Clayton

R. E. Taylor

August, 1971

Blacksburg, Virginia

ACKNOWLEDGEMENTS

The author at this time would like to recognize just some of the many people who helped him along the way. First and foremost are his wife Mrs. Kathleen Turner and his parents Mr. and Mrs. Percy Turner. Their understanding and love always made the work seem worthwhile.

Dr. Parbhakar Ghare deserves special recognition as he has been the author's advisor for four years through the M.S. and Ph.D. degrees. Dr. Ghare's amazing ability to see through difficult problems and to be able to guide the author through these in an unbelievably friendly manner made the work quite enjoyable.

Mrs. Cathy Hall, Miss Dot Nolen, and Mrs. Marion Price deserve credit for their ability to read the author's handwriting and type it into finished form. This in itself deserves a degree for them.

Personal friends such as Dr. Edwin Ruark and the brothers of Sigma Lambda Social Fraternity provided the friendship and encouragement that is very helpful.

Finally, the author's committee, Drs. J. W. Schmidt, M. H. Agee, and R. E. Taylor of the Industrial Engineering and Operations Research Department and Dr. E. R. Clayton of the Business Department provided a wealth of knowledge upon which the author frequently drew.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES (CONT.)

LIST OF TABLES (CONT.)

LIST OF FIGURES

CHAPTER I

PROBLEM DESCRIPTION

Introduction, Discussion of Unconstrained Problems, and Computational Techniques

The objective of this dissertation is the study of multistep directional procedures in general for solving certain optimization problems. Various forms of optimization problems are considered, unconstrained as well as constrained and those with experimental objective functions as well as algebraic objective functions. These multistep procedures are compared with other procedures to bring out the similarities and differences. Specifically, where sufficient similarities exist between a known procedure and the corresponding multistep formulation, the procedure is modified to fit the exact multistep form for improved performance.

Chapter I is essentially an introductory chapter. First, the unconstrained problem is described and a test function presented. The two major directions available for directional search procedures (coordinate and gradient) are discussed next. Then, the solution of simultaneous linear equations by gradient methods is reviewed leading to the extension to unconstrained optimization. Finally, in Chapter I, the constrained problem is presented and some well known solution procedures are discussed.

Chapter II examines various multistep algorithms and applies them to the simple test function. The ten most promising of these

algorithms are next applied to two other test functions.

Chapter III examines the use of those multistep concepts on constrained functions, both by examining some well known algorithms to see how they can be adapted for multistep techniques, and by proposing a new algorithm. Chapter III then presents this new algorithm in depth and shows the algorithm is basically a penalty formulation. This algorithm is next used to solve a "real world" problem to demonstrate its efficiency and the applicability of nonlinear programming to "real world" problems.

Chapter IV first presents in summary form most of the research and the conclusions that can be derived. Finally, some thirteen areas for future research are listed and discussed.

It is, by now, quite obvious that the research covers an extremely broad area. At times, therefore, the results may appear to be a series of disjointed works, but together they form a complete package. Three main areas are mentioned. They are:

(1) Simultaneous linear equations.

(2) Unconstrained optimization.

(3) Constrained optimization.

These areas do have relationships. Basically, area (1) is covered to bring to light the concepts of incomplete relaxation and multistep theory that are used later for both areas (2) and (3). Area (1) also develops an algorithm for the solution of these equations that is used in area (2) to develop an algorithm for unconstrained

optimization. Areas (2) and (3), of course, are directly related because any constrained problem can be transformed into a series of unconstrained problems by using a penalty formulation. This is used a great deal in this dissertation as most experiments are run on the simpler unconstrained problems with the results extended to constrained problems.

A great number of computer programs are used in this research. The number is so great that they have not been included in the text. A manual is available, however, showing any or all of the programs. Requests should be addressed to the Industrial Engineering and Operations Research Department, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061.

For the statement of the problem, the objective function can be assumed to be algebraic with no loss of generality as in most experimental problems the objective function can be assumed to have an underlying algebraic form although it is unknown. Similarly, it is well known (see Courant [6]) that any constrained optimization can be transformed into an equivalent uncontrained problem by using penalty functions as discussed later in this chapter. This means that the statement of the problem can be in terms of an unconstrained problem with algebraic objective function without deviating from the goal of considering constrained problems as well as experimental objective functions.

The constrained problem may be simply stated as finding the minimum of a mathematical function $f(\bar{X})$ for an X included within a

set Γ. The set Γ is bounded by the constraints. The minimum of this function is then defined as

$\bar{X} | \bar{X} \in \Gamma$ such that

$$f(X) \geq f(\bar{X}) \text{ for all } X \in \Gamma.$$

This constrained minimization problem (hereafter called MPC) may be defined, then, as:

$$\text{Find } \bar{X}, \text{ if it exists, such that } f(\bar{X}) = \min_{X \in \Gamma} f(X).$$

The set Γ is the feasible region of the problem. The unconstrained minimization problem is one where $\Gamma = E^n$ (N dimensional euclidian space). It may be stated as: Find $\bar{X}$ such that

$$f(\bar{X}) = \min_{X \in E^n} f(X).$$

This is referred to as MP to distinguish from the constrained case. It is easily noticed that these two definitions can be altered to include the maximum problem simply by reversing all inequalities. All discussions will be concerning MP or MPC but if it is necessary to refer to the maximum problems, then they shall be designated as Max P and Max PC.

The minimization problem (MP) is analagous to finding the point

of lowest elevation in a valley. The constrained minimization problem (MPC) is similar to finding the point of lowest elevation on one of the farms located within the valley. If the point of lowest elevation within the valley, the solution of the unconstrained problem, happens to be on the farm, then it is obviously the point of lowest elevation on the farm, the solution of the constrained problem. In this case, the consideration of the constraints is superfluous. If the point of lowest elevation within the valley does not lie on the farm, then it becomes necessary to solve the constrained problem.

Quite often a mathematical function, and valleys and farms for that matter, may have several apparent minima. For example, the farm may have several points that are lower than the immediate neighboring region around those points, but there is no quick way of knowing which of these apparent lowest points is the absolute lowest point. These apparent low points are called local minima as distinguished from the absolute minimum. The type of function being considered will tell immediately whether a low point can be guaranteed to be the absolute minimum or not; but for the time being the problem of finding local minima may be defined as (LMPC):

Find $\bar{X}\varepsilon\Gamma$, if it exists, such that for some open ball $B_\delta(\bar{X})$ around $\bar{X}$ with radius δ

$$X\varepsilon B_\delta(\bar{X})\Omega\Gamma\rightarrow f(X) \geq f(\bar{X}).$$

The unconstrained local minimum problem (LMP) appears once again when $\Gamma = E^n$ or when

$$B_{\delta}(\bar{X}) \cap \Gamma = B_{\delta}(\bar{X}).$$

It is outside the scope of this research to demonstrate the conditions for MP as opposed to LMP; but it may be summarized by saying that if f(X) is strictly convex over Γ, then a solution to LMP or MP is a unique solution to MP and if f(X) is convex then a solution to LMP or MP is a solution to MP. If f(X) is not convex over Γ, a solution to LMP cannot be said to be a solution to MP; but, of course, a solution to MP is always a solution to LMP. To limit the scope of this research, convexity will be assumed for all problems unless otherwise stated. The most widely used test for convexity is the test of the Hessian maxtrix H(X)-the matrix of second partial derivatives. If H(X) is positive definite over Γ, $\Theta(X)$ is strictly convex over Γ. If H(X) is positive definite at a candidate point X°, then X° is a solution to LMP but not necessarily to the MP. More detailed discussion of the tests for convexity can be found in Mangasarian [4], Gue and Thomas [16], or Schmidt [28].

To establish a beach head in the area, a sample problem is used to illustrate these concepts. A function very similar to Rosenbrock's function is chosen as the objective function:

$$f(X) = 100(x_1 - x_2)^2 + (1 - x_1)^2 + (x_2 - 2x_3)^2.$$

Since $\Gamma = E^3$, the problem is an unconstrained minimization problem (MP or LMP). The Hessian matrix H(X) for the problem is

$$\begin{vmatrix} 202 & -200 & 0 \\ -200 & +202 & -4 \\ 0 & -4 & +8 \end{vmatrix} .$$

It is elementary to demonstrate that this matrix is positive definite as all the principal leading minors are greater than zero. The function then is strictly convex and any solution encountered is a solution to MP. Concern over local minima, therefore, is not necessary.

The function can be seen to have many interaction terms and there is a fairly steep ridge on the response surface. This means that any technique that reaches the minimum of this function in a reasonable number of iterations justifies further consideration. All proposed algorithms are tested first on this function which has an absolute minimum at $X^* = (1, 1, .5)$ with $f(X^*) = 0$. In this research $\bar{X}$ and X^* are the same. All methods originate at $X^\circ = (4, 4, 4)$ and the methods are terminated at

(1) Error = .0001 where Error - $[(1 - x_1)^2 + (1 - x_2)^2 + (.5 - x_3)^2]$.

or (2) N(the number of iterations) > 1000,

whichever comes first. A fixed value of error is chosen as a stopping criteria as opposed to some value of f(X) since, as Ghare [11] mentions, f(X) can become quite flat in the local area around X^* while error is steeper. Also, error is a better measure of $X^* - X$.

Perhaps the most widely used set of techniques for unconstrained optimization is the directional search concept which is a search

along some improving direction. The basis for this method is that if X* is a minimum point of f(X), it also is a minimum along any direction X = X*+ u at X*. This holds globally for convex functions and locally for non-convex functions.

The procedural steps for a directional search algorithm are given below:

(1) Choose a direction u^1. Any improving direction qualifies for u.

(2) Determine a minimum along $X^\circ + u^1 = X^1$.

(3) Choose another direction, u^i, linearly independent of u^{i-1}.

(4) Determine the minimum along $X^i + u^i = X^{i+1}$.

(5) Repeat 3 and 4 until $X^{i+1} = X^*$ or the error is less than ε.

One candidate for the improving direction is the direction of steepest descent which is opposite to the gradient direction at a point. The gradient of a function at a point is defined as the matrix of partial derivatives of the function at that point. Mangasarian [24] says a function f has a partial derivative at X° with respect to $x_i (i = 1, \ldots, n)$ if the expression

$$\frac{f(x^\circ_1, \ldots, x^\circ_{i-1}, x^\circ_i+\delta, x^\circ_{i+1}, \ldots, x^\circ_n) - f(x^\circ_1, \ldots, x^\circ_n)}{\delta}$$

tends to a finite limit when δ tends to 0. The limit is the partial derivative of f with respect to x_i at X°, and is designated by $\frac{\delta f(X^\circ)}{\delta x_i}$. The 1xn dimensional vector of partials is called the gradient of f at X° and is denoted by $\nabla f(X^\circ)$.

That is,

$\nabla f(X^\circ) = (\frac{\delta f(X^\circ)}{\delta x_1}, \ldots\ldots, \frac{\delta f(X^\circ)}{\delta x_n})$. This vector is called the gradient because it points in the direction for which the response surface has the steepest positive slope. This was shown by Wilde and Beightler [29]. If this is the line of steepest ascent, a "small" move along this line will produce the greatest rate of increase in Θ. Similarly, a small move along the direction described by $-\nabla f(X^\circ)$ will produce the greatest rate of decrease in $f(X)$.

The next logical move is to proceed along this direction, $-\nabla f(X^\circ)$, until the function stops decreasing. This step size is designated as ρ. The determination of ρ for a particular move is relatively simple for there exists the function $f(X^\circ - \rho\nabla f(X^\circ))$ where the only unknown is ρ. The objective is to determine where this function reaches a minimum. This can be done by a uni-dimensional search on ρ such as Fibonacci search, or incrementating, or an optimum ρ can be calculated by direct differentiation.

Elementary calculus demonstrates that a necessary condition (sufficient also for convex functions such as the test function of this thesis) for this function to obtain an optimum is for the first derivative to vanish. That is

$$\frac{\delta f(X^\circ - \rho^*\nabla f(X^\circ))}{\delta \rho^*} = 0.$$

It is necessary to differentiate, set equal to zero, and solve for ρ. Then, the point must be tested for a minimum. In the case of a convex function, this is not necessary; and as long as ρ^* produces a local minimum which is better than the original point, the

method can proceed. Quite often the algebra involved in the solution for ρ is complicated enough so that a one-dimensional search on ρ is more efficient.

The next step in the method is to calculate a new gradient at this point X^1 called $\nabla f(X^1)$, a new ρ^* and a new point X^2, etc., until the process converges to a sufficiently small value. The sequence generated is a one such that $f(X^\circ) > f(X^1) > .... > f(X^N)$, where X^N is the last point.

The gradient algorithm then may be stated as:

(1) Select an initial point X° and determine the gradient of the function at $X^\circ(\nabla^\circ(X^\circ))$, set $i = 0$.

(2) Calculate ρ as the step size minimizing the function along the direction $-\nabla^i(X)$. (That is,

$$\rho = \rho^*, \text{ such that } f(X^i - \rho\nabla^i(X) > f(X^i - \rho^*\nabla^i(X))$$

for all $0 \leq \rho \leq \rho^*$.)

Do this either by a search on ρ or by direct differentiation such as

$$\frac{\delta f(X^i - \rho\nabla^i(X)}{\delta\rho} = 0 \mid \rho = \rho^*.$$

(3) Determine the new point ($X^{i+1} = X^i - \rho\nabla^i(X)$). Test for optimality; if $X^{i+1}=X^*$, stop. If not, set $i = i+1$, calculate the new gradient $\nabla^i(X)$, and go to 2.

Convergence, to at least a local minimum for a bounded problem, is easy to show as the function value is monotonically decreasing but N cannot be proven to be finite. Hereafter, an algorithm is

said to converge if $|X^* - X|$ can be made smaller than any $\sigma > 0$ in a finite number of steps. In fact, it is quite simple to generate a problem where N is infinite as this is the normal case. This is the major disadvantage of the gradient technique. Any time a function has a pronounced ridge, the gradient technique has an extremely slow rate of convergence (see Figures 1 and 2).

The rate of convergence of an algorithm can be measured by either or both of the following ratios:

$$(1) \quad \frac{f(X^{i+1})}{f(X^i)}$$

$$(2) \quad \frac{|X^* - X^{i+1}|}{|X^* - X^i|} .$$

Both of these ratios measure the improvement at each stage of optimization. The problem with the direction of steepest descent is that these ratios generally decrease with increasing i. That is, the rate of improvement slows down as the algorithm progresses. This is the reason that quite often N is infinite as shown in Figure 2.

Various modifications have been suggested to overcome this slow rate of convergence. One such proposal is to use a direction different from the actual direction of steepest descent. Examples of this are the pattern search method of Hooke and Jeeves [21] and the deflected gradient method of Fletcher and Powell [10]. These are discussed and shown later in this thesis. Another suggestion is to use the direction of steepest descent but do not go the exact distance ρ each time. This was proposed first by Kantarovitch, as described by Fadeev and

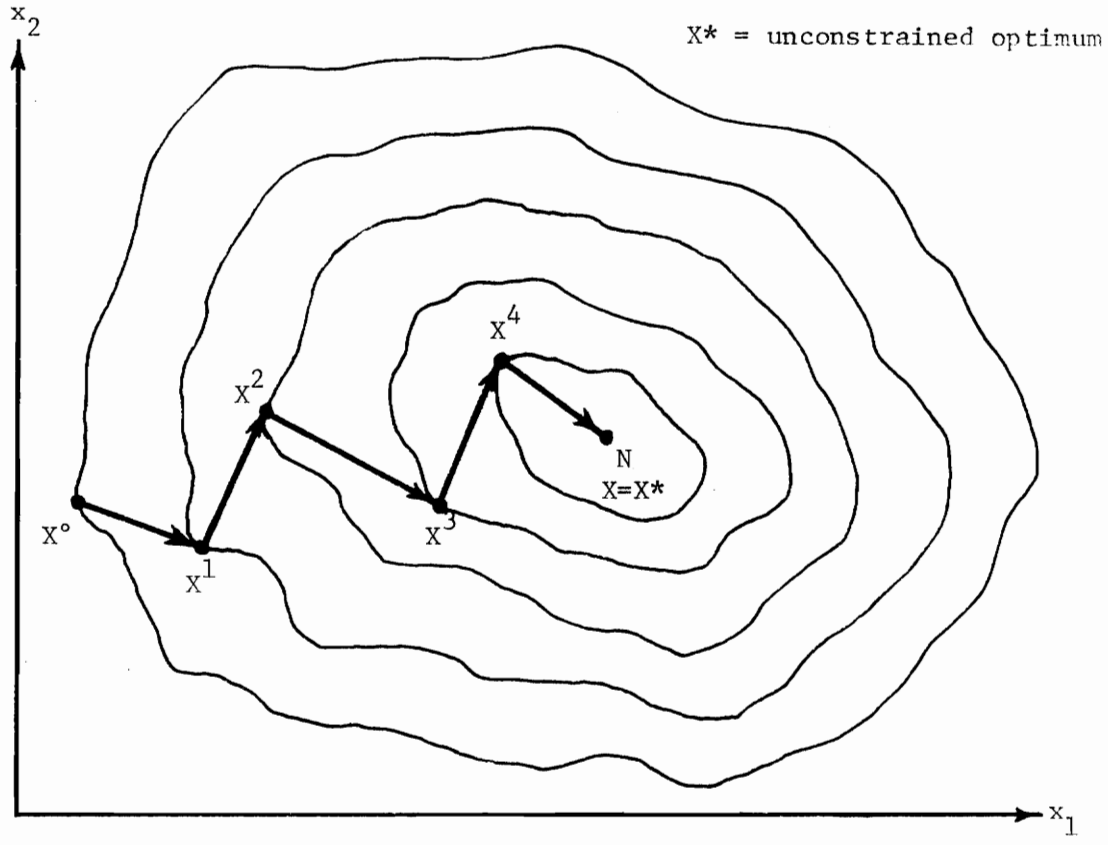


Figure 1. The Gradient Technique - No Ridge.

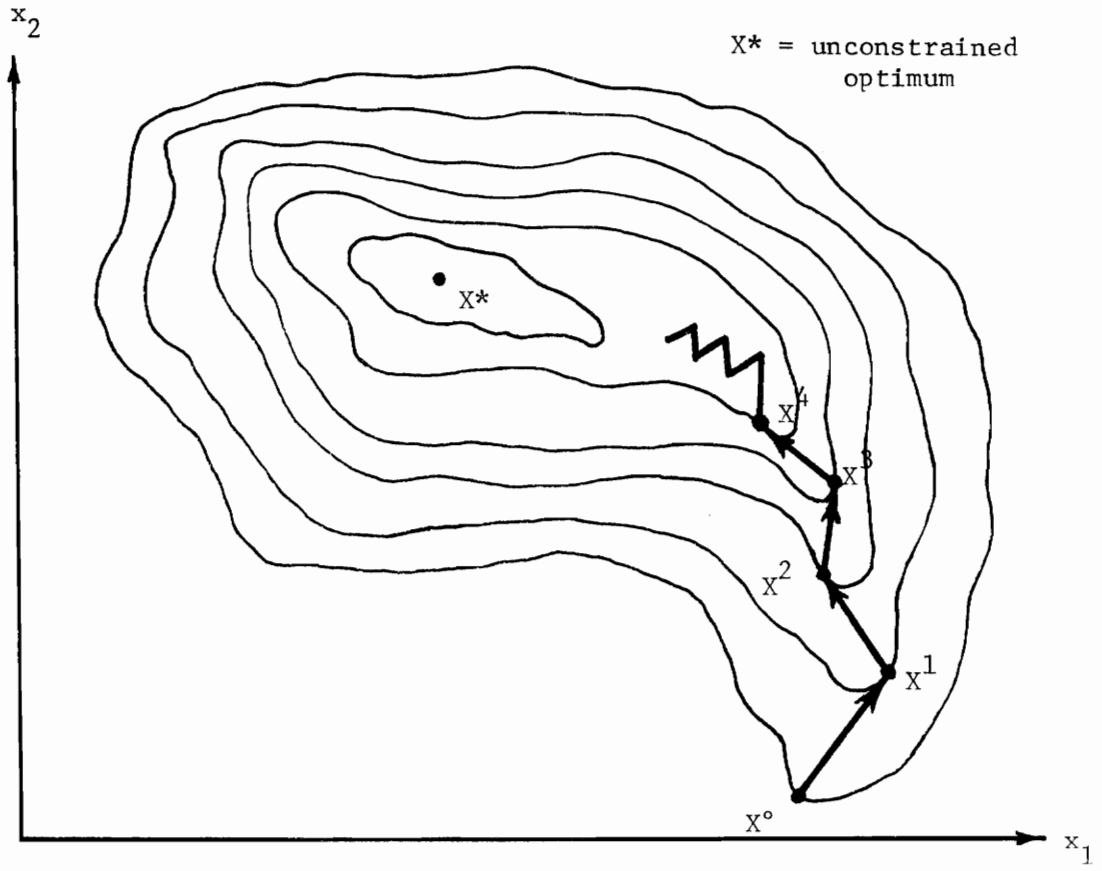


Figure 2. A Ridge and the Gradient Technique.

Fadeeva [8], in his solution of simulataneous equations and enlarged a great deal by Ghare [12] to other mathematical functions. The examination of taking different step sizes in both the gradient and the coordinate directions is the first goal of this thesis. To do this, the works of Kantorovitch and Ghare are used a great deal.

The basis for different step size techniques can be seen quite easily by considering a figure as developed by Fadeev and Fadeeva [8] and shown in Figure 3. From any initial point X°, there exists a continuous line to the optimum point X^*. Along this line the function $f(X)$ is continuously decreasing. Several such lines are shown in Figure 3, and the curved line $X^\circ \rightarrow X^*$ is the continuous line described above. The ideal procedure starts at X° and proceeds along this line to X^*. The direction of steepest descent is the tangent line shown at X°. So, a procedure using the direction of steepest descent consists of a series of tangent lines as shown in Figure 3.

It is interesting to note that the optimal curve, $X^\circ \rightarrow X^*$, coincides with a direction of steepest descent if all steps were of infinitesimal length. Perhaps, then, some where between the infinitesimal length and the complete minimization step, there exists a step size such that the number of iterations required for convergence is less than the normal steepest descent method. (There is a trade off function here where very small step lengths take an extremely large number of iterations due to the small step size, and the direction of steepest descent takes many iterations due to the slow rate of convergence described earlier.) This is the hopeful conjecture

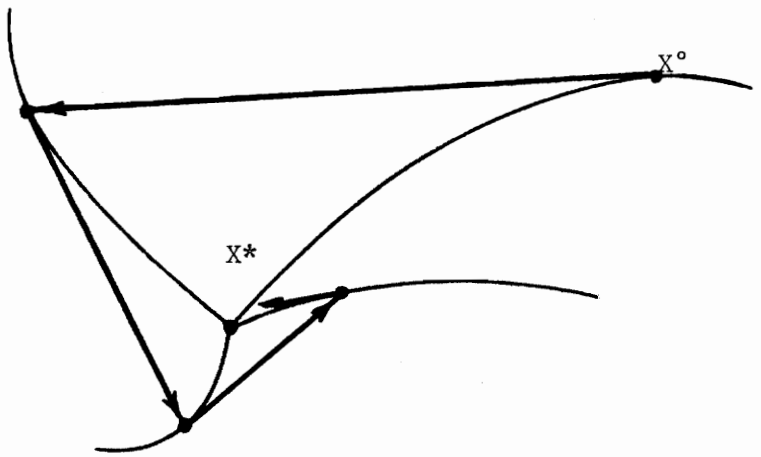


Figure 3. The Optimal Search Trajectory and the Gradient Technique.

of using step sizes other than ρ.

It is interesting to replace $\nabla(X^i)$ in all the arguments above by another improving direction where movement is only permitted along the coordinate axes of the variables one at a time (i.e., $\Delta x_i = 0, i \neq j$). In other words, movement is permitted only on the coordinate axes and then only one variable at a time. In considering a function without highly irregular contours, this offers some promise; but, intuitively, the method could be quite slow unless some acceleration modification is allowed.

Perhaps the simplest method of this sort, as developed by Friedman and Savage and described by Wilde and Beightler [30] is the sectioning or one-at-a-time method. In this method all variables except one are held constant and a search is conducted along that variable's coordinate direction to determine the minimum point. That variable is held constant, and another is allowed to vary, etc, until all n variables are examined with no improvement. The sectioning method has the advantages of being very simple and easy to program; also it performs quite well when the response contours are spherical or nearly so (see Figure 4). When the contours differ significantly from hyperspheres, the method becomes quite inefficient and, in some cases, quite ineffective (see Figures 5 and 6). It is, therefore, necessary to judge sectioning, when used with no other method, as inappropriate unless the contours of the response surface are nearly spherical. Of course, it is possible to incorporate other methods with sectioning to improve performance.

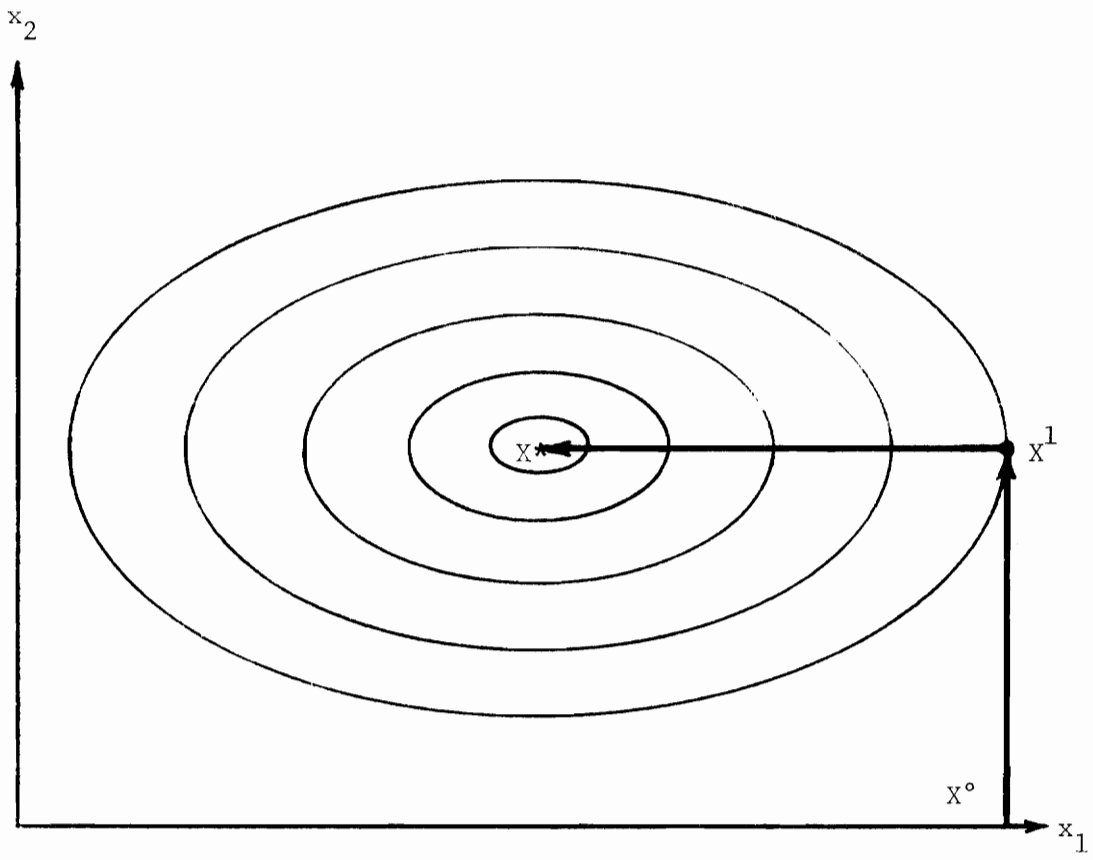


Figure 4. Sectioning - No Interaction or Ridges.

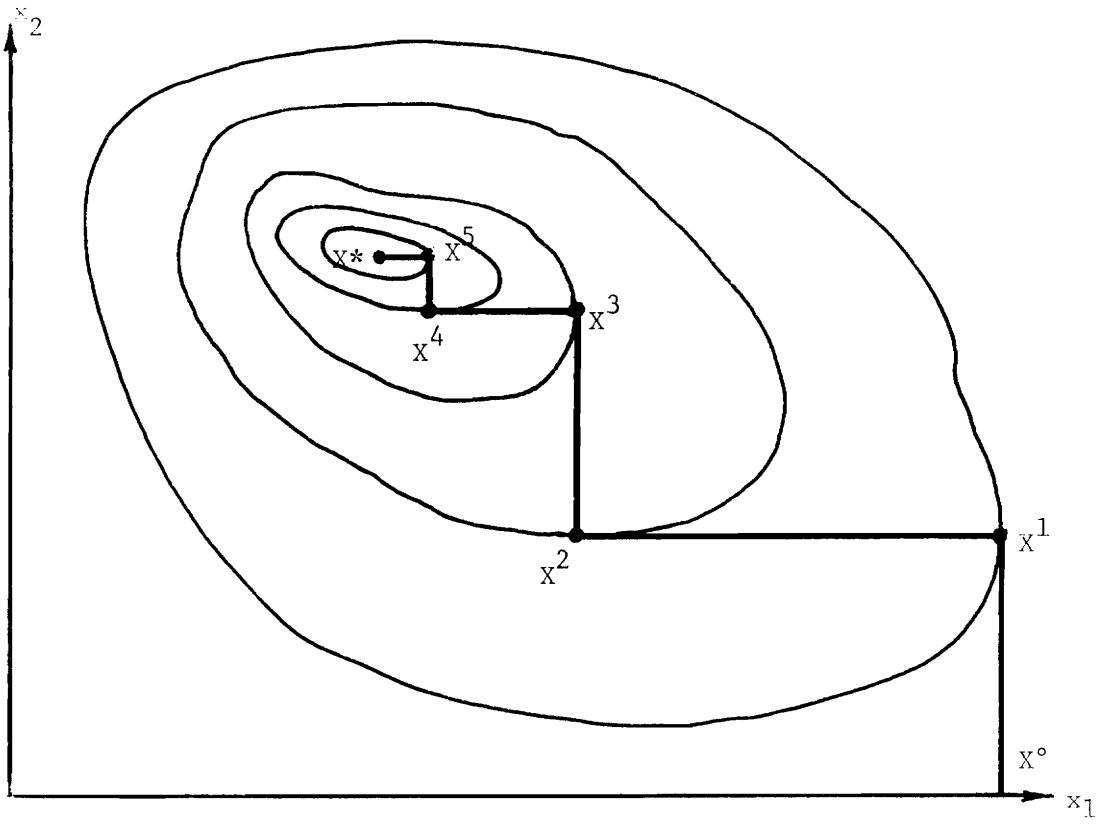


Figure 5. Sectioning Interaction - No Ridge.

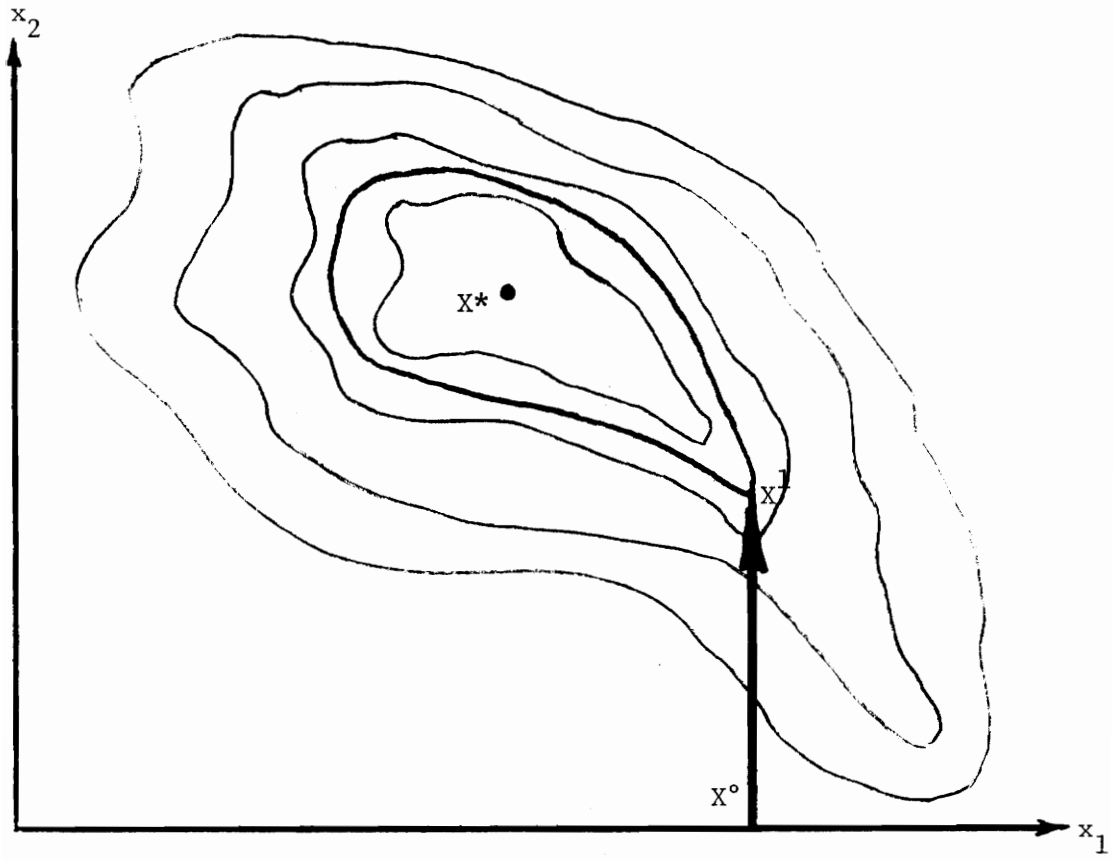


Figure 6. Sectioning Interaction and Ridge.

Another coordinate direction method that is quite easy to program with acceptable valley following properties is the Pattern Search method of Hooke and Jeeves [21] as described by Wilde and Beightler [32]. The motivation behind this method is simply that a successful move in the past may justify a further move in that direction. In short, the method starts off with short choppy steps and enlarges these steps if the direction continues to produce better results. If the direction fails, the method cuts back to short steps and attempts to establish a new direction. The main advantage to this method over sectioning and steepest descent is that in the presence of non-spherical contours, it tends to move rather rapidly along valleys to the optimum. Another advantage that both enjoy is the fact that no derivatives are required.

As an example Wilde and Beightler [30] have shown that after 200 iterations the results shown in Table I were obtained when attempting to minimize Rosenbrock's function

$$f(X) = 100\ (x_2 - x_1^2)^2 + (1 - x_1)^2$$

starting at $X^\circ = (-1.2,1.)$.

This indicates that pattern search is both efficient and effective in the minimization of Rosenbrock's function. Experience has shown that in general it is a "good" method. It is also possible to improve the performance of this basic method by some modifications. Some of these possible modifications are presented and discussed later in this research. To be able to develop and modify these

Table I. Comparison of Methods on Rosenbrock's Function

Method	f(X*)	X_1*	X_2*
Sectioning	3.882	-.970	.945
Gradient	2.578	-.605	.371
Pattern Search	.803	---	---
Optimum	0.0	1.	1.

procedures, the pattern search method is described below as a step-by-step algorithm.

(1) Select an initial point X° with a function value $f(X^\circ)$, set $i = 1$, Kount = 0, $j = 1$, and δ_{min} = smallest step size desirable.

(2) Form $\Delta X = (0, 0, 0, \ldots, \delta_i, \ldots 0, 0)$

(all $\delta x_j = 0$, $j \neq i$, $\delta x_i = \delta_i$).

(3) Test the function at $X^j + \Delta X$. If $f(X)^j + \Delta X) < f(X)^j$, then set (a) Kount = 1, (b) $f(X^j) = f(X^j + \Delta X)$ and (c) $X^{j+1} = X^j + \Delta X$. If $f(X^j + \Delta X) > f(X)$, then test the function at $X^j - \Delta X$. If $f(X^j - \Delta X) < f(X^j)$, then set (a) Kount = 1, (b) $f(X^j) = f(X^j) - \Delta X$, and (c) $X^{j+1} = X^j - \Delta X$. If $f(X^j - \Delta X) > f(X^j)$, set $X^{j+1} = X^j$.

(4) Set $i = i + 1$. If $i < N$, go to 2.

(5) If Kount = 1, go to 6.

Set $\delta_i = \delta_i/2$. If $\delta_i \leq \delta_{min}$, STOP as apparent optimum has been found.

If $\delta_i > \delta_{min}$, set Kount = 0, $i = 1$, and go to 2.

(6) Set $X^{j+1'} = 2X^{j+1} - X^j$.

If $f(X^{j+1}) \leq f(X^{j+1})$, set $X^{j+1} = X^{j+1'}$, and $j = j+1$

Set $i = 1$, Kount = 0, and go to 2.

If $f(X^{j+1'}) > f(X^{j+1})$, set $j = j+1$, $i = 1$, Kount = 0, and go to 2

A multistep search technique is a directional search technique where the step sizes are not constant and indeed one may never be equal to another. By some rule or calculation the individual step sizes are varied with the objective of improving convergence. The rules for determination of the step sizes will vary a great deal

but, in every case, it is hoped that these rules will guarantee better convergence for a k step multistep procedure than for k steps of a one step procedure. Most multistep procedures also employ an acceleration technique similar to the one used by Hooke and Jeeves in Pattern Search to pursue any pattern development to the fullest. A s-step multistep method may be defined as:

$$X^1 = X^\circ + \sum_{j=1}^{s} a_j s_j$$

where a_j is the jth step size along the jth direction s_j. If an acceleration technique is employed, then $X^2 = X^\circ + a_2(X^1 - X^\circ)$ where a_2 is chosen to minimize $f(X^\circ + a_2(X^1 - X^\circ))$.

The intuitive reasoning behind multistep procedures is simply that one step procedures consider only that stage's return. A s-step procedure, however, chooses s step sizes, hopefully, so that the final functional $f(X^1)$ or $f(X^2)$ is minimized. The problem of course is in determining the best step sizes. Most often these step sizes are approximated as is discussed later.

Simultaneous Linear Equations and Computational Techniques

In the solution of linear simultaneous equations, especially when the system of equations is near singular, gradient procedures take a long time to converge. Multistep techniques were first developed for improving this convergence. Fadeev and Fadeeva [8] did much research in the application of gradient techniques to the solution of

simultaneous linear equations and Ghare [12] proposed extensions of these results. Both of these works are quite important to the development of this research and hence are summarized below.

Assuming that A is positive definite (necessary since the procedure minimizes a quadratic function where A is the Hessian matrix) the system of equations is defined as

$$AX = F.$$

Kantarovitch proposed (presented by Fadeev and Fadeeva [8]) minimizing through the correct choice of X the function

$$H(X) = (AX, X) - 2(F,X)$$

which differs from the error function $(A(X^* - X), X^* - X)$ by only a constant value. (Note in this research (A,B) signifies the scalar product of vectors A and B.) He then proceeds to show that the direction of steepest descent is given by the residual $r = F - AX$. By solving for the optimum step size, he obtains the following gradient procedure for the solution of simultaneous linear equations:

(1) Select X°, set $i = 0$.

(2) Set $X^{i+1} = X^i + \alpha_i r_i$ where

$$r_i = F - AX^i = r_{i-1} Ar_{i-1},$$

$$\alpha_i = \frac{(r_i, r_i)}{(Ar_i, r_i)}, \text{ and}$$

$$f(X^{i+1}) = f(X^i) - \frac{(r_i, r_i)^2}{(Ar_i, r_i)}.$$

(3) Test X^{i+1} for solution. ((r_i, r_i) may be compared to a

minimum or ($X^{i+1} - X^i$, $X^{i+1} - X^i$) may be compared to a minimum.) If $X^{i+1} = X^*$ then stop. If not, set i = i+1 and go to 2.

It can be proven that this method converges to the solution X* with the rate of geometric progression as shown by Fadeev and Fadeeva [8]. As an example Fadeev and Fadeeva chose the following problem:

$$A = \begin{vmatrix} 1 & .42 & .54 & .66 \\ .42 & 1 & .32 & .44 \\ .54 & .32 & 1 & .22 \\ .66 & .44 & .22 & 1 \end{vmatrix}$$

$$F = [.3 \quad .5 \quad .7 \quad .9 \ ]$$

$$X^* = [1.2577938 \quad .0434873 \quad 1.0391663 \quad 1.4823929]$$

Their results are summarized in Table II.

Fadeev and Fadeeva pursue the subject further in an effort to improve the convergence as the results are promising. They discuss methods of altering the step size. First they examine a method where the length of residual vector is minimized at each iteration. This produced good results but the method, since the step size is less than the step size for the steepest descent method, falls under the broader classification of under relaxation.

If α^i is the step size which minimizes H(X) in the steepest descent method and γ^i is a multiplier used to alter this step (i.e., $\Delta X^i = \gamma^i \alpha^i r^i$), a whole new class of procedures appears. These

Table II. The Residual Gradient Method and Simultaneous Linear Equations

X°	0	0	0	0
X^{1}	.1515525	.2525875	.3536335	.4546575
X^{25}	-1.2573084	.0435634	1.0389273	1.4820379
X^{52}	-1.2577937	.0434873	1.0391662	1.4823928

are divided into the two categories

(a) $0 < \gamma^i \leq 1$ Underrelaxation

(b) $1 \leq \gamma^i$ Overrelaxation

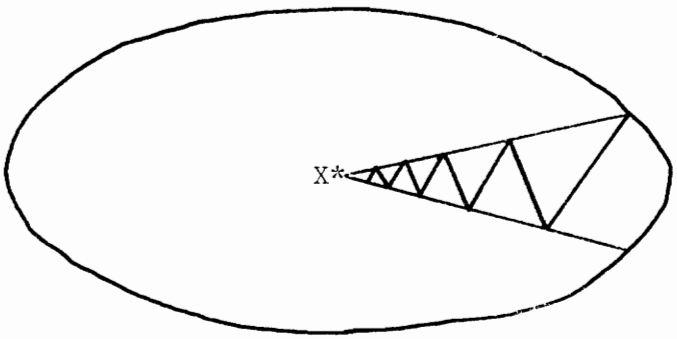
It can be shown that convergence is possible in the range $0 < \gamma^i < 2$. The intuitive argument for these and particularly for underrelaxation is that, perhaps by staying closer to the optimal trajectory discussed earlier in this chapter, convergence can be improved. Overrelaxation may improve convergence by magnifying the "jump" but this seems to be open to doubt.

Figure 7 shows the hopeful conjecture of this concept. Fadeev and Fadeeva [8] tested the procedures with $\gamma^i = .8$ and $\gamma^i = 1.2$ on the systems of equations described earlier. The results appear below comparing the number of iterations required for the same approximate reduction in error function. As can be seen, underrelation gave the best efficiency and overrelaxation the worst; but all were effective. These results seem to be typical.

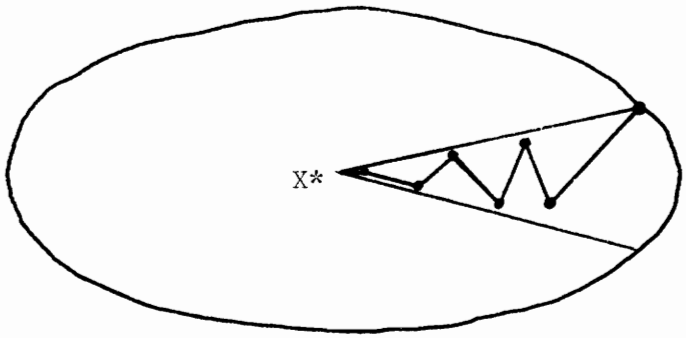
Table III. Comparison of Gradient Methods on Test Equations

Method	Number of Iterations
Underrelaxation (.8)	35
Steepest descent (1.0)	52
Overrelaxation (1.2)	62

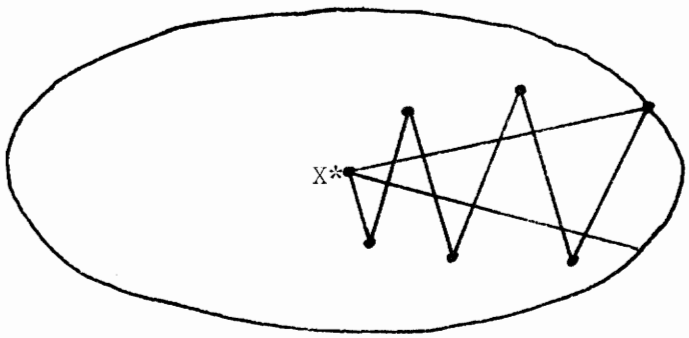
Earlier in this chapter, a method called minimal residuals was mentioned and dropped because it was to fall under a broader



Steepest Descent



Underrelaxation



Overrelaxation

Figure 7. Comparison of Gradient Methods.

classification. It falls into the underrelaxation class as it can be shown that the multiplier for this method is always less than or equal to the multiplier for the steepest descent method. It has the advantage of determining a unique step size for each iteration while underrelaxation in general does not, since a constant multiplier (e.g., .7 or .8) is used for each step.

The next logical step, due to Kantarovitch again covered by Fadeev and Fadeeva [8], is what would be the best sequence of multipliers (γ^1, γ^2, ... γ^k) so as to yield the minimum error after the k steps. The simplest procedure is to form simultaneous linear equations with the multipliers being the only unknowns; k simultaneous linear equations in k unknowns for a k step procedure. This would yield unique values of the multipliers so that the best results could be obtained. This was done on the test problem with a two-step method and very favorable results. After four iterations the following results were obtained as shown by Fadeev and Fadeeva [8].

$$X^4 = [-1.2577931 \quad .0434873 \quad 1.0391657 \quad 1.4823923].$$

This compares with the results given in Table III for the other methods. The concept then seems to have considerable promise but the problem of determining the multipliers seems to add considerably to the mathematical difficulty.

Ghare and Turner [13] recently proposed another method for the solution of simultaneous equations that is actually a directional search method. It reacts very well to systems of near singular equations and it performs as well on non-singular equations. The

procedure is presented below. Basically there are two classes of methods to solve simultaneous equations. They are

(1) Sequential orthagonalization

(2) Directional relaxation

The methods by Gauss-Jordan, Gram Schmidt, and Cholesky fall into class (1) while those by Kantarovitch, and Hestens fall into class (2). In general, class (1) methods generate solutions in a predetermined number of computations and hence are often very efficient and effective. They do, however, accumulate round off errors. If the matrix has a low condition number, then this round off error is not too critical. It is well known, however, that systems with high condition numbers are very sensitive to round off error. A high condition number implies near singularity.

Directional relaxation methods do not accumulate errors, but in general they have a slow rate of convergence. The higher the condition number, therefore, the more attractive directional relaxation methods become. This is true in spite of the fact that convergence becomes slower with higher condition numbers. Directional relaxation methods have the tendency to make a substantial improvement in the first step and thereafter significantly smaller improvements. This happens when the starting point is chosen outside the valley. The first step then moves into the valley at or near the bottom. Now since the valley is very steep, for high condition numbers, subsequent steps would quickly reach the sides of the valley and improvements would be small. It would be desirable

to find an algorithm that would keep the same order of magnitude of improvement throughout the successive iterations rather than successively smaller improvements as in the normal residual methods. Intuitively, a step out of the valley, but in the direction such that (X*-X, X*-X) = E is reduced, would mean that the next step should carry a significant improvement as it moves back into the valley and so forth. The algorithm by Ghare and Turner does just that.

The algorithm generates an auxiliary problem BX = 0 by multiplying each equation in the original problem, AX = F, by a multiplier u_j chosen such that (U,F)/F,F) is as near zero as possible. This means that the following is true:

(1) Both problems have nearly the same condition numbers.

(2) X* is a solution for both.

(3) E is the same for both but s = (F - AX, F - AX) and k = (A(X*-X), X*-X) are different.

(4) The contour valleys for the two problems are different so a point at or near the bottom of one valley would be outside the valley for the other.

A step-by-step algorithm is given below:

Step 0: The search is started at the origin unless the origin is in or near the valley of the original. If it is, another point that is outside the valley is chosen and the origin is transferred to that point.

Step 1: Let 1 denote the m dimensional vector all elements of which are 1. 1 = (1, 1, ..., 1). Summing all the equations in the

original problem yields the hyperplane.

$$H = (1A,X) = (1,F).$$

Using the direction of steepest descent F, the point where a line from the origin in the direction F intersects H can be found as

$$X^0 + \alpha F = \alpha F | \alpha = \frac{(1, F)}{(1A,F)}.$$

Call this point $X^1 = X^0 + \alpha F$.

Step 2: Determine multipliers u_j such that (u, F) = 0.

(one solution is $u_j = n - \sum_i \frac{F_i}{F_j} \Big| F_j \neq 0$

$u_j = 1 | F_j = 0$. Create an auxiliary problem P_a by multiplying each equation j in the original problem by the multipliers u_j yielding

$$BX = \Theta.$$

Determine the hyperplane

$$\bar{H}: (1B,X) = (1,\Theta) = 0.$$

The point X^2 can be found as the point where a line from X^1 in the direction Θ intersects $\bar{H}$ as

$$X^2 = X^1 + \lambda\Theta \text{ where } \lambda = - \frac{(1B,X^1)}{(1B,\Theta)} .$$

Step 3: Transfer the origin to X^2 and return to Step 1.

Wilson's system, taken from Fadeev and Fadeeva [8], was solved using the algorithm. The matrix A of Wilson's problem has a

condition number (P number) of about 3000, so it is nearly singular. The system is given below:

$$\begin{aligned} 10x_1 + 7x_2 + 8x_3 + 7x_4 &= 32 \\ 7x_1 + 5x_2 + 6x_3 + 5x_4 &= 23 \\ 8x_1 + 6x_2 + 10x_3 + 9x_4 &= 33 \\ 7x_1 + 5x_2 + 9x_3 + 10x_4 &= 31 \end{aligned}$$

After 2 iterations (4 steps), the Ghare-Turner algorithm reduced the error to 14×10^{-8}. Gauss-Jordan inversion procedure and Gaussian elimination (Type 1 methods) reduced the error to 31.9×10^{-8} and 16.4×10^{-8} respectively. After 4 iterations of a steepest descent method, error was still .05. This shows that the method performs as well as Gauss-Jordan or Gaussian elimination and has a higher rate of convergence than the method of steepest descent. The method, therefore, seems to offer much promise for application to near singular systems.

Ghare [12] recently proposed the extension of these concepts to other mathematical functions. Before examining his work, however, to emphasize the potential savings, Rosenbrock's function is attacked with a two-step gradient technique where C_1 and C_2 (the respective multistep multipliers) are varied. The results appear in Table IV (note that $C_1 = 1$, $C_2 = 1$ is the normal method of steepest descent for mathematical functions).

Empirically, from Table IV, the following hypotheses can be proposed.

Table IV. Steepest Descent and Rosenbrock's Function

Multipliers	Number Steps	Function	x_1	x_2
$C_1 = 1.$ $C_2 = 1.$	20	.19149	1.4373	2.0674
$C_1 = .1$ $C_2 = .2$	20	4.11072	-1.0206	1.0583
$C_1 = .01$ $C_2 = .02$	20	4.26816	-1.0155	1.0767
$C_1 = 1.;$ $C_2 = .5$	20	.19137	1.4372	2.0670
$C_1 = .5;$ $C_2 = .5$	20	.02703	.8652	.7579
$C_1 = .2$ $C_2 = 1.0$	20	.0245	.8559	.7264
$C_1 = 2.0$ $C_2 = 1.0$	20	6.31381	3.5124	12.3410
$C_1 = 2.0$ $C_2 = .5$	20	1215.807	2.4435	2.4869
$C_1 = .5$ $C_2 = 2.0$	20	72.75209	1.1824	.5454
$C_1 = .75$ $C_2 = 1.0$	20	1.70164	- .3043	.0949
$C_1 = 1.0$ $C_2 = .75$	20	.00139	.9644	.9311

(1) Underrelaxation (e.g., $C_1 = .5$ $C_2 = .5$) is better than straight steepest descent ($C_1 = 1.0$; $C_2 = 1.0$).

(2) Overrelaxation (e.g., $C_1 = 2.0$ $C_2 = 1.0$) is worse than straight steepest descent.

(3) There is a combination(s) of multipliers for a multistep technique that will produce better results than straight steepest descent, underrelaxation, or overrelaxation, but the problem is determining what these multipliers are.

(e.g., $C_1 = .2$, $C_2 = 1.0$, and $C_1 = 1.0$, $C_2 = .75$ are both better than underrelaxation which is the best of the one-step methods)

It is toward the determination of the best set of multipliers mentioned in (3) that Ghare directed his attention. Many of his observations are important and necessary for the developments of this thesis so they are now covered.

First he points out that the error function minimized by Kantarovitch is quadratic so the extension of his concepts to general mathematical problems should follow directly. He proceeds then to note that for twice differentiable convex functions the following observations are true.

(1) $\frac{\delta F}{\delta t}$ along $X = X^\circ + tu$ is non decreasing.

(2) $\frac{\delta F}{\delta t}$ along $X = X^\circ + tu$ changes sign once and only once and that is at the unique minimum for that line.

(3) The convergence of a gradient procedure in the neighborhood

of an optimum is not slow if $\nabla(X)$ is relatively large.

(4) If the neighborhood is relatively flat (i.e., $\nabla(X)$ small), then $E = (X^* - X, X^* - X)$ is a better measure of error.

(5) In the steepest descent method $k(k \leq n)$ consecutive directions are linearly independent and a transformation exists such that the transformed directions are orthogonal.

(6) $\rho_k > \rho_{k+n}$ where $\rho_k = \min_{\rho \in R} F[X \quad -\rho\nabla(X^\circ)]$.

Defining multistep techniques as below, (this definition is somewhat different than his) Ghare makes further observations.

$$X^1 = X^i + \sum_{i=1}^{k} C_i \rho_i u_i$$ where ρ_i is as defined before, C_i is the step multiplier, and u_i is the ith direction.

$b = \min_{b \in R} F[X^j + b(X^1 - X^j)]$ then yields

$$X^{j+1} = X^j + b(X^1 - X^j).$$

(7) If u_1 and u_2 are orthogonal ($(u_1, u_2) = 0$) and $(X^* - X_o, u_1)$ and $(X^* - X_o, u_2)$ are both positive, there exists a direction $C_1u_1 + C_2u_2$ which is better than either of the other two.

Note $(X^* - X_o, u_1)$ being positive simply says that u_1 is an improving direction.

(8) $$\max_{C_1C_2} (C_1u_1 + C_2u_2) = D_1(u_1) + D_2(u_2)$$

where $D(u) = \max \Delta E$ for direction u.

(9) The reduction in E through two successive iterations of a one step procedure cannot exceed the reduction in E through the best two step procedure.

(10) Observation 7 holds when $k > 2$ as long as all directions are orthagonal.

(11) Observation 10 holds when the directions are independent and not necessarily orthogonal but all C_i may not be positive.

(12) One k step iteration is better than k one step iterations.

(13) One k+1 step procedure is better than a k step procedure.

(14) Maximum k = n since n+1 directions would not be independent.

These observations form a basis for more research in multistep techniques. The problem confronting the researcher is of course that since $X^* - X^\circ$ is not known, $C_i (i = 1, \ldots, k)$ cannot be uniquely determined. Ghare [12] then proposes six techniques for approximating the optimum $C_i (i=1, 2, \ldots, k)$. They are the following:

(1) $C_i \alpha (\nabla(X), u_i)$.

(2) $C_i \alpha (H(X)\nabla(X), u_i) \mid H(X)$ = Hessian matrix.

(3) $C_i \alpha \rho_i \quad (\nabla(X) + \rho_i u_i), u_i) = 0$.

(4) C_i = constant.

(5) C_i = some predetermined decreasing sequence.

e.g., $\sqrt{k}, \quad \sqrt{k-1}, \ldots \quad \sqrt{1}$.

(6) $\frac{\delta}{\delta C_i} F(x + \sum_i C_i u_i) = 0$

or equivalently,

C_i chosen as $\min_{C_i} F(X + \sum_i C_i u_i)$.

Chapter II takes these six suggestions and creates twenty-two algorithms. Ten of the best of these are chosen for further examination on two well known functions. Most of the effort is expanded to get a "feel" for the operation of these methods by solving these selected problems. The test functions chosen are:

(1) A created convex quadratic function

$$f(x) - 100(x_1 - x_2)^2 + (1 - x_1)^2 + (x_2 - 2x_3)^3$$

(2) Rosenbrock's function

$$f(x) = 100(x_2 - x_1^2)^2 + 1 - x_1)^2$$

(3) Woods' function

$$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + x_3 - 1)^2 + 90(x_3^2 - x_4)^2$$

Constrained Optimization and Computational Techniques

Earlier in this chapter the mathematical programming problem was defined as:

Find $\bar{X} | \bar{X} \varepsilon \Gamma$ such that

$f(X) \geq f(\bar{X})$ for all $X \varepsilon \Gamma$.

When $\Gamma = E^n$, the problem is unconstrained or the feasible region is not bounded by constraints. The constrained problem arises when $\Gamma \neq E^n$ or when the feasible region is constrained by some binding constraints. Γ is always used in this research to designate the feasible region whether there are constraints or not.

An alternative statement of this problem could be:

Find $\bar{X}$ such that

$f(X) \geq f(\bar{X})$ for all $X \in E^n$ satisfying

$r_i(X) \leq 0 \quad i = 1, 2, \ldots, m.$

In this case $\Gamma = E^n \cap K$ if $r_i(X) \leq 0 \quad (i = 1, \ldots, m) \rightarrow X \in K$.

When X* (unconstrained optimal point) $\in K$, the constraints are not binding and hence the solution procedures for unconstrained optimization are quite adequate. (In fact even when $X^* \notin K$, unconstrained procedures can be used by incorporating the constraints into the objective function as will be seen later.) When $X^* \notin K$, some alterations must be made to the solution procedures or new ones must be developed.

When $f(X)$ and $r_i(X)$ (i=1, ..., m) are all linear and the X's are not restricted to integer values, the problem becomes a linear programming problem which has received considerable interest in the literature. The real start came in 1947 when George Danzig devised the simplex algorithm for solving the general linear programming problem. Since then, there have been many algorithms developed to capitalize on special characteristics of problems; but most revert back to the simplex in some form. Some good references in the area are Hadley [17], Gass [14], Llewellyn [23], Chung [15], Ackoff and Sassieni [1], and Hillier and Lieberman [20]. Two of the algorithms mentioned above are revised simplex, and dual simplex (see Hadley [17]).

When the objective function $f(X)$ is not linear, many problems can develop. In this case if $r_i(X) = 0$ (i=1, ..., m), it is possible

to incorporate the constraints into the objective function, known as the Lagrangian formulation, as $F(X,\lambda) = f(X) - \lambda[r(X)]$. It can be shown (Hadley [17]) that under certain conditions a solution to $F(X,\lambda)$ is also a solution to $f(X)$ so unconstrained optimization can be used on $F(X)$ to obtain the solution to the constrained problem $f(X)$. This is an underlying theme of this thesis and is covered later to a greater extent under discussion of penalty functions.

This can be expanded (since all constraints, with the correct usage of slack variables, can be specified as $r_i(X) = 0$ (i=1, 2, ...,m)) in general terms for a nonlinear programming problem. The methods of calculus then can be used to generate the Kuhn-Tucker necessary conditions for optimality. They are discussed in depth by Hadley [18] and so are not covered in detail here. These conditions can be used directly to solve some small uncomplicated problems but are not computationally efficient for most problems. The Kuhn-Tucker conditions have been very important as a basis for development of procedural algorithms, particularly in quadratic and convex programming.

When $f(X)$ is the sum of linear and quadratic forms and $r_i(X)$ (i=1, ..., m) is linear, the problem becomes a Quadratic Programming problem which has also received considerable interest in the literature. After the introduction of slack and/or surplus variables the quadratic problem can be formulated in matrix notation as

$$\min f(x) = C'X + X' DX$$

$$\text{s.t. } AX = b.$$

Since the feasible region is convex and if $f(X)$ is convex, there is no

local optima. (This of course happens when D is positive definite.) Although the problem is more complicated than a linear programming problem, there are still several algorithms available for use. Most of the quadratic programming algorithms use the Kuhn-Tucker conditions as a base and manipulate the problem to generate a X that satisfies these Kuhn-Tucker necessary conditions. Hadley [18], Kunzi, Krelle, and Oettli [22], and Gue and Thomas [17], all discuss Quadratic Programming algorithms. Some of the better known ones are those of Thiel and van de Panne, Beal, Wolfe, Frank and Wolfe, and Barankin and Dorfman.

Often $f(X)$ and $r_i(X)$ $(i=1,\ldots,m)$ are separable [i.e., $f(x_1, \ldots, x_n) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$]. Then, the original non-linear problem can be approximated by a form solvable by the simplex method of linear programming. In general this yields a solution to the approximating problem or an approximation to the non-linear problem. Only when $r_i(X)$ $(i=1, \ldots, m)$ and $f(X)$ have the proper convexity can the approximation be said to be an approximation to global optimum. Hadley [18] presents two methods for approximating solutions to nonlinear programming problems which he calls the S and δ methods.

Integer programming is a nonlinear programming problem that would be a linear programming problem if some or all the variables were not restricted to integer values. If L = set of all variables that are restricted to integer values, then the integer programming problem may be stated as:

$$\min f(X) = C'X$$

$$\text{s.t. } AX = b$$

$$\text{where } i \varepsilon L \rightarrow x_i \text{ an integer}$$

Round off can be used for problems where the values are large but even then the solutions may not be optimal.

If the integer values are limited to 0 or 1, then it is called a zero-one programming problem. These are unique problems in that they are quite different from most other N.L.P. problems, so they require specialized solution procedures. Hadley [18] discusses these problems to some depth. Branch and bound and cutting plane methods are two approaches to solving this problem.

Earlier in this chapter the gradient method for unconstrained optimization was presented. It is interesting to propose using this same method with some alterations on the general nonlinear programming problem starting with some feasible solution X°. (In the arguments to follow, "gradient direction" may be replaced at any time with any "improving direction" without loss of generality.)

As long as $X^n = X^{n-1} + \rho^{n-1}\nabla^{n-1}(X)$ is feasible (i.e., $r_i(X^n) \leq 0$ $i = 1, 2, \ldots, m$), the algorithm presented earlier applies directly. If, however, X^n defined above is not feasible, then at least one $r_i(X^n) > 0$. (For notation assume $r_i(X^n) > 0 \rightarrow i \varepsilon k$.) By some procedure, then $\lambda_i(i \varepsilon k)$ is calculated such that $r_i(X^{n-1} + \lambda_i \nabla(X^{n-1}) = 0$. ρ^{n-1} is then chosen as $\min_{i \varepsilon k} \lambda_i$ and the iterative process continues. There are various ways of determining this best step ρ^{n-1} and

Hadley [16] discusses them to some depth. It is possible that $\min_{i \in k} \lambda_i = 0$ when X^{n-1} lies on at least one active constraint. In this case, some direction other than $\nabla(X^{n-1})$ must be chosen. There are several ways of doing this, and some of the better known ones are discussed below.

One such direction is that described by J. L. Greenstadt [11] in a method called a Ricocheting Gradient method. In this method, Greenstadt proposes moving along the gradient direction until a set of constraints is violated. At this time, the method finds the point of intersection of the gradient and the constraint(s) being violated. If there is more than one constraint sufficiently close to being violated, then there must be a point "nearby" where all these constraints intersect. The method finds this point, and by means of a quadratic sub-problem generates the direction that moves along the constant function contour that forms the smallest angle with the normal to the constraint. This direction is chosen as the ricochet direction, and it is continued until another constraint is satisfied as an equality. The approximate mid-point of this line is used as the next starting point for the gradient direction and the iterative process continues.

Another method of operating near the constraints quite similar to this, but different in that it tends to move along constraints rather than away from them, is the gradient projection method of Rosen [26, 27] and discussed rather extensively by Hadley [18]. In general, the gradient projection method operates the same as

ricocheting gradient until a set of constraints is violated. The point of intersection of the last gradient and the constraint(s) is determined as before, but from this point on, the procedure differs from the others. It is possible to generate a "projection maxtrix" that "projects" the gradient direction onto the orthogonal complement of the subspace generated by the strict equalities. It projects the gradient onto the feasible region. Now if the projected direction $\neq$ 0, this can be taken as the "new direction," and movement can be generated until either another constraint is encountered or until an extreme point is found, and the iterative process continues. Now if the projected direction = 0, either the optimum has been found or one or more of the equalities needs to be relieved so that it again becomes an inequality. The method has rules that tell which, if any, of the equality constraints need to be removed to obtain a projected direction $\neq$ 0. This can also be used to determine whether the optimum has been found or not; for if the projected direction = 0 no matter which constraints are relieved, as long as they are not violated, an optimum has been found.

Rosen has had considerable computational experience and has shown in Part I [25] that, in general, on linear programming problems it reaches the optimum in fewer iterations than simplex; but it does require more time. He also observes that, for a well behaved nonlinear problem, the time required is not much greater than a linear problem of comparable size.

Zoutendijk [31] proposed the use of a subprogram at each iteration

to determine the best feasible direction in which to move. In general, this method maximizes the cosine [minimizes the angle] of the angle between the gradient direction and the chosen direction while requiring that the chosen direction lies within the feasible region. Then the step size is determined as before and the iterative process continues. The details of the procedure can vary somewhat and can become quite complicated as the method of feasible directions is, in reality, a class of methods depending upon how the procedure is carried out. This procedure can be developed so general that other methods (e.g., gradient and gradient projection) can be shown to be in this class of methods. (See Canon, Cullum, and Polak [14]). This of course is only interesting as a pedagogical tool for the methods are indeed quite different.

Recently a type of approach to solutions called Dynamic Programming has appeared that is computationally very efficient for some limited nonlinear programming problems. Hadley [18] and Nemhauser [25] are excellent references for dynamic programming. The basic ideas of dynamic programming were developed by Richard Bellman in the early 1950's while working for the Rand Corporation. The method is very efficient on functions with few linear constraints, and it is helpful if the objective function is separable as discussed earlier. Basically, the method sequentially optimizes in stages. That is, it makes stage wise decisions keeping in mind at all times previous decisions and the effect upon the next decision. Books have been written on this class of methods, so it is beyond the scope of this

thesis to present it to any depth. The method is computationally efficient in that it works very well when it does work; but its use is restricted to a limited class of problems. This research does present later an application of dynamic programming in unconstrained optimization that should increase the range of application.

The final class of methods to be considered is that in which the constraints are incorporated into the objective function to form an unconstrained problem whose solution X* is also a solution to the constrained problem. An example of this is that of the Lagrangian formulation $F(X,\lambda)$ seen earlier. There are many other methods of accomplishing this incorporation, and two of the best references are Fiacco and McCormic [9] and Beltrami [2]. This formulation is called the penalty concept as the imbedded constraints tend to "penalize" or "reward" the search such that the search terminates at the constrained optimum.

Historically, the penalty concept is due to Courant [6] who formulated the idea around 1943. His underlying theme for this development is that the constraints are indeed costs that should be charged against the objective function if they are violated. These costs then should tend to infinity so that in the limit the constraints would not be violated.

Using such an argument, the penalty function P(X) for each constraint should possess the following desirable properties:

(1) P(X) should have correct convexity for the problem considered.

(2) If X* for f(X) $\varepsilon\Gamma$, then the penalty should disappear. (P(X*) = 0.)

(3) If X* for f(X) $\notin\Gamma$ and X** is the solution to the constrained problem, then $P_i(X^*) = \infty$ in the limit for each constraint i violated.

(4) P(X) ε C^2 (This is desirable but not absolutely necessary depending upon the solution procedure employed.)

The Courant type argument generates a sequence of solutions actually outside the feasible region (when X* $\notin\Gamma$) that converge in the limit to X**. Fiacco and McCormic [9] propose an alternative method that generates a sequence of feasible points that converge in the limit to X**. Instead of penalty, they propose an actual "advantage" to being feasible; and by decreasing the multiplier of this advantage, the sequence tends to converge to X**. The advantage of this over the penalty, of course, is that the procedure can be terminated at any time with a feasible solution.

This class of methods using the penalty concept shows that any constrained problem can be formulated as an unconstrained problem so the techniques of unconstrained optimization can be applied to nonlinear programming. The major emphasis of this research, therefore, is on unconstrained techniques particularly multi-step concepts. Through the use of penalty formulation, these results can be applied to nonlinear programming. It is interesting here to note that as Ghare [12] says:

"Since every constrained problem can be transformed into

an equivalent unconstrained problem, constrained problems are also within the reach of gradient methods."

This next chapter deals with the development of multistep algorithms for the unconstrained problem. The algorithms are presented, discussed, and tried on the three functions described earlier. Finally, comparisons are made of these multistep algorithms with some of the more classical methods to show that multistep concepts have been used in the past in various ways. It is important to remember that in the previous and forthcoming conceptual arguments the terms "gradient direction" can mean any improving direction (e.g., projected gradient, feasible direction, coordinate directions, etc.)

CHAPTER II

MULTISTEP DIRECTIONAL METHODS DEVELOPMENTS AND APPLICATIONS

Introduction, One-Step Methods

Chapter I states that since any constrained problem can be formulated as an unconstrained problem, the methods of unconstrained optimization are quite interesting and applicable. In particular, Chapter I proposes the use of multistep directional techniques, which are basically unconstrained methods, in constrained optimization. This chapter concentrates on examining some potential multistep directional procedures for unconstrained optimization. The six suggestions of Ghare are explored one by one so that new procedures are developed and tried. These are then compared with some well known classical procedures. As a basis for practical comparison, the three variable, convex test function described earlier is used.

The basic form of a multistep procedure as described earlier is used and is repeated below:

$$X' = X^j + \sum_{i=1}^{k} C_i \rho_i u_i \quad \text{where} \quad \rho_i = \text{step length to an optimum,}$$

C_i = step multiplier, and

u_i = direction.

then $$b = \min_{b \in R} F[X + b(X' - X^j)]$$

yielding $$X^{j+1} = X^j + b(X' - X^j).$$

To get a handle on the efficiency and effectiveness of the

various methods to be examined the test function was first attacked by these three one step gradient procedures:

(1) Underrelaxation (.8).

(2) Complete relaxation (gradient).

(3) Overrelaxation (1.2).

The results of these trials are presented in Table V with all times corrected as closely as possible to IBM 360 Model 65, this will be done in all cases. In all tables to follow the entries under Function are the value of the function at the termination point and the entries under Error are $(X^* - X, X^* - X)$ at the termination point. The numbers under Comp. and Exec. are the compilation and execution times respectively. In general, under relation seems to offer promise while complete and overrelaxation seem to be effective but more inefficient. This seems to follow the pattern that was demonstrated in Chapter I on the solution of simultaneous linear equations.

Multistep Concept I

Now to proceed to examine the six proposals. The first suggestion was to determine C_i given that $C_i \; \alpha(\nabla^i(X), U_i)$ or the individual step lengths are proportional to the scalar product of the gradient and the chosen direction. Two candidates for U_i are immediately apparent:

(1) $U_i = -\nabla^i(X)$.

(2) $U_i = \Delta X \mid \delta X^j = 0$ for all $j \neq i$, $\delta X^i = \alpha$

where α is a step size.

The first is the direction of steepest descent and the second is a

coordinate direction. The intuitive appeal is simply that if a direction is an improvement then the amount of the improvement should help determine the individual step multipliers.

Taking the direction of steepest descent first, the idea can be reformulated as:

$$C_i \alpha \beta || \nabla^i(X) ||$$

where $\nabla^i(X)$ is the gradient at the ith point and β is the constant necessary to obtain the proper range of step. If ρ is defined (as before) as the step length to an optimum, it is logical to alter the range of the step so as to fall "close" to ρ. The step multipliers, therefore, are defined as:

$$C_i = \beta\rho^i ||\nabla^i(X)|| \quad i = 1, \ldots, k$$

where $\beta = \dfrac{1}{||\nabla^1(X)||}$.

This means that the first step length will always be ρ^1 and all others ($i = 2, \ldots, 1$) will be "close" to ρ^i but in general will be somewhat less as $||\nabla^i(X)||$ in general should decrease as i gets larger.

The gradient algorithm as presented in Chapter I in the first section can be used with the following modifications:

Step (3) Determine the new point $X^{i+1} = X^i - C^i\rho^i\nabla^i(X)$. Test for optimality. If $X^{i+1} = 1^*$, then STOP. If not, set $i = i+1$. If i greater than k, go to 4. If not, calculate the new gradient $\nabla^i(X)$ and go to 2.

Step (4) Set $\nabla^i(X) = X^{i+1} - X^i$. Determine ρ for this new

direction as in step 2. Determine the new point $X^{i+1} = X^i + \rho\nabla^i(X)$. Test for optimality. If $X^{i+1} = X^*$, STOP. If not, set i = i+1, determine the new gradient $\nabla^i(X)$, and go to 2.

The test function was run using the above algorithm with direct differentiation on ρ, since the function is convex. The results are given in Table VI. Note k = 3 is the maximum since N = 3 and k = 1 would only yield a one-step steepest descent method every two steps. So it is left out. Note that again all times are corrected as closely as possible to an IBM 360 Model 65. In comparison with single step methods, k = 2 is better than the gradient or overrelaxation methods; but not as good as underrelaxation. It was expected that k = 3 would yield a better result than k = 2 but a closer examination of the three directions shows that U_2 is orthogonal to U_1 since $C_1 = 1$; but U_3 is not in general orthagonal to U_2; hence the orthogonality as discussed by Ghare [12] in his observation 7 seems to be quite important. It should be worthwhile to watch for this in further experimentation.

Next, the coordinate axes are chosen as the direction as described earlier in this section. This procedure is a modified pattern search where the incremental step lengths are proportional to the partial derivative of the function in that direction. The procedure may be defined as below:

$$\delta X_j = \beta\frac{\delta f(X)}{\delta X_j}, \quad \delta X^i = 0 \quad i \neq j,$$

or in matrix notation,

$$\Delta X^i = [0, 0, \ldots \frac{\beta \delta f(X)}{\delta X_j}, \ldots 0].$$

Now the problem arises as to how to determine k. k = n is the most obvious since then all variables are examined. The test function with a constant step size and k=2 was run, however, to see how this affects the results. Naturally, the variables had to alternate (e.g., the sequence used was 1-2, 2-3, 3-1, 1-2, ...). After 1000 iterations, the function value was still .3033423 and the error was .6825932. It will be seen later that this is much worse than k=n (Table XVIII) which terminated in 214 iterations. Therefore, k < n is dropped from consideration throughout the remainder of this research for all coordinate direction algorithms.

An interesting examination would also be to experiment with k = 2N and k = 3N to see how this affects the efficiency. The expected result would of course be that k = 3N is worse than k = 2N which is worse than k = N but all would be effective. This is done for the methods to be covered.

Naturally, one step size would have to be fixed and the others calculated. With this in mind, ΔX^1 is chosen to be .00141424 (for reasons of consistency as will be seen later) so that β is uniquely determined as

$$\left(\frac{\delta f(X)}{\delta X_1}\right)^{-1} (.001424).$$

The pattern search algorithm of the first section of Chapter I

can be used with the δ_i as defined above and the following other modification.

Step (6) $X^{j+1^1} = X^j + \rho\ (X^{j+1} - X^j)$

where ρ is determined as the step length to an optimum as before. The rest of step 6 is the same.

To summarize, the only differences in the mechanics are:

(1) δX_i is proportional to $\frac{\delta f(X)}{\delta X_i}$,

(2) The acceleration step now searches for an optimum instead of just doubling.

The results of this method appear in Table VII.

Here k = N and k = 3N kicked out too soon for reduction of step size going too far; but both were being effective. Surprisingly, k = 2N yields the best results. There seems to be no analytical reasoning for this except that, by sheer coincidence, k = N seems to have gotten "hung up" on a ridge and flounders around somewhat. A look at the actual output shows that this does seem to be true as the step size is cut frequently in the procedure. This should be noted, therefore, and watched as further results are presented.

An important part of this chapter in the discussion of the proposed multistep procedures is to compare these proposed procedures with some similar classical ones. For this first proposal a search of the literature fails to find any well known procedure with properties similar to this. Most of the other five proposals, however, are similar to one or more classical methods.

Table V. Computational Results of One Step Procedures - Test Function

Method	Function	Error	Comp.	Exec.	# iterations
Underrelaxation	.000030600	.000041414	.11	.58	163
Gradient	.000341668	.000780943	.13	3.55	1000
Overrelaxation	.001992707	.004544236	.12	3.59	1000

Table VI. Computational Results of $C_i\alpha(\nabla_i(X)_1, \nabla_i(X))$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.00000205	.000004689	.28	1.94	411
3	.000701625	.001604872	.22	4.05	1000

Table VII. Computational Results of $\delta X^i\ \alpha(\nabla^i(X), \nabla X)$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
N	.06936478	.156251	.40	5.67	917
2N	.001083870	.000067224	.46	1.14	159
3N	.002006149	.00451941	.41	1.11	162

Multistep Concept II

The next proposal is $C_i\ \alpha(H^{-1}(X)\nabla(X), U_i)$ which says the individual step lengths are proportional to the scalar product of $H^{-1}(X)\nabla X$ and the direction chosen. Once again, the candidates for U_i are:

(1) $U_i = -\nabla^i(X)$.

(2) $U_i = \Delta X \mid \delta X_j = 0 \quad j \neq i \quad \delta X_i = \alpha$

where α is a step size.

The hessian of the test function was shown earlier to be:

$$\begin{vmatrix} 202 & -200 & 0 \\ -200 & 202 & -4 \\ 0 & -4 & 8 \end{vmatrix}$$

By the cofactor expansion method $H^{-1}(X)$ can be developed as:

$$\begin{vmatrix} .5 & .5 & .25 \\ .5 & .505 & .2525 \\ .25 & .2525 & .25125 \end{vmatrix}$$

Since the test function is quadratic, this is quite easy to obtain; but for a non quadratic function, the hessian would change with each iteration so an inverse routine would have to be an integral part of this procedure.

It is important to note that for a quadratic function the step given by $-H^{-1}(X)\nabla^o(X)$ would go directly to the optimum in one iteration, so one algorithm could be to impose a quadratic approximation

approximation at each iteration and find the optimum for that approximation, etc. That is not a multistep concept, however, and is not explored here.

Choosing $U^i = -\nabla^i(X)$ as the first to examine, the objective is to have

$$X^{i+1} = X^i + \beta(H^{-1}(X)\nabla^i(X))\rho^i(\nabla^i(X)) \qquad i = 0, 1, 2, \ldots, k$$

for k steps, then a search along $X^k - X^\circ$ to find the optimum in that direction. The gradient algorithm developed in this chapter under Concept I then can be used. The results on the test function are given in Table VIII.

Of course, k = 1 would yield in two steps a one step method of steepest descent so it is not explored. Also, again, U_2 is orthogonal to U_1 but U_3 is not orthogonal to U_2; so from previous observations, k = 2 should be a better method which indeed it is. Here is further reinforcement of the desirability for orthogonal directions. Also, the method is very efficient reaching optimum in 37 iterations; but for a non-quadratic problem this would involve 37 inverse calculations.

What then would happen if $H^{-1}(X)$ were replaced by H(X) and the method run just as before? The expected result would be that it should perform quite well here as H(X) and $H^{-1}(X)$ are of the same order for a quadratic problem. But maybe not so for a non-quadratic problem.

The results using this method with H(X) are given in Table IX.

The expected results are therefore true. The method is fairly efficient and k = 2 is once again the better of the two due perhaps to the orthogonality.

Since H(X) here is used in place of $H^{-1}(X)$, not all steps are improving; so what would happen if an upper bound of 1 were placed on $C_i (C_i \leq 1)$? The results are shown in Table X. They are highly promising as k = 2 is again quite efficient; but this time nearly as efficient as when using $H^{-1}(X)$. For this quadratic problem, H(X) can be used in place of $H^{-1}(X)$ with much success. This should be explored in greater depth. Finally once again k = 2 is better than k = 3 and U_2 is orthagonal to U_1 but U_3 is not orthagonal to U_2 so here is further empirical evidence that the orthagonality is important.

Now, if U_i is chosen to be the coordinate direction, the objective is to run a modified pattern search where δX^i =

$$\beta \frac{|A_i|}{|A_1|} \quad A_i = \text{ith row of A, and}$$

$$A = H^{-1}(X)\nabla^i(X).$$

The Pattern Search algorithm developed in this chapter under Concept I can be run with the modification regarding δX^i. The results for this on the test function algorithm are shown in Table XI.

The results are interesting as the procedure is in general effective and efficient, in terms of number of iterations. Some problems do exist, however. For example, in experimental optimization where pattern search is quite effective, it would be necessary

Table VIII. Computational Results of $C_i\alpha(H^{-1}(X)\nabla^i(X), \nabla^i(X))$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.000038813	.00044977	.23	.11	37
3	.000044231	.000094542	.21	.74	146

Table IX. Computational Results of $C_1\alpha$ $(H(X)\nabla^i(X),-\nabla^i(X))$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.000013131	.00002995	.43	.99	217
3	.000043539	.00009965	.23	4.72	987

Table X. Computational Results of $C_i\alpha$ $(H(X)\nabla^i(X),-\nabla^i(X))$ $C_i \leq 1$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.000071196	.000093859	.34	.24	45
3	.00043668	.00099872	.41	4.85	969

to approximate the hessian at each iteration which would be difficult and costly. Also it would be necessary at each iteration to generate an inverse. It is important to note here that k = N is the best while k = 2N and k = 3N get progressively worse; but all are effective. This is what would be expected as pointed out earlier in this chapter and should be true except for coincidence where the method may bog down in a valley.

Now, instead of generating an inverse at each iteration, what would happen if $H^{-1}(X)$ were replaced by H(X) as in the steepest descent method? As before, this should perform fairly well on the quadratic test function but questionably on non-quadratic functions. The results of doing this on the test function appear in Table XII.

These results are very promising and quite surprising, K = N and k = 3N are better than with the inverse, while k = 2N is worse. Perhaps the conclusion can be drawn that H(X) is a "good" substitution for $H^{-1}(X)$ for this quadratic problem. The results suggest also that perhaps it is necessary to adjust the test steps according to the potential for improvement but that the specific method of doing this is not too important. Also, once again k = N is the best k = 2N and k = 3N become progressively worse which supports the expected results.

A quick look at the output programs shows that the method using H(X) has the quickest initial convergence and the one using $H^{-1}(X)$ has the best latter convergence. A program was tried therefore where H(X) was used in the first ten iterations and then $H^{-1}(X)$. The

results appear in Table XIII. The results of this are also interesting but there just isn't enough sound evidence to make a judgement as much of these differences could be due to the effect of valleys or some other unknown reason. This should be explored in more depth.

Now, all pattern search methods using the hessian in some form or another have one similar property. They all tend to improve slowly at first, apparently establishing a pattern, then they "jump" a very large amount straight to the optimum. This is an intriguing property; and if it could be understood, perhaps some use could be made of it as an ending routine of another search procedure.

It is interesting to compare these concepts with a well known method developed by Fletcher and Powell [10] and discussed by Wilde and Beighter [30], the deflected gradient method. A detailed procedure is not presented as the reference presents a thorough analysis. The procedure in general capitalizes on the fact that the optimum step for a quadratic function is $-H^{-1}(X)\nabla(X)$ which, as stated before, goes straight to the optimum in one iteration. The procedure generates better and better estimates of $H^{-1}(X)$ by information at each iteration until at the nth stage (n+1 sometimes because of roundoff error) $H^{-1}(X)$ is generated so that $X^N = X^*$ for convex quadratic problems. The procedure on non quadratic problems just keeps operating until $\nabla^k(X) = 0$ so that $X^k = X^*$ $(k > N)$. After the first N iterations, this is the successive quadratic approximations mentioned in this chapter on page 56.

The multistep concepts developed by consideration of $H^{-1}(X)$ are

Table XI. Computational Results of δX^i $\alpha(H^{-1}(X)\nabla^i(X),\Delta X)$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
N	.00033263	.000002979	.42	.85	146
2N	.00006604	.00002086	.43	1.14	207
3N	.00011417	.00001491	.46	2.06	260

Table XII. Computational Results of δX_i $\alpha(H(X)\nabla^i(X),\Delta X)$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
N	.00041545	.000093492	.44	.53	80
2N	.000021875	.000032402	.53	1.51	213
3N	.000035912	.000080737	.40	1.35	223

Table XIII. Computational Results of $\delta_i\alpha(H(X)\nabla(X),\Delta X)$ then $(H^{-1}(X)\Delta(X),\Delta X)$

k	Function	Error	Comp.	Exec.	# iterations
N	.00143657	.000001080	.49	.72	118
2N	Not	Run			
3N	.000005465	.00000827	.46	.74	108

quite similar to the deflected gradient method. The deflected gradient chooses a direction other than the steepest descent by premultiplying the gradient by $H^{-1}(X)$ at each iteration. The multistep techniques attempt to choose one step size in the improving direction ($-\nabla(X)$ or ΔX) by premultiplying by $H^{-1}(X)$. The motivating power behind both then (the use of the second order terms to improve convergence) is the same.

Multistep Concept III

The next proposal was $C_i \alpha a_i$ where a_i is determined by

$$(\nabla(X+a_iU_i),U_i) = 0.$$

This, in general, says that C_i is proportional to that step size necessary to make the i+1st gradient orthagonal to the ith direction. This, of course, is the same as choosing a_i such that $f(X+a_iU_i)$ is minimized (X^{i+1} is the minimum point along the direction U_i from X^i). Since all other methods choose their step sizes in relation to this a_i, which is the same as ρ, this is really not a new method but there are some unique ideas generated.

The two candidate directions are, once again, the direction of steepest descent and the coordinate directions. The later is a multistep adaptation of one-at-a-time search and is not explored here as the literature treats this sectioning method quite well. Perhaps a further study should be made on using sectioning with different degrees of relaxation from a multistep standpoint.

The direction of steepest descent does offer some possibilities. First, suppose $C_i = 1$, $i = 1, 2, \ldots, k$ or a complete relaxation for k steps and then a search along the pattern established. The results of this procedure on the test function appear in Table XIV. The results are very encouraging since, without any highly theoretical developments, the optimum can be reached in 42 iterations. This is dramatically better than the straight gradient procedure and even much better than underrelaxation. It would have been expected, however, that k = 3 would be better than k = 2 but the results show differently. This could be due to the fact that, for a quadratic problem, the search is somewhat planar which would mean k = 2 would yield a very good direction.

Next, underrelaxation is tried for k steps and then a search (i.e., $C_i = .8$ $i = 1, 2, \ldots, k$). The results on the test function appear in Table XV. These results are also interesting, but not significantly different from complete relaxation (k = 1 is worse but k = 2 is better). The whole reasoning behind underrelaxation was to develop a technique where a pattern could be established to yield better points than complete relaxation. By incorporating multistep concepts into complete relaxation, the pattern is automatically developed. This could be the reason for the lack of improvement. Once again, k = 2 is better than k = 3 but the earlier developed rule of orthogonality or the planar search pattern seems to hold here.

Using this same reasoning, it could be expected that overrelaxation for k steps and then a search would yield no better results and

perhaps worse as the search deviates greatly from the optimal trajectories. The results of multistep overrelaxation on the test function appear in Table XVI. The results are as expected except, once again, k = 2 is better than k = 3; but U_3 is not orthogonal to U_2 and U_2 is orthogonal to U_1 so the previously observed relations could hold. Also, the search is somewhat planar as mentioned earlier.

Finally, underrelaxation for k-1 then complete relaxation then search is tried (i.e., $C_i = .8$ i = 1, 2, ..., k-1; $C_k = 1$). The results are given in Table XVII. The results are not startling as no significant difference can be detected. Perhaps, then, the actual individual step lengths of a multistep method cannot be estimated to substantially improve the procedure. There is very little evidence to support this yet but it bears watching.

This concept is quite similar to one developed by Shah, Buehler, and Kempthorne [29] that reaches the optimum of a quasi quadratic function in 2N-1 iterations. The method is called "Parallel Tangents" and is discussed quite extensively by Wilde and Beightler [30]. Their method also can use any direction as a starting point, as long as it is not within the tangent plane. It also involves finding minimum points of given directions and accelerating once an optimum is found.

A gradient partan procedure generates the same first five points as a 2 step complete relaxation multistep technique as described above. From there on, gradient partan generates three more points and searches between the first and the fourth very similar to a 3 step complete relaxation multistep technique. The main difference is in

Table XIV. Computational Results of $C_i \alpha\ a_i \mid (\nabla(X+a_iU_i),U_i) = 0$ on Test Function (Complete Relaxation)

k	Function	Error	Comp.	Exec.	# iterations
2	.000011199	.000024499	.18	.14	42
3	.000039537	.000090452	.23	.84	204

Table XV. Computational Results of $C_i\ \alpha\ a_i \mid (\nabla X+a_iU_i),U_i) = 0$ on Test Function (Underrelaxation)

k	Function	Error	Comp.	Exec.	# iterations
2	.000017246	.00002738	.18	.19	74
3	.000023592	.000034621	.18	.48	148

Table XVI. Computational Results of $C_i\ \alpha\ a_i \mid (\nabla(X+a_iU_i),U_i) = 0$ on Test Function (Overrelaxation)

k	Function	Error	Comp.	Exec.	# iterations
2	.000035053	.00007716	.21	.73	171
3	.005054901	.01155411	.22	3.42	1000

the manner in which the directions are chosen.

The last concept considered in Tables XIV-XVII also bears a striking resemblance to the conjugate directions methods developed and discussed by Fletcher and Powell [10], Hestens and Stiefel [19], Beveridge and Schechter [3], and others. Particularly, the conjugate gradients are of interest as the gradient direction is one of the directions chosen for consideration in this research. The methods are covered quite well in the literature so detailed procedures are not presented. It is sufficient to say that M^j (direction) is chosen by consideration of $\nabla^j(X)$ and M^{j-1} such that M^j is conjugate to M^{j-1-i} $i = 0, 1, \ldots, j-1$ $j \leq N$. ($j = N \rightarrow$ the procedure starts over). The way this is done is

$$m_i^j = -\frac{\delta f^j}{\delta x_i} + \sum_{\ell=1}^{j-1} \beta^\ell m_i^\ell .$$

It is very obvious, therefore, that conjugate gradients is a multistep methods or, at least, they are very similar. Specifically, the jth direction is always chosen as a vector combination of j-1 directions (where max(j-1) = N). Since this combination is not a straight sum, incomplete relaxation of the j-1 directions is used.

Multistep Concept IV

The fourth proposal is C_i = constant, or the step size multiplier for each iteration is the same. The candidates for U_i are once again the direction of steepest descent and the coordinate

direction. The direction for steepest descent was covered in the previous discussions as:

(1) Complete relaxation for k steps ($C_i = 1.0$),

(2) Underrelaxation for k steps $C_i = .8$), and

(3) Overrelaxation for k steps ($C_i = 1.2$).

These three categories exhaust the possibilities for C_i = constant so no more needs to be said.

The other candidate for U_i (the coordinate directions) leads directly to an altered pattern search procedure identical to that presented earlier where a search is performed over the pattern established direction instead of simply doubling. Here $\delta_i = h$ (some constant) and the pattern search algorithm in Concept I of this chapter directly applies. ($\delta_i = .001414214$ for the reason of consistency as before). The results of this method on the test function appear in Table XVIII.

The results are very promising but not as good as some previous pattern search type methods. (e.g., $\delta_i \alpha A_i \quad A = (H^{-1}(X)\nabla(X))$. In general, however, the method does offer some potential so some further research would be justified. The next step would be to compare this with the pattern search method to see just how much better searching along the direction is than simply doubling.

As stated throughout this section, δ_i = constant is identical to the pattern search method as developed by Hooke and Jeeves [21] and discussed by Wilde and Beighter [30] except for the acceleration apparatus. The only difference between the two methods is that pattern

search doubles the move while this method searches for the optimum along $(X^n - X^\circ)$

Multistep Concept V

The fifth proposal is $C_i \alpha$ some decreasing sequence (i.e., $C_1 > C_2 > \ldots > C_{k-1} > C_k$). Once again the candidates for u_i are the direction of steepest descent and the coordinate directions.

Considering first the direction of steepest descent, the multipliers are defined as

$$C_i = \sqrt{k-I+1} \; / \; \sqrt{k}.$$

This means that the multiplier starts at 1 for i - 1 and monotonically decreases to $1/\sqrt{k}$ for i = k. The expected results would be that the method shows promise and that k = I is a better method than k = I-1 for all $I \geq N$. The performance on the test function is shown in Table XIX. As expected, k = 3 is better than k = 2 and the method is fairly successful. When compared against the complete relaxation multistep method, however, the efficiency is not particularly outstanding. (Number iterations = 42 for k = 2 on complete relaxation.) The extra effort involved in calculating the sequence does not seem to be worthwhile. It does compare very favorable with the best one step method, however. (Number iterations = 163 for underrelaxation.)

The entire concept of multistep theory is to break any slow converging pattern, so to offset some of the additional calculations maybe ρ could be assumed to be constant for the k steps instead of

Table XVII. Computational Results of $C_i = .8$ $i = 1, 2, \ldots, k-1$; $C_k = 1$ on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.000037943	.000084466	.31	.31	75
3	.000147705	.000007455	.22	.41	124

Table XVIII. Computational Results of δ_i = constant Test Function

k	Function	Error	Comp.	Exec.	# iterations
N	.000035488	.00003572	.36	1.09	214
2N	.000000059	.00000003	.34	1.79	280
3N	.00049605	.000045133	.44	2.39	352

Table XIX. Computational Results of C_i α Decreasing Sequence on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.00003704	.000081718	.29	.58	144
3	.000029914	.000048684	.31	.56	124

being calculated each time. The results of this on the test function using this method are shown in Table XX. The output is somewhat exciting. Apparently the estimation for ρ does not hold two iterations in the future, but does fairly well for one additional iteration. (285 iterations compared to 144). In experimental searches where an expensive Fibonacci, or similar one variable search, must be performed each time, maybe it would be advantageous to assume constant ρ for two iterations. This should be explored further.

An interesting offshoot of this line of reasoning is to consider ranking the variables in order of decreasing magnitude of partials (i.e., $(x_1, x_2, \ldots, x_n) \rightarrow (|\frac{\delta f}{\delta x_1}| > |\frac{\delta f}{\delta x_2}| > \ldots > |\frac{\delta f}{\delta x_n}|)$ and take decreasing step sizes for this sequence of variables within each iteration. [This is really a pattern search technique where the gradient direction is used instead of coordinate.] The hopeful conjecture is that for each variable a larger partial should necessitate a larger step and vice versa.

The mechanics of this would be similar to the gradient algorithm used earlier in this section with the following changes:

(1) Statement 2 must first rerank all the variables in terms of decreasing absolute values of the partials.

(2) The sequence must operate on each variable within an iteration rather than between iterations.

The results of this method on the test function are given in Table XXI. The algorithm is quite efficient but a quick look at Table XIX shows there is at least no improvement over the simple Decreasing Sequence

concept, hence, the ranking appears superfluous.

Finally, the method in Table XXI is run again this time with ρ calculated the first step only and then assumed constant for k steps as before (Table XX). The results appear in Table XXII. Once again this seems promising for one iteration in the future (k = 2 - 213 iterations versus 150) but not for two iterations (k = 3 - 936 iterations versus 132). It does appear to offer some promises for experimental searches as mentioned earlier in this chapter.

In considering the coordinate directions, it is necessary to rank the variables at each iteration to obtain the order for the decreasing sequence or else the last variable in original ranking would have extremely slow convergence. The pattern search algorithm as developed in Concept I of this chapter can therefore be used with the following changes:

(1) $\delta_i = \sqrt{N-I+1}\ (.001)$

(Note that this is the reason for establishing all pattern search increments so that the average step size is .001424 which is δ_2 above.)

(2) Step 2 of the algorithm must first include a reranking of the variables for each iteration.

The results appear in Table XXIII.

The method appears to be quite efficient and offer some potential; but when compared with the constant δ_i given in Table XVIII (214, 280, 352 iterations respectively), there is some question as to the benefit of ranking the variables.

It is worthwhile to note here that in comparing this concept with

Table XX. Computational Results of $C_i\alpha$ Decreasing Sequence - ρ Held on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.000000222	.000000066	.24	1.37	285
3	.00052619	.001200851	.26	4.32	1000

Table XXI. Computational Results of $C_i\alpha$ Decreasing Sequence - Ranked Variables on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.00006133	.00014029	.32	.87	150
3	.00001253	.00000654	.34	.94	132

Table XXII. Computational Results of $C_i\alpha$ Decreasing Sequence - Ranked Variables - ρ Held on Test Function

k	Function	Error	Comp.	Exec.	# iterations
2	.00017793	.00001159	.39	1.54	213
3	.00004398	.00007778	.31	6.36	936

Table XXIII. Computational Results of $\delta_i \alpha$ Decreasing Sequence on Test Function

k	Function	Error	Comp.	Exec.	# iterations
N	.0000517	.000090536	.48	1.89	241
2N	.000112688	.000023880	.36	1.79	213
3N	.000038097	.000084317	.41	2.31	312

the straight pattern search method (doubling instead of searching), after 100 iterations on $F = 100\ (x_1 - x_2)^2 + x_3^2$, this method yielded a functional value of .01123 while pattern search yielded .2500. Of course the question arises as to whether this is due to the searching or the ranking. This should be explored further.

This would be a good place to evaluate the concept of holding ρ constant for 2 iterations. Pattern search is used extensively in experimental problems and this paper proposes searching instead of doubling. An empirical study could be made on the closeness of the approximation by comparing the calculated ρ's of functions like in Table XXIII with calculating every other ρ on the same function. Finally a review of the literature reveals no classical method with similar properties.

Multistep Concept VI

The sixth proposal is to determine C_i by solving

$$\frac{\delta}{\delta C_i} F(X + \sum_i C_i u_i) = 0$$

or, in general, to determine $C_i (i = 1, 2, \ldots, k)$ so that $F(X^k)$ is minimized. The individual step multipliers, therefore, are chosen not to minimize $F(X^i)$ but to minimize $F(X^k)$ the final function value. It would be expected that this method would be highly efficient and effective provided some method of efficiently determining C_i can be found.

The problem comes of course in determining C_i. The expression

of the above equations would be difficult and algebraically quite complicated. If, on the other hand, a multivariate search over C_i is conducted on $F(X + \sum_i C_i u_i)$, the algebra is not as difficult but the procedure seems to be little better than a multivariable search over X which was the original problem. Possibly one or both of these ideas could prove beneficial and should be explored later; but there should be some way of stage wise solving for these C_i. Perhaps the concepts of dynamic programming could help. (Since C_i now is the ith decision stage, it is shown as C^i to avoid confusion with variables. In general, superscripts imply stage number and subscripts imply variable number.) Considering a k step procedure the sequence of points, not decisions, appears in Figure 8. The objective is to maximize Σr_i.

The methods of dynamic programming should apply very well with the problem being the choice of u_i and r_i, the return at stage i. Considering that the sole interest lies in minimizing $F(X^k)$, perhaps the problem would be more manageable as a terminal stage optimization problem. Redrawing Figure 8 then to show this and renumbering in the order of decisions, in the ideology of dynamic programming, is done in Figure 9.

Considering the direction of steepest descent first as a candidate for u^i, it is helpful to note that for k = 1, $C^i = \rho$ as it is the complete relaxation method. The previously found value of ρ of course is:

$$C^i = \rho = [-202x_1\nabla_1 + 200x_2\nabla_1 + 200x_1\nabla_2 - 200x_2\nabla_2 + 2\nabla_1 + 4x_3\nabla_2$$

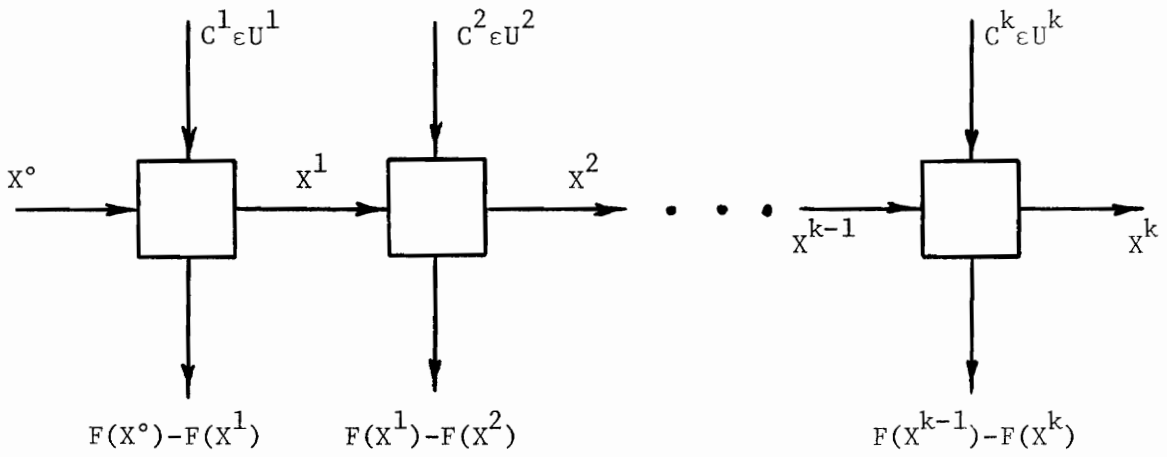


Figure 8. Stage Optimization and Multistep Theory.

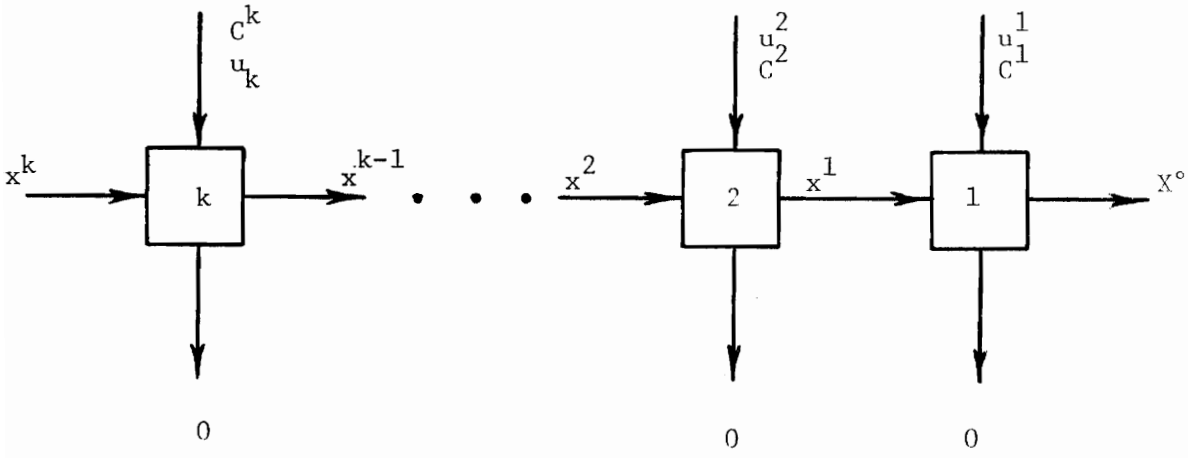


Figure 9. Dynamic Programming and Multistep Theory.

$$+ 4x_2\nabla_3 - 8x_3\nabla_3]/(202\nabla_1^2 - 400\nabla_1\nabla_2 + 202\nabla_2^2 - 8\nabla_2\nabla_3 + 8\nabla_3^2]$$

It follows then that

$$Q^1(x^1,C^i) = 0 + 100[x_1^1 - C^1\nabla_1^1 - x_2^1 + C^1\nabla_2^1]^2 + [1 - x_2^1 + C^1\nabla_2^1]^2 + [x_2^1 - C^1\nabla_2^1 - 2x_3^1 + 2C^1\nabla_3^1]^2$$

where C^1 is defined above, superscripts imply stage number and subscripts imply variable number. Without proceeding further, it can be seen that this method will get algebraically quite complicated since C^1 is quite long; and for $Q^2(X^2, C^2)$, X^1 must be replaced by $[X^2 - C^2\nabla(X)^2]$, etc. Therefore, even a two-stage problem would be very difficult but once solved quite efficient. This should be explored further.

Jumping ahead to the next candidate for u_i (the coordinate directions) it can be seen that the algebra will not be as complicated since $X^i = X^{i+1} + C^{i+1}$ or $[x_1^{i+1}, \ldots x_i^{i+1} + C^{i+1}, x_{i+1}^{i+1}, \ldots x_n^{i+1}]$. Only the ith variable undergoes transformation but similar to sectioning, the step is relatively large. Considering the test problem, Figure 10 applies.

The solution of this considering the test function is:

$$f^0(x^0) = 100(x_1^0 - x_2^0)^2 + (1 - x_2^0)^2 + (x_2^0 - 2x_3^0)^2$$

$$Q^1(X^1,C^1) = 100(x_1^1 + C^1 - x_2^1)^2 + (1 - x_2^1)^2 + (x_2^1 - 2x_3^1)^2$$

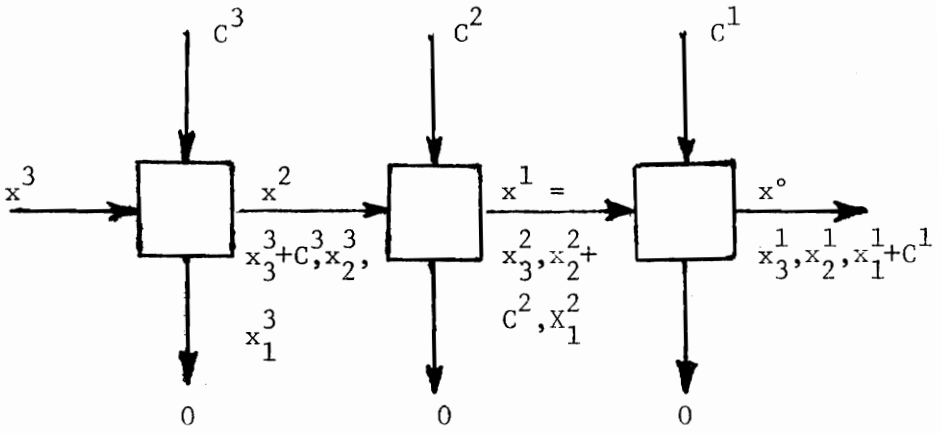


Figure 10. Dynamic Programming and Test Function

$$\frac{\delta Q^1}{\delta C^1}(X^1,C^1) = 0 = 200\ (x^1 + C^1 - x_2^1) = 0$$

$$C^1 = x_2^1 - x_1^1$$

$$f^1(X^1) = (1 - x_2^1)^2 + (x_2^1 - 2x_3^1)^2$$

$$Q^2(X^2,C^2) = (1 - x_2^2 - C^2)^2 + (x_2^2 + C^2 - 2x_3^2)^2$$

$$\frac{\delta Q^2}{\delta C^2}(X^2,\ C^2) = 1 + 2x_2^2 + 2C^2 - 2x_3^2 = 0$$

$$C^2 = x_3^2 - x_2^2 + \frac{1}{2}$$

$$f^2(X^2) = (\tfrac{1}{2} - x_3^2)^2 + (x_3^2 + \tfrac{1}{2} - 2x_3^2)^2$$

$$Q^3(X^3,C^3) = (\tfrac{1}{2} - x_3^3 - C^3)^2 + (x_3^3 + C^3 + \tfrac{1}{2} - 2x_3^3 - 2C^3)^2$$

$$\frac{\delta Q^3}{\delta C^3}(X^3,C^3) = 0 = -2x_3^3 - 2C^3 + 1 = 0$$

$$C^3 = \tfrac{1}{2} - x_3^3.$$

Since $X^N = [4,\ 4,\ 4]$,

$$C^3 = .5 - 4 = -3.5,$$

$$C^2 = .5 - 4 + .5 = -3, \text{ and}$$

$$C^1 = 1 - 4 = -3.$$

Since $x_1^3 = 4$, $x_2^3 = 4$, and $x_3^3 = 4$,

The calculated X is $x_1^0 = 1$, $x_2^0 = 1$, and $x_3^0 = .5$. This yields X* which was expected. This method, then, looks extremely promising as it leads to X* in one iteration. A quick look at the solutions for C^1, C^2, and C^3 show that $X^0 = X^*$ for any starting poing X^k.

It is outside the scope of this analysis to go much further in this area; except to note that this method is extremely promising for a quadratic case, but for a degree higher than 2 the unique solutions for C^i are more difficult to obtain than in the quadratic case. Hence, it seems beneficial to propose a series of quadratic approximations for non-quadratic problems. A procedural description of such a concept appears below:

(1) Set i = 0.

(2) Calculate H^i (the hessian matrix) and $\nabla(X^i)$ (the matrix of partial derivative). Form the quadratic approximation as:

$$y(X) = y(X^0) + (X - X^0, \nabla X_o) + \frac{1}{2} (X - X^0, H(X - X^\circ)).$$

(3) Using the D.P. procedure, determine the optimum C = $(C^1, C^2, \ldots C^k)$ that would find X* for the quadratic approximation. Test for optimum. If not, set i = i+1, X = X*, and go to 2. If X* is optimum, STOP.

This algorithm is quite simple and easy to apply but it can be simplified further by noting that X* in step 2 can be uniquely determined for n variables by formulating the quadratic approximation in general terms of H and $\nabla(X)$. [In all developments following, H_{ii} signifies the element H(I,I)]. This is done for n = 2 below:

$$y(x) = y(x^0) + C^1\nabla_1 + C^2\nabla_2 + \frac{1}{2}\,[H_{11}(C^1)^2 + 2H_{12}C^1C^2 + H_{22}(C^2)^2]$$

$$\frac{\delta y(x)}{\delta C^1} = \nabla_1 + H_{11}C^1 + H_{12}C^2 = 0$$

$$C^1 = \frac{-H_{12}C^2 - \nabla_1}{H_{11}}$$

$$f^2(C^2) = \left(\frac{-H_{12}C^2 - \nabla_1}{H_{11}}\right)\nabla_1 + C^2\nabla_2 + \frac{1}{2}\left[H_{11}\left(\frac{-H_{12}C^2 - \nabla_1}{H_{11}}\right)^2\right.$$

$$\left. + 2H_{12}\left(\frac{-H_{12}C^2 - \nabla_1}{H_{11}}\right)C^2 + H_{22}(C^2)^2\right]$$

$$\frac{\delta f^2(C^2)}{\delta C^2} = 0 = C^2\left[H_{22} - \frac{(H_{12})^2}{H_{11}}\right] - \left[\frac{H_{12}\nabla_1}{H_{11}} - \nabla_2\right]$$

$$C^2 = \left[\frac{H_{12}\nabla_1}{H_{11}} - \nabla_2\right]/\left[H_{22} - \frac{(H_{12})^2}{H_{11}}\right]$$

This means that for any 2 variable quadratic convex problem, the optimum X* can be reached in one iteration by $x_1 = x_1 + C^1$ $x_2 = x_2 + C^2$ where C^1 and C^2 are as defined above. Further, non-quadratic, non-convex problems can be approximated using this concept. Rosenbrock's function is solved later in this chapter using this method.with very good results.

Extending the general results to other values of N, the following is obtained.

N = 1 $\quad C^1 = -\nabla_1/H_{11}$.

N = 2 $\quad C^1 = (-\nabla_1 - H_{12}C^2)/H_{11}$

$$C^2 = (-\nabla_2 + H_{12}\nabla_1/H_{11})/(H_{22} - (H_{22} - (H_{12})^2/H_{11}).$$

N = 3 $\quad C^1 = (-\nabla_1 - H_{12}C^2 - H_{12}C^3/H_{11}$

$$C^2 = [-\nabla_2 + H_{12}\nabla_1/H_{11} + H_{12}H_{13}C^3/H_{11} - H_{23}C^3]/[H_{22} - (H_{12})^2/H_{11}]$$

$$C^3 = -[T\nabla_1 + S\nabla_2 + \nabla_3 + H_{11}TY + TH_{12}R + H_{12}SY + H_{13}Y + H_{22}SR + H_{23}R]/[H_{11}T^2 + 2TH_{12}S + 2H_{13}T + H_{22}S^2 + 2H_{23}S + H_{33}]$$

where $\quad Q = H_{11}H_{22} - (H_{12})^2 \quad R = (-H_{11}\nabla_2 + H_{12}\nabla_1)/Q$

$$S = (H_{12}H_{13} - H_{11}H_{23})/Q \quad Y = (-\nabla_1 - H_{12}R)/H_{11}$$

and $\quad T = [-H_{12}S - H_{13}]/H_{11}.$

N = 4 $\quad C^1 = (-\nabla_1 - C^2H_{12} - C^3H_{13} - C^4H_{14})/H_{11}$

$$C^2 = [H_{12}\nabla_1/H_{11} - \nabla_2 + C^3H_{12}H_{13}/H_{11} + C^4H_{12}H_{14}/H_{11} - C^3H_{23} - C^4H_{24}]/[H_{22} - (H_{12})^2/H_{11}].$$

It is very obvious that beyond the third stage the algebra becomes horrendous. It should be possible to develop all of this in general terms for any specified k variable problem; but, once again, the algebra quickly gets difficult. Another approach would be to make a two or three variable quadratic approximation regardless of the number of variables. The variables would have to be rotated and it would be

expected that the three variable approximation would be better than the two variable and the results would get worse as N gets larger. This is tried later on Wood's function with very good results.

Finally, it seems helpful to approach this problem from the general standpoint of unspecified N and solve for the N steps. This is started below:

$$Y(X) = Y(X^\circ) + \sum_{i=1}^{N} C^i \nabla_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} H_{ij} C^i C^j + \frac{1}{2} \sum_{i=1}^{N} H_{ii} (C^i)^2$$

By an appropriate change in the ordering of the variables, the first step can always be on variable 1, the second on variable 2, etc., which means

$$\frac{\delta Y(X)}{\delta C^1} = \nabla_i + \sum_{j=i+1}^{N} H_{ij} C^j + H_{ii} C_i = 0$$

$$C^1 = (-\nabla_1 - \sum_{j=2}^{N} H_{1j} C^j)/H_{11}$$

$$Y(X) = Y(X^\circ) + \sum_{i=2}^{N} C^i \nabla_i + \sum_{i=2}^{N} \sum_{j=3}^{N} H_{ij} C^i C^j + \frac{1}{2} \sum_{i=2}^{N} H_{ii} (C^i)^2$$

$$+ \nabla_1 (-\nabla_1 - \sum_{j=2}^{N} H_{1j} C^j)/H_{11} + \sum_{j=2}^{N} H_{1j} C^j (-\nabla_1 - \sum_{j=2}^{N} H_{1j} C^j)/$$

$$H_{11} + \frac{1}{2} H_{11} ((-\nabla_1 - \sum_{j=2}^{N} H_{1j} C^j)/H_{11})^2$$

$$\frac{\delta Y(X)}{\delta C^2} = 0 = \nabla_2 + \sum_{j=3}^{N} H_{2j} C^j + H_{22} C^2 - H_{12} \nabla_1 / H_{11} + H_{12} [-\nabla_1 / H_{11} -$$

$$H_{12}C^2/H_{11} - \sum_{j=3}^{N} H_{1j}C^j/H_{11}] - (H_{12}/H_{11})$$

$$(H_{12}C^2 + \sum_{j=3}^{N} H_{1j}C^j) + H_{11}(-\nabla_1/H_{11} - H_{12}C^2/H_{11} -$$

$$\sum_{j=3}^{N} H_{1j}C^j/H_{11})(-H_{12}/H_{11})$$

$$C^2 = (-\nabla_2 - \sum_{j=3}^{N} H_{2j}C_j + H_{12}\nabla_1/H_{11} + H_{12} \sum_{j=3}^{N} H_{1j}C^j/H_{11})/(H_{22} -$$

$$(H_{12})^2/H_{11})$$

etc. Again the algebra becomes cumbersome.

One other approach can be generated by examining this same problem using matrix notation. The approximation can then be stated as:

$$Y(X) = Y(X^\circ) + C\nabla + \frac{1}{2}(H,HC)$$

The first deviative with respect to C is

$\nabla + HC$ which it set equal to 0, yields the well known result

$$C = -H^{-1}\nabla.$$

The problem can be formulated as $HC = -\nabla$, a system of n simultaneous equations in n unknowns. Any of the methods for solution of simultaneous linear equations can then be applied to obtain this vector of increments C. In fact, it has been shown that the deflected gradient method of Fletcher and Powell [10] follows this format by

iteratively determining better and better estimates of the hessian inverse. Using this argument, maybe it would be profitable not to solve the system of linear equations completely but instead stop when a reasonable estimate of the solution is obtained. Then another quadratic approximation can be made, another estimate of the solution determined, and the iterative process continued.

As discussed earlier in this thesis, residual gradient methods inherently make a very large step toward the solution, as long as the starting point lies outside the valley of the system of equations. One step of this class of methods should, therefore, be explored as the "reasonable" estimate of the solution mentioned above. The problem, of course, would be to insure that the starting point does indeed lie outside the valley or at least alter the procedure when it does not. The method by Ghare and Turner [13] shown earlier in this thesis is of interest since it recognizes the fact that the first iteration of a residual gradient yields a "large" step and attempts to keep the steps large. Using these arguments, the following algorithm can be developed.

(1) Select the initial point X° and set $i = 0$.

(2) Evaluate $\nabla(X^i)$ and $H(X^i)$ and form the system of simultaneous equations,

$$H(X^i)\ C = -\nabla(X^i).$$

Set $C = 0$. Normalize this system and change signs where necessary to obtain $-\nabla(X^i) \geq 0$.

(3) If C = 0 is inside the valley, go to (4). If not, set $C = -\alpha\nabla(X^i)$ where

$\alpha = [1, -\nabla(X^i)]/[1H(X^i), -\nabla(X^i)]$ and go to (5).

(4) Set $C = \alpha[1H(X^i)]$ where

$\alpha = [1, -\nabla(X^i)]/[1H(X^i), 1H(X^i)]$ and go to (5)

(5) Set $X^{i+1} = X^i + C$. Evaluate X^{i+1} for optimum. If not, set $i = i+1$ and go to 2. If X^{i+1} is the optimum, STOP.

In step (3), it is necessary to determine proximity of the point to the "valley" of the system. A quick way to evaluate this is to examine the residuals. If they all have the same sign and are not close to 0, then the point lies outside the valley.

In step (3), also, if the point is outside the "valley," the normal Ghare-Turner first step is taken and the result used to increment X in step 5. If, however, it lies inside the valley, then in step (4) a different method is used. The normal direction for the summed normalized equation is given by the coefficients of $1H(X^i)$. Now since $[1H(X^i), C] = [1, -\nabla(X^i)]$ is the summed equation, substituting for C yields

$$[1H(X^i), \alpha\ 1H(X^i)] = [1, -\nabla(X^i)]$$

and

$$\alpha = \frac{[1,-\nabla(X^i)]}{[1H(X^i),1H(X^i)]} .$$

Intuitively, when the point lies within the valley, the direction given by the normal to the hyperplane is a better direction and will allow

"large" steps to be taken. To illustrate this Figure 11 depicts the procedure when the point (origin) lies inside the valley for a two dimensional problem where Figure 12 shows the procedure when the point lies outside the valley.

This concept is tried on Rosenbrock's and Wood's functions with varying degrees of success as is shown later. It is interesting to notice, however, that since the procedure solves for conditions where $\nabla(X) = 0$, saddle points and/or local optima may be expected.

All of this research into dynamic programming and quadratic approximations seemed to skim over the use of these concepts for any direction other than coordinate directions and for any functions other than quadratic functions or rather functions that can be approximated by quadratic functions. This does not mean that other directions and polynomials of degree higher than two cannot or should not be examined. The only reason these were isolated is they are easy to work with and seem to offer the greatest potential. In fact, a quadratic approximation considering $u_i = -\nabla^i(X)$ should also be an efficient method and could also be solved in general for k and N.

Next, a literature review finds that there are few classical procedures with properties similar to these proposed methods. Sectioning as mentioned earlier is similar in that it takes large suboptimization steps but it considers only one step at a time with no consideration for the effect upon other steps.

Craig and Levy [7] proposed a very efficient method that uses the direction of steepest descent and minimizes the function by

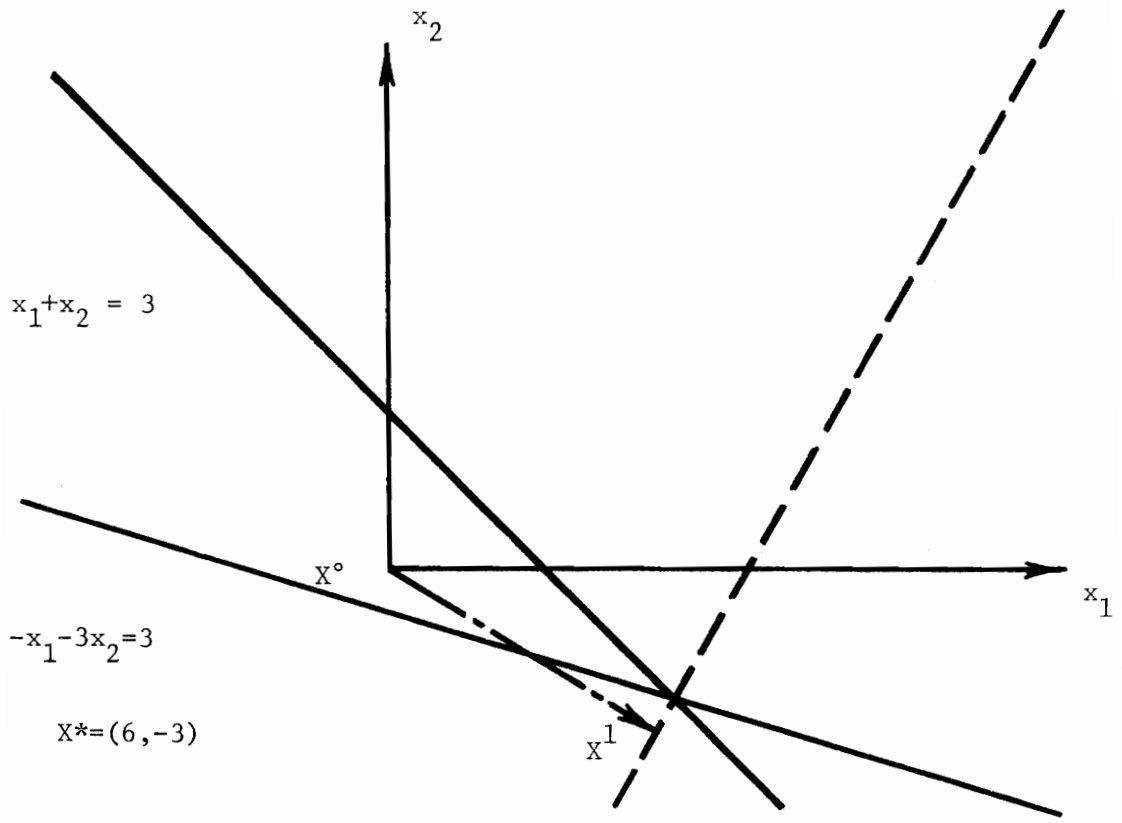


Figure 11. Solution of Simultaneous Linear Equations Using Normal Direction.

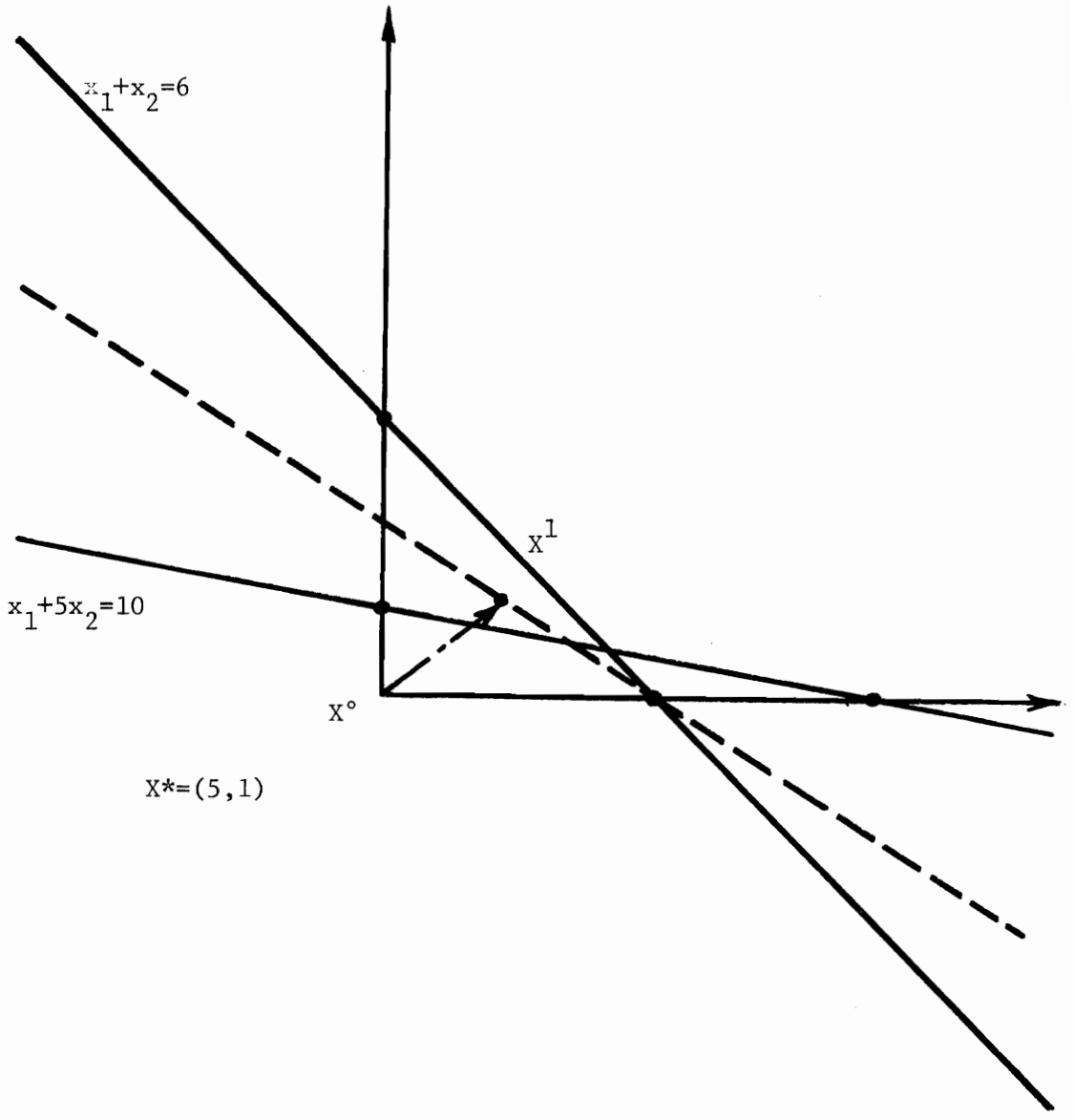


Figure 12. Solution of Simultaneous Linear Equations Using Steepest Descent.

choosing the step length that yields the minimum point considering the previous k steps. It is, therefore, called the super memory gradient and chooses C such that the expression

$$F[X + \sum_{i=1}^{k} C^i \Delta X^i + C^k(-\nabla(X^k)]$$

is minimized. The algorithm solved Wood's function in 6 iterations, which is quite efficient; but an iteration here is very long and is not comparable to an iteration as counted in this work.

In essence, "super memory gradient" does the same thing as a multistep approach. They both attempt to choose the step lengths such that $F(X^k)$ is minimized. It should be interesting to see how some of the multistep concepts developed here work on functions such as Wood's and Rosenbrock's so comparisons can be made.

This multistep concept is similar to another method in an entirely different field. That is Kantarovitch's multistep residual method for simultaneous linear equations described earlier in Chapter I. In fact, Kantavarovitch's work provided the motivation for this dynamic programming approach. Both concepts attempt to determine k multipliers simultaneously so as to yield minimum $f(X^k)$. The procedures for determining the multipliers are somewhat different but the objective is the same.

The Deflected Gradient method of Fletcher and Powell has been mentioned several times as being similar. In this case, both sequentially make quadratic approximations. The only difference lies in how the solution to the approximations is calculated. Smith [32] also proposed a similar technique that makes successive quadratic approximations.

Multistep Performance on Other Functions

All the previous sections have been developing many different algorithms and trying them on a simple problem. The real test, of course, is how they work on more complex problems. This section tries some of these methods on Rosenbrock's and Wood's functions. Rosenbrock's function has already been described. Wood's function is:

$$F(X) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2$$

$$+ 10.1\,[x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$$

$$F(X^*) = 0 \quad X^* = [1,\ 1,\ 1,\ 1]$$

$$X^0 = [-3,\ -1,\ -3,\ -1].$$

It has a non optimum stationary point at (-.9679, .9471, -.9695, .9512) such that f = 7.876. As a starting point, Rosenbrock's function is attacked using various multipliers for a two step procedure. The problem, of course, in using such a procedure is the determination of ρ since the function is not convex. In Table XXIV, a Bolzano search is conducted on ρ using the described techniques.

All of these methods compare very well with those given in Table I and some show a large improvement but the problem once again is to determine ρ. A quick look at the output programs shows that quite often a ρ is chosen that leads to a much worse point than the starting point. Possible cures would be:

(1) Iterate forward slowly to determine ρ.

Table XXIV. Multistep Steepest Descent and Rosenbrock's Function (200 iterations)

Range of ρ^i	C_1 1. C_2 1.	C_1 .7 C_2 .7	C_1 1. C_2 .7	C_1 1. C_2 .5	C_1 .5 C_2 .5	C_1 .2 C_2 1.
0 to 10 ρ^{i-1}	.14836	.01905	.15171	.15344	.11874	.0104
0 to 1.	.17175	.00025	.00779	.00146	.00150	.00008
0 to 100ρ^{i-1}	.16881	.00005	.49196	.00152		
0 to 2.	.16905	.00011	.00375	.00455	.00081	.00003
0 to .5	.00188	.00575	.10602	.04068	.00654	.00320

(2) Use Bolzano but check the new point if it is not better adjust the range and try again.

Both would lead to points that must be improvements. They are both tried on Wood's function as shown in Table XXV.

In Table XXV "corrected" simply means the function value is evaluated at the new point j. If it is no better than the original point, the range of search on ρ is altered by dividing the maximum ρ by 10I, where I is the number of times ρ_{max} is cut. If it is the acceleration step, then ρ is set equal to 1 when no improvement is found. "Super correction" on the acceleration step simply means the search on ρ is altered by dividing the maximum ρ by 2^I, where I is the number of times ρ_{max} is cut.

The first judgement to make on these results is that multistep techniques do indeed offer potential for Wood's function as the results are much better than with the full gradient. Next, an almost arbitrary decision is made that Bolzano will be used as the search procedure for ρ and will be corrected. The normal step will have a ρ between 0. and .5, and the acceleration step will have a ρ between 0. and 10. The correction on the normal ρ will be $\rho_{max} = .5/10I$ where I is the number of times ρ_{max} is cut, and the correction on the acceleration will be $\rho_{max} = 10./2^I$ where I is the number of times ρ_{max} is cut.

A review of the results on the test function show that the following methods seem to offer the greatest potential and hence are used on Wood's and Rosenbrock's functions.

Table XXV. Multistep Steepest Descent and Wood's Function

	No. Iterations	Function
(1) k = 1 Complete relaxation Iterative search on ρ	200	$.3417712 \times 10^{-1}$
(2) k = 1 Underrelaxation Iterative search on ρ	200	.8355188
(3) k = 3 Complete relaxation Iterative search on ρ	200	$.286789 \times 10^{-4}$
(4) k = 3 Underrelaxation Iterative search on ρ	200	$.31932 \times 10^{-5}$
(5) k = 3 Underrelaxation Bolzano search on ρ 0→.5 corrected (normal step) ρ 0→2.0 corrected (acceleration step)	200	$.452774 \times 10^{-4}$
(6) k = 3 Complete relaxation Bolzano search on (a) ρ 0→.5 corrected (normal step) ρ.5→1.5 corrected (accelerated step)	200	$.39797 \times 10^{-4}$
(b) ρ 0→ .5 corrected (normal step) ρ 0→2.0 corrected (acceleration step)	200	$.1122477 \times 10^{-4}$
(c) ρ 0→ .5 corrected (normal step) ρ 0→3.0 corrected (acceleration step)	200	$.445907 \times 10^{-7}$
(d) ρ 0→ .5 corrected (normal step) ρ 0→10. super corrected (acceleration step)	147	$.1556167 \times 10^{-5}$

(1) k = 1 complete relaxation

(2) k = 1 underrelaxation

(3) k = N complete relaxation

(4) k = N underrelaxation

(5) k = N $\delta_i \alpha$ $(\nabla(X), u_i)$

(6) k = N decreasing sequence on variables within each Iteration

(7) k = N δ_i = constant

(8) k = N $C_i \alpha$ decreasing sequence

(9) k = N $\delta_i \alpha$ $A_i | A_i < (H^{-1}(x)\ \nabla(x))$

(10) k = N $C_i \alpha$ $(H^{-1}(x)\ \nabla(x),\ u_i)$

Once again all methods are terminated at Error = .0001 or N = 1000 whichever occurs first. The starting points are those discussed earlier and, finally, all discussions are withheld until all results have been presented. Results are presented in Tables XXVI to XXXV.

The one-step method of steepest descent does find the optimum point of Rosenbrock's function after 840 iterations but does not locate the optimum point of Wood's function after 1000 iterations. One-step underrelaxation on the other hand does find the optimum point of both and as expected does it in fewer iterations. The N step complete relaxation method is about the same as the one-step underrelaxation so it does perform quite well.

The N step underrelaxation method shows a large reduction in the number of iterations required to find the optimum points. This is not startling, but it is quite important to note that underrelaxation does work on non-convex functions as long as the points are

Table XXVI. (1) k = 1, Complete Relaxation - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.1995164 \times 10^{-4}$	$.998995 \times 10^{-4}$	840
Wood	$.1364094 \times 10^{-1}$	$.3784435 \times 10^{-1}$	1000

Table XXVII. (2) k = 1, Underrelaxation - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.1989517 \times 10^{-4}$	$.9994155 \times 10^{-4}$	495
Wood	$.361537 \times 10^{-4}$	$.9996317 \times 10^{-4}$	316

Table XXVIII. (3) k = N, Complete Relaxation - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.3978125 \times 10^{-4}$	$.3660467 \times 10^{-4}$	363
Wood	$.3507733 \times 10^{-4}$	$.7101662 \times 10^{-4}$	585

Table XXIX. (4) k = N, Underrelaxation - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.1895518 \times 10^{-4}$	$.7028216 \times 10^{-5}$	123
Wood	$.3498606$ c 10^{-4}	$.9714245 \times 10^{-5}$	192

Table XXX. (5) k = N, $\delta_i \alpha$ (∇x, u_i) - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.4112833023 \times 10^{-1}$	$.4106891632 \times 10^{-1}$	28*
Wood	$.787697697 \times 10^{-1}$	$.7753488541 \times 10^{-1}$	652**

*Step size cut too small
**Local optimum

Table XXXI. (6) k = N, Decreasing Sequence on Variables - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.394524 \times 10^{-1}$	$.772835 \times 10^{-1}$	112*
Wood	$.3026813 \times 10^{-1}$	$.388329 \times 10^{-1}$	85*

*Could not locate a better point.

Table XXXII. (7) k = N, δ_i = Constant - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.325019 \times 10^{-2}$	$.15471 \times 10^{-1}$	1000
Wood	$.7876962 \times 10^{-1}$	$.7758809 \times 10^{-1}$	356*

*Local minimum

Table XXXIII. (8) k = N, $C_i \alpha$ Decreasing Sequence - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.209740 \times 10^{-4}$	$.9713201 \times 10^{-4}$	171
Wood	.1092631	.118603	15*

*Could not locate a better point.

Table XXXIV. (9) k = N, $\delta_i \alpha A_i \mid A_i = (H^{-1}(x) \nabla(x))$ - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.233154 \times 10^{-3}$	$.1153086 \times 10^{-2}$	930
Wood	$.787652016 \times 10^{-1}$	$.7760736465 \times 10^{-1}$	314*

*Local optimum.

Table XXXV. (10) k = N, $C_i \alpha (H^{-1}(x) \nabla(x), u_i)$ - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.3675035 \times 10^{-5}$	$.1864767 \times 10^{-5}$	69
Wood	Altered step size ended on a high point causing exponent overflow.		

tested before the actual step is taken.

Now, there are several methods that are tried where the step size is altered by some predetermined standard (methods 6, 8, and 10). Since these test functions are not convex, it is to be expected that changing the step sizes frequently as in the above methods would pose problems for with steep contours a change in a step size could drastically affect the results. The outputs of these methods show that indeed this is the case. Most of the time the methods jump up and down ridges so they sometimes eventually get "hung up." This could happen by accidentally "jumping" to an extremely high point where no point in the search vicinity can be found that is better than the previous best point. When they do work, however, they do work quite well (e.g., method 8 on Rosenbrock). Of course, a possible remedy would be to test the points chosen and adjust if they are worse. This seems, however, to lose the concept of "altering step sizes;" and it imposes a great deal of extra computations. It seems that more often than not the adjustment would have to be made here. It should be safe to assume, then, that any method that has a great deal of varying of step sizes may not work very efficiently on non-convex functions.

Methods 5, 7, and 9 all involve using coordinate directions in a pattern search type routine. It is interesting to note that these multistep coordinate direction methods do not seem to have the "ridge following" property that pattern search enjoys. This is not too startling, however, since, given good past results, pattern search

will accelerate to follow the ridge while the multistep approach does not have quite as much acceleration along a ridge. Perhaps some type of acceleration mechanism could also be built into the multistep coordinate direction methods. Another problem of pattern search appears in that it will quite often go to a local optimum. In all three cases here (Wood's function) the search stops at the nearest local optimum as shown. This also is not too startling.

Several other points are worth mentioning here. First, $k = N$ - Complete Relaxation did not perform quite as well as may be expected. A look at the output programs show that quite often (in fact a large percentage of the time) the acceleration step size was chosen as the maximum and the improvement was substantial. Perhaps the range could be increased even more. The step is already 10 times the calculated step but more may help.

Next, methods 3, 4, and 8 are run with $k = 3$. The results are highly surprising as shown in Tables XXXVI, XXXVII, and XXXVIII. These results are very startling, but not necessarily upsetting. In the case of non-convex functions the optimum number of steps in a multistep procedure seems to be different for the different functions. The best that can be said is that there is evidence that $k = N$ should always be a "good" number (N directions are independent and the experimental results show $k = N$ is always a fairly efficient procedure). $k = N$ then is used in this research as the best available procedure.

In retrospect, perhaps the reason $k \neq N$ sometimes performs better than $k = N$ lies simply in the fact that valleys are not straight in

Table XXXVI. k = 3 Complete Relaxation - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.2279553 \times 10^{-4}$	$.9110167 \times 10^{-4}$	280
Wood	$.3571645 \times 10^{-4}$	$.966814 \times 10^{-4}$	90

Table XXXVII. k = 3 Underrelaxation - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.3819741 \times 10^{-4}$	$.3426911 \times 10^{-4}$	80
Wood	$.3256649 \times 10^{-4}$	$.839305 \times 10^{-4}$	86

Table XXXVIII. k = 3 $C_i\alpha$ Decreasing Sequence - Computational Results

Problem	Function	Error	No. Iterations
Rosenbrock	$.479102 \times 10^{-5}$	$.2382453 \times 10^{-4}$	4
Wood	.1069949	.572544	8*

*Could not locate a better point.

all cases. In fact, in Rosenbrock's and Wood's function they are indeed not straight. Sometimes $k \neq N$ may "pick up" the curve and follow it. It is impossible to predict curves in the valley so there seems to be little benefit in exploring the concept. Instead, as said earlier $k = N$ is always a "good"choice.

Next, the series of quadratic approximations with a Dynamic Programming approach as described earlier in this chapter is run on Rosenbrock's function which again is neither convex nor quadratic. After only five (5) iterations, the error was reduced to $.93 \times 10^{-7}$ so the procedure was terminated. This is extremely promising as only Gauss's method to least squares does any better and this D.P. approach is not limited to least squares formulation.

Next, the quadratic approximations for $N = 2$ and $N = 3$ as described earlier in this chapter are tried on Wood's function. The results are shown in Table XXXIX. At first glance, the results are not encouraging, but it is very important to note that the number of iterations is not accurate. An iteration is counted each time a variable is changed; so to compare this with any of the previous methods (where four variables are changed each iteration) is really quite unfair. A better measure would be to divide by four - the number of variables. In that case, the two variable approximation almost converged in 250 iterations while the three variable approximation converged to the optimum in 177 iterations. This shows they compare very favorably with other methods. Had the results for four variable approximations been available, the results should have been

much more impressive.

Finally, in both the above approximations the variables being considered were rotated one at a time so the sequence for the three variable approximation was 123, 234, 341, 412, 123, ... A run was made with the three variable approximation where the variables were rotated two at a time (i.e., 123, 341, 123, ...). After 1000 iterations, the error value was .117748 so the method was nowhere near terminating. It seems safe to say, then, that for an approximation where the number considered is less than the number of variables in the original problem, the variables should be rotated one at a time.

The quadratic approximation method, where steps are approximated by solving simultaneous linear equations, described earlier in this chapter was tried on Rosenbrock's and Wood's function, where the step size was held to 10 units or less, with results as shown in Table XXXX.

These are interesting as Rosenbrock's function was solved in 62 iterations with a very fast execution time. In fact, this is the best recorded except for the two variable quadratic approximation where the steps were determined using a dynamic programming approach. These two methods are very similar except that the one using linear equations can be extended to more complicated functions when the dynamic programming approach so far is only accurate up to N = 3.

It is quite interesting to note that earlier mention was made that these methods could quite easily lead to local optima and saddle points. In Wood's function, the local minimum was found as shown. The starting point was moved to X° = (3, -1, 3, -1) to see what would

Table XXXIX. Quadratic Approximation on Wood's Function

N	Function	Error	Comp.	Exec.	No. Iterations
2	$.1979284 \times 10^{-3}$	$.5382972 \times 10^{-3}$	.28	3.62	1000
3	$.3582612 \times 10^{-4}$	$.9668675 \times 10^{-4}$	.49	2.65	708

Table XXXX. Quadratic Approximation by Linear Equations

Function Considered	Function Value	Error	Comp.	Exec.	No. Iterations
Rosenbrock	$.4684538 \times 10^{-2}$	$.9412707^{-5}$	.33	.41	62
Wood	Local optimum in 10 iterations				

happen and the procedure went to a saddle point in 20 iterations with X^{20} = (-.031251, .165971, -.031258, .19264). This should be carefully watched when using these procedures.

In both these methods, the step size was held to 10 units or less. This was an arbitrary decision so perhaps a different step size constraint would be better. This is very similar to what occurs in the Gauss-Levenberg procedure discussed by Wilde and Beightler [30].

Finally, the normalizing and changing of signs is quite important to the success of this procedure. This can be seen quite easily by considering the case where the origin is nearly on the hyperplane. Alpha, the step size, would be very small and the procedure would not move. By normalizing and changing signs, this possibility is eliminated.

All the work in this chapter has been toward understanding multistep concepts as related to unconstrained optimization. These results do have applications in the area of constrained optimization. Chapter III attempts to discuss how these concepts can be applied to the constrained problem.

CHAPTER III

MATHEMATICAL PROGRAMMING APPLICATIONS OF MULTISTEP CONCEPTS

Application to Existing Algorithms

Chapter I defined unconstrained and constrained optimization problems and described some of the solution procedures available for constrained optimization. Chapter II developed multistep concepts and algorithms for use in unconstrained optimization. This chapter first describes the application of these multistep techniques to some existing constrained optimization algorithms and then proposes a new modification to a well known method-penalty formulations. Finally, a solution to a "real world" nonlinear programming problem is examined in light of multistep concepts.

The constrained minimization mathematical programming problem was defined as:

Find $\bar{X} \mid \bar{X} \varepsilon \Gamma$ such that

$$f(X) \geq f(\bar{X}) \text{ for all } X \varepsilon \Gamma$$

$$\Gamma = E^n \; \Omega k \mid r_i(X) \leq 0 \quad (i = 1, 2, \ldots, m) \rightarrow X \varepsilon k$$

The general problem was then broken into several classes and solution procedures were mentioned that could solve problems in each of these classes. Any time the solution procedures involve directional search, incomplete relaxation and multistep concepts are applicable. This chapter discusses only a few of the more general algorithms, but it is important to understand that the same type of reasoning applied

to these few algorithms can also easily apply to any directional search method.

One of these general algorithms is the Ricocheting Gradient method of J. L. Greenstadt [11] which is applicable to the general nonlinear programming problem described earlier. A macro description of the algorithm was presented in Chapter I. As implied there, the method starts at some interior feasible point and moves along the gradient until (1) a free optimum is found, (2) the direction is no longer profitable, or (3) a set of constraints is violated. It is interesting to note that in this phase the method uses complete relaxation. It should be profitable to employ multistep reasoning or at least incomplete relaxation (probably underrelaxation) in this stage. Particularly, one of the multistep methods that bases the step size upon the amount of improvement available could be quite beneficial [e.g., $C_i\alpha(\nabla(X^i), U_i)$ and $C_i\alpha(H^{-1}(X)\nabla(X^i), U_i)$]. Also the decreasing sequence type methods could be quite useful as it seems desirable to shorten the steps as the search approaches the constraint boundary to allow the search to switch directions before a constraint boundary is encountered. An intuitive argument for all this is that, by a better search in the interior before moving to the constraints, it may be possible to reduce the search along the constraints which is much more costly.

If the method finds a free optimum, of course the procedure terminates; if the direction is no longer rewarding, the above discussion applies; but if the direction encounters a constraint, then the

ricocheting gradient method finds the intersection of the direction and the constraint(s) violated. Here again successive application of underrelaxation would be much simpler and should follow the constraints also. Once it has found this intersection, the method searches the immediate region for the intersection of all the active constraints and moves to that point. (A short correction step since threshold limits are considered in determining what constraints are active.) Here the method gets quite complicated; but, in general, by means of quadratic subprograms it generates contour directions that move back into the feasible region by maximizing the rate of movement away from the active constraints. This is continued until another set of constraints is violated. Then triangularization is used to generate the point along these directions that is the "furthest" from the constraints. The procedure restarts at this point. As mentioned before, multistep concepts or incomplete relaxation could have been used to stay within the feasible region instead of generating these subprograms at active constraints; also underrelaxation could have been used to find the new starting point along the ricocheted direction instead of triangularization.

Rosen [26], [27] has proposed another gradient type searching procedure for constrained optimization. In Part I [26], he develops the algorithm for the nonlinear programming problem with linear constraints. In Part II [27], he discussed how this can be extended to the general nonlinear programming problem with general nonlinear constraints. Essentially, the method estimates the constraints by

tangent linear hyperplanes at the point in question, makes the normal calculations and moves, and then corrects this to the actual constraints. This discussion is, therefore, on the general method for linear constraints.

Rosen's method is well suited for the application of incomplete relaxation and/or multistep concepts. At each iteration the method develops an "improving direction" (the projected gradient) and as Chapter II stated several times its results can be applied to any "improving direction." Assuming the starting point is an interior feasible point, the same discussion that applied to Greenstadt's first few steps applies to Rosen's first few steps. As long as there are no active constraints, the projected gradient direction is the gradient direction so multistep concepts and incomplete relaxation can be applied. In fact, it does seem advantageous to move around the feasible region before going to a constraint for the reasons stated before.

Once a constraint(s) is encountered, the directions are not the gradient directions but are the projections of these upon the boundary of the feasible region. This is equivalent to a search within the n-k dimensional feasible hyperspace. It is interesting to note that the method follows this projected gradient until another constraint(s) is violated. At this point, if the gradient has reversed directions (measured by the scalar product of the gradient on the projected gradient), then a quadratic approximation is used to determine the approximate optimum along the direction. Instead of doing

this, it should be possible to use underrelaxation when necessary to backtrack. It is also conceivable that the step along a projected gradient could be proportional to the magnitude of the potential improvement [$C_i\alpha(\nabla(X^i), U_i)$ or $C_i\alpha(H^{-1}(X^i)\nabla(X^i)]$. Finally, it is reasonable that a decreasing sequence of step sizes would be profitable.

The method of feasible directions by Zoutehdijk [31] is in reality a broad class of methods. In fact, as mentioned earlier, it can be developed so broadly that most directional type methods would fall under this class. Basically, the method determines the direction to be used at each iteration by maximizing a small linear or nonlinear subprogram $\max(\nabla(x^i), S)$ subject to some normalizing constraint (S is the direction to be used). The choice of the normalizing constraint determines what particular method is being used. Additional constraints can be added that make the directions chosen conjugate directions which for quadratic problems insures convergence in a finite number of iterations. The subprogram is maximizing the cosine of the angle between the gradient of the function at a point and the direction S while requiring that S must lie in the feasible region. The similarity to Rosen's function can now be readily seen.

As with all the other directional type algorithms, once the feasible direction has been obtained, the optimum along that direction and/or the point where it encounters another constraint must be determined. As before, this is where the multistep concepts and incomplete relaxation may be used. In particular, if the starting point

is an interior point, then these concepts can be used with the first few steps. All that has been said about the previous methods would again apply.

The final class of methods to be considered here is that class where the constraints are imbedded into the objective function to form an augmented unconstrained objective function whose solution is also the solution to the constrained problem. This class of methods is called the general penalty formulation. As mentioned in Chapter I, the most well known of these is the Lagrangian formulation. There are many other formulations, however, and they may be divided into two main classes:

(1) Exterior point penalty functions

(2) Interior point penalty functions.

The first class involves adding the constraints to the objective function as penalties and reducing the penalty such that the sequence of solutions converges to the constrained optimum. The second class does the same by adding the constraints in to impose reward for feasibility and decreasing that reward as the sequence continues.

Since the major advantage to these methods is that a constrained problem can be formulated as an unconstrained problem or at least as a sequence of unconstrained problems, any of the unconstrained optimization methods can be used. This means that any of the developments in this research for unconstrained optimization have direct applications in the penalty formulations. In fact, in Chapter I, it was shown that this is the reason for the research into multistep and incomplete

relaxation methods for unconstrained optimization. Once again, any of the results developed in Chapter II can be used to solve these penalty imbedded unconstrained problems.

To demonstrate this, a penalty type method is now forumulated and used to solve a small nonlinear programming problem. Some of the results of Chapter II are applied to this solution procedure.

A problem used by Kunzi [22] to evaluate Beal's method is given below

$$\begin{aligned} \min Q &= -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2 \\ \text{S.T.} \quad & x_1 + x_2 \leq 2 \\ & x_1 \geq 0 \\ & x_2 \geq 0. \end{aligned}$$

The penalty formulation chosen (exterior point) is

$$F(x_p) = (f(x) + \frac{1}{2} \sum_{i=1}^{m} e^{kg_i(x)}$$

where $f(x)$ = objective function and

$g_i(x)$ = constraint functions ($i = 1, \ldots, m$) which yields

$$F(x_p) = -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2 + \frac{1}{k} e^{k(x_1+x_2-2)} .$$

Since e^x quickly becomes quite large, care must be exercised when working with a computer. Limiting the value of x to a preset maximum should resolve this problem.

To solve this for one iteration, k is chosen to be 1 and a 2 step complete relaxation gradient type procedure is used to solve

the unconstrained function starting at $X^\circ = (4,4)$ with $C^* = 1.5, .5$, and $f(X^*) = -5.5$. After one iteration (3 steps) the following results

$$X = (1.49851, \quad .5012521) \qquad f(X) = -5.491289.$$

Actually k should be increased, the problem solved again, and so forth until convergence to X^* can be shown. In this case error was already at 3×10^{-6} so the procedure was terminated. Since $e^{kg_i(X)}$ quite easily gets too large for a computer, it may seem desirable to increment k slowly at each iteration as $g_i(X)$ gets closer to 0. Doing this on the above problem, however, shows that it takes approximately 20 iterations to reach error = 3×10^{-6} so this concept is dropped from consideration.

Proposed Penalty Formulation

Ghare recently proposed an alternative method that actually bears resemblance to the penalty formulations and to Greenstadt's ricocheting gradient. Essentially, the method performs a normal gradient search from an interior feasible point until a constraint(s) is encountered. At this time the method ricochets back into the feasible region by minimizing a penalty function and the iterative process continues. A step-by-step algorithm for the procedure for minimization and linear constraints is given below: (note for non-linear constraints it is only necessary to estimate the constraint by a linear constraint and then correct as necessary as Rosen [27] demonstrates.)

The problem is

$\min f(X)$

s.t. $g_i(X) \leq 0 \qquad (i = 1, 2, \ldots, m)$

where $g_i(X)$ is linear $\qquad (i = 1, 2, \ldots, m)$.

(1) Normalize all constraints as below:

$$g_i^1(X) = g_i(X)/|g_i(X)| \leq 0.$$

Formulate the penalty function as shown

$$h(b) = \sum_{i=1}^{m} k_i e^{g_i(X)},$$

(2) Pick X°, if infeasible or non-interior, follow the path of steepest descent for the penalty function until an interior feasible point is obtained. The ideal starting point would be the minimum X^h of the penalty function h(b). Set i = 0 and go to 3.

(3) Follow the functional steepest descent direction of the unaugmented objective function until a constraint is encountered or an extreme point is found. $[X^{i+1} = X^i - \rho\nabla f(X^i) | \rho = \min [\rho'', \rho']$

$$\rho^i = \min_i \rho^i \mid g_i(X + \rho^i \nabla f(X)) = 0 \qquad i = 1, 2, \ldots, m$$

and $\rho'' \underline{\Delta} (\nabla(x - \rho\nabla f(X)), \nabla f(X)) = 0.]$

Check X^i for optimality. If optimum STOP. If not, continue.

If $\min[\rho'', \rho'] = \rho''$, set i = i+1 and go to 3.

If $\min[\rho'', \rho'] = \rho'$, go to 4.

(4) Evaluate $(\nabla f(X), \nabla h(X)) = \Theta$. If $\Theta > 0$, set $U = -\nabla h(X)$ and go to 5. If $\Theta < 0$, determine direction U where

$$U = -\nabla f(X) - k\nabla h(X)$$

$$k = \frac{(-\nabla f(X), -\nabla f(X))}{(-\nabla h(X), -\nabla f(X))}.$$

If $U = 0$, set $U = -\nabla h(X)$ and go to 5.

(5) Search along U until h(b) starts to increase. Call that the new point.

$$X^{i+1} = X^{i} + \rho U \mid (\nabla h(X + \rho u), u) = 0.$$

Test X^{i+1} against all constraints. If any are violated, then multiply that constraint's penalty by a positive value k_i to increase its importance, and go back to 4. If no constraints are violated, set $i = i+1$ and go to 3.

Solving a Typical Constrained Problem

Now, to investigate the efficiency of the above method and primarily to demonstrate the applicability of nonlinear programming to real world problems the following problem is solved (note this is a maximization problem).

The xyz chemical processing company has some spare factory capacity available it would like to use. There are five products that it can produce in various quantities to utilize this capacity. These five products can be sold for various profits per ton produced, in hundreds of dollars, as shown in the matrix C = [15 21 5 102 78]. The market for these products is quite small so the profit per unit

decreases quite rapidly as the tons sold increases. This quadratic effect can be shown by the matrix G shown below (twice G would be the hessian matrix of the problem and the objective function is given by $CX + X^TGX$).

$$G = \begin{vmatrix} -10 & .2 & 0 & .1 & 0 \\ .2 & -.1 & 0 & 0 & 0 \\ 0 & 0 & -.01 & .2 & 0 \\ .1 & 0 & .2 & -12 & 0 \\ 0 & 0 & 0 & 0 & -15 \end{vmatrix}$$

It is important to notice that sometimes the sale of one product helps the profit of another as they can be sold as a package. This can be seen by examining the positive elements of G. The objective is to maximize the profit given by $CX + X^TGX$. Since spare capacity is being used to produce these products, the capacity is a restriction. The blending unit for example would be available only 70 hours per month. It requires the following amount of time per ton for each product.

Product	Blending (hours)	Preheat (hours)	Evaporator (hours)
1	.0251	0	.015
2	10.01	.251	.021
3	.25	.384	.003
4	.0775	.752	.161
5	.400	.103	.028
Time available	70	2.7	.8

The preheater and evaporator also have required times and times

available as shown in the chart.

The problem can now be formulated as

$$\max CX + X^T GX$$

$$\text{s.t. } g_i(X) \leq 0 \quad i = 1, 2, \ldots, m$$

or $\max 15x_1 + 21x_2 + 5x_3 + 102x_4 + 78x_5 - 10x_1^2 + .4x_1x_2 + .2x_1x_4$

$$- .1x_2^2 - .01x_3^2 + .4x_3x_4 - 12x_4^2 - 15x_5^2$$

subject to

$$.251x_2 + .384x_3 + .752x_4 + .103x_5 \leq 2.7$$

$$.025x_1 + 10.01x_2 + .25x_3 + .0775x_4 + .400x_5 \leq 70$$

$$.015x_1 + .021x_2 + .003x_3 + .161x_4 + .028x_5 \leq .8$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_3 \geq 0$$

$$x_4 \geq 0$$

$$x_5 \geq 0.$$

These important points should be noted.

(1) Since all constraints are linear, the feasible region is convex.

(2) The matrix G is negative definite, so the objective function is strictly concave. This in conjunction with 1 means

there is a unique maximum.

(3) The feasible region is convex, so the penalty minimization concept should always tend to pull the search back into the interior of the feasible region.

(4) To test for optimality an interior point, the gradients need to be compared to 0; and to test for optimality a point lying on a constraint, the gradients must be orthogonal to the constraint and point outward for an optimum to exist. This means that the scalar product of the gradient and the perpendicular to the constraint must be close or equal to one. These two tests can be used to test for optimum. In particular, if the above scalar product is greater than some value close to 1, the point can be called optimum or if the gradient is equal to 0, then the point can be called optimum.

To proceed with the algorithm:

Step 1: The normalized constraints are:

$$.319107x_2 + .488196x_3 + .95605x_4 + .130948x_5 - 3.432625 \leq 0$$

$$.000249x_1 + .099672x_2 + .002489x_3 + .000772x_4 + .003983x_5 - .697008 \leq 0$$

$$.547845x_1 + .766983x_2 + .109569x_3 + 5.880204x_4 + 1.022644x_5 - 29.218408 \leq 0$$

$$-x_1 \leq 0$$

$$-x_2 \leq 0$$

$$-x_3 \leq 0$$

$$-x_4 \leq 0$$

$$-x_5 \leq 0$$

Step 2: $X^\circ = [.01 \quad .01 \quad .01 \quad .01 \quad .01]$

which is an interior feasible point.

Step 3: $\nabla f(X) = C + GX$

$$\begin{vmatrix} 1.5 + 20x_1 + .4x_2 + .2x_4 \\ 2.1 + .4x_1 + .2x_2 \\ .5 + .02x_3 + .4x_4 \\ 10.2 + .2x_1 + .4x_3 + 24x_4 \\ 7.8 + 30x_5 \end{vmatrix} .$$

$\nabla f(X^\circ) = [207.5 \quad 8.1 \quad 4.7 \quad 256.2 \quad 307.8]$.

Calculation of ρ'':

If $U(X^i)$ = direction used at X^i, then

$U(x^\circ) = U_1 + \nabla(X^\circ) = C + 2GX^\circ$.

$U(x^1) = U_2 = \nabla(x^\circ + \rho'' U_1) = C + 2GX^\circ + \rho'' 2GU_1$.

For U_2 to be orthagonal to U_1,

$0 = (U_1, U_2) = (U_1, U_1 + 2\rho'' GU_1)$.

That yields

$(U_1, U_1) + (U_1, 2\rho'' GU_1) = 0$, and

$$\rho'' = \frac{(U_1, U_1)}{2(U_1, GU_1)} ,$$

which by substitution is

$$\rho'' = - \frac{[\nabla(X^\circ), \nabla(X^\circ)]}{2[\nabla(X^\circ), G\nabla(X^\circ)]}.$$

In this problem, $\rho'' = .03945382$.

Calculation of ρ'

Since $g_i(x^\circ + \rho^i \nabla(x^\circ)) = b_i \quad (i = 1, \ldots, m)$,

$$\rho^i = \frac{b_i - \sum_{j=1}^{N} a_{ij} x_j}{\sum_{j=1}^{N} a_{ij} \frac{\delta g_i(x)}{\delta x_j}} \qquad (i=1, \ldots, m).$$

In this problem,

$\rho^1 = .02927366$

$\rho^2 = .2786541$

$\rho^3 = .04146573$

$\rho^4 = -.0006754$

$\rho^5 = -.00047615$

$\rho^6 = -.00199848$

$\rho^7 = -.00009826$

$\rho^8 = -.00012870$

Dropping all negative ρ^i yields

$\rho' = .02927366$

which is smaller than $\rho'' = .03945382$

so $\rho = .02927366$. This step size leads to

$X^1 = (.4434257 \quad .6248048 \quad .1564775 \quad 2.989061 \quad 2.284562)$. Here constraint 1 is satisfied as a strict equality, and the function = 316.6887. The gradients $\neq 0$ and the scalar product = .90 so the procedure continues.

Step 4: $\nabla f(X^1)$ = (6.878518 21.05238 6.192494 30.413800 9.463150).

$\nabla h(b)$ = (-.641596 -.1626472 -.3655985 .907246 .03146017).

$(\nabla f(X^1), - h(X^1)) = \Theta < 0$, so k must be determined. This leads to

$U = -\nabla f(x^1) - k\nabla h(x^1)$ =

(62.544310 35.16389 37.912330 -48.300170 6.733621).

Step 5: A search in this direction shows $h(X^1 + \rho U)$ reaches a minimum at

X^2 = (3.132829 2.136850 1.786708 .912156 2.574107).

No constraints are violated.

Step 3: $\nabla f(X^2)$ = (-46.61936 21.82574 5.329127 81.44949 .776794).

ρ'' = .04547612

ρ^1 = .00764627

ρ^2 = .2085736

ρ^3 = .0374776

ρ^4 = .06720012

ρ^5 = -.09790498

ρ^6 = -.335272

ρ^7 = -.01119904

ρ^8 = -3.313755

Dropping all negative ρ' yields

ρ' = .00764627

which is smaller than ρ''

so ρ = .00764627. This leads to

X^3 = (2.7763630 2.303735 1.827455 1.534939 2.580046)

where the function = 255.7324. Constraint 1 is satisfied as a

equality.

Step 4 The iterative process continues. Table XXXXI shows the rest of the iterations for this method.

Now the method seems to find itself in a "nested" constraint region or an area where several constraints are close together; also the vector combination tries to follow the constraint as the gradients are nearly perpendicular to the constraining hyperplane. The penalty formulation then flounders around in Step 4, continually increasing the penalty, until a minimum is finally found far away from the nested constraints. In fact, it was located close to X^1 where then the method essentially starts over and goes through the same 10 iterations to X^{10} with $f(X^{10}) = 316$, etc.

In judgment, the method follows constraints quite well but sometimes has difficulty locating a penalty minimum. It is interesting to speculate that this could happen near X* since the vector combination in that case should fall very close to the constraint itself making it difficult to locate a minimum. This does happen for at X^{10} six penalty formulations are searched before a feasible minimum is found and this lies on the constraint in question within three decimal places ($\rho' = .0002176$).

An attempt to check for optimality by the scalar product seems to work quite well. As mentioned earlier, close to the optimum the vector combination would tend to follow a constraint; so finding a minimum would be difficult. The scalar product would, therefore, never reach one but rather some value close to 1. In

Table XXXXI. Chemical Problem and Penalty Formulation

Iteration	Function Value	Penalty Value
1	316.6887	3.72111
		1.945409
2	255.7324	2.25140
		2.058816
3	296.1557	2.39779
		2.354674
4	309.42400	2.52326
		2.520576
5	312.0236	2.55728
		2.556231
6	313.3728	3.57576
		3.443511
7	327.2014	3.86605
		3.84197
8	331.5307	3.98867
		3.910407
9	330.3513	4.02154
		3.93621
10	336.8623	7.10619
		6.687298

this problem, for example, the scalar product becomes .931 at X^1 with $f(X^7) = 327$ and changes very little through X^{10}. (At X^{10} the scalar product is .920.) If the program had been terminated then at .93 for the scalar product, the point would have been sufficiently close to the best value found by the search. The optimality criteria then seems to work quite well.

It is now obvious that, as the scalar product of the gradient and the normal to the constraint approaches 1, the method tends to "bog down." This isn't surprising for the reason as stated above concerning the penalty minimization. It would be quite simple to use Rosen's projected gradient for a final step or two as a way to "close in" on the optimum point. The active constraints are known beforehand and should not change as the steps would have to be made. This would improve the accuracy as measured by $(X - X^*, X - X^*) = E$ but should not improve significantly the functional value $f(X)$.

In retrospect, this tendency of the search to go to a point like X^{10} and then diverge, go back to X^{10}, etc. is very similar to what sometimes happens in the Newton-Raphson iterative process for finding roots of polynomials. In that case, if the steps are chosen to be one half the calculated steps then convergence can be shown. This is tried on the problem with steps at .5, and .1 of the calculated step, all starting at the original point

$$X^\circ = [.01 \quad .01 \quad .01 \quad .01 \quad .01].$$

With one half the calculated step, the search followed the same path (only slower) to approximately the same point X^{10} where $f(X^{10}) = 336$.

This time, however, instead of a major divergence, the search would flounder, go backwards a short distance and then jump back to X^{10}. This is encouraging for it shows a strong desire to go to that point and it dramatically illustrates the difficulty in minimizing the penalty when the search is near the optimum. Once again, the retreat is almost directly along the constraint. This time the scalar product reached a maximum at X^{25} with $f(X^{25}) = 335.753$ and the scalar product = .938. The improvement does not seem to justify the extra computations necessary.

Next, using steps at .1 of the calculated step, the search followed the same path. This time the search stopped moving around X^{10}, floundered about as before, but stayed very close to X^{10}. Here f(X) stayed around 335 to 336. The scalar product peaked at X^{59} with $f(X^{59}) = 336.3$ and the scalar product equal to .937. Once again the improvement does not seem to justify the extra computations.

Next the procedure is tried at three different starting points for the same function. They are

$X^{\circ} = [.01 \quad .01 \quad .01 \quad .01 \quad .01]$

$X^{\circ} = [\ 1. \quad 1. \quad 1. \quad 1. \quad 1.]$

$X^{\circ} = [\ 2. \quad 0 \quad 2. \quad 0. \quad 0.]$.

In all cases, the search went to approximately the same point as would be expected and the scalar products peaked at about .93.

To increase the efficiency and effectiveness of this algorithm, several things are necessary:

(1) First, some method of searching in a "nest" must be developed.

A possibility would be to use the direction perpendicular to the closest constraint and make a very short step in this direction. When an intersection is reached, the search could revert back to the penalty formulation with only a short step used.

(2) Secondly, some sort of final moving toward the true optimum would be profitable. Presently the program terminates when the optimality criteria approaches .93 - .94. At this point if a projected gradient is used as mentioned earlier, only one or two steps would be necessary to get very close to X*.

(3) Finally, the ricocheting pattern lends itself very well to acceleration mechanisms as discussed in Chapter II under multistep theory. A search along the direction defined by two or more of these points could greatly enhance the ability to "follow" a constraint.

Using the best calculated point, X* for the chemical problem is X^{10} and the profit at this point would be $33,690. Therefore, approximately 1 ton of product 1, 4 tons of product 2, .2 tons of product 3, 1.8 tons of product 4, and 2.3 tons of product 5 should be produced. If the scalar product is used as the optimality criteria, the search would have stopped at X^{7} with the profit about $32,720. At this point, about 1.2 tons of product 1, 2.6 tons of product 2, .4 tons of product 3, 2.2 tons of product 4, and 2.4 tons of product 5 should be produced. This is a rather significant change in E but not much change in the profit. The suggested termination procedure would have improved this.

This concludes the basic research of this dissertation. The

next and final chapter attempts to bind together the results to make a "package" and then further research areas are mentioned and discussed.

CHAPTER IV

SUMMARY RESULTS AND EXTENSIONS

One of the objectives of this research was to explore the field of incomplete relaxation and multistep theory to learn as much as possible and to show where further work can be done. With that goal, the first three chapters of this work reported the research done. The results were given with little comparative merit evaluation. This chapter first summarizes the research areas explored with a discussion of results and significance. Since there have been many areas explored, there is a great amount of research remaining to be done. The final part of this chapter mentions thirteen of these areas.

Since one objective of the research was to study multistep directional procedures for solving unconstrained and constrained optimization problems, Chapter I starts by defining the unconstrained and constrained problems and discusses the importance of convexity in these problems. Next a sample convex quadratic problem is presented and discussed. This problem is used as the test function for unconstrained optimization. The two major directions considered in this work (gradient and coordinate) are next defined and discussed along with a brief consideration of optimal trajectories. Chapter I then proceeds into a discussion of directional techniques as applied to the solution of simultaneous linear equations. In particular, steepest descent methods are explored using the concepts of incomplete and complete relaxation and multistep methods. It is found that, in

general, for the solution of simultaneous linear equations, underrelaxation is better than complete or overrelaxation for one step methods; but that good multistep methods give the best results. Chapter I next presents a directional method for the solution of simultaneous linear equations that possesses good convergence properties for near singular systems. Next, Chapter I discusses the application of these concepts of incomplete relaxation and multistep theory to unconstrained optimization problems. Some suggestions for multistep methods are made that are to be used in Chapter II when the applications to unconstrained optimization are explored. Finally, Chapter I presents a brief discussion of some well known procedures for the solution of nonlinear mathematical programming problems with a particular emphasis on the directional following methods.

Chapter II begins by attacking the test function with one step gradient methods using underrelaxation, complete relaxation, and overrelaxation. Underrelaxation is found to be much better than the other two as approximately 1/8 as many iterations are necessary. Next, some multistep procedures mentioned in Chapter I are used on the test function. Twenty-two different procedures are tried often with three different sets of steps for each forty-nine methods total. Each of these concepts is compared with one or more similar well known procedures. Some of the well known procedures that are discussed are deflected gradient, gradient pattern, pattern search, conjugate gradients, and super-memory gradient.

The goal of all this work on the test function was to choose the most promising methods for further experimentation. The following were chosen as offering the most potential for comparative purposes (k = the number of steps in the multistep method):

(1) k = 1, gradient direction-underrelaxation.

(2) k = 1, gradient direction-complete relaxation.

(3) k = N, gradient direction-complete relaxation.

(4) k = N, gradient direction-underrelaxation.

(5) k = N, coordinate direction - where step sizes are proportional to the scalar product of the gradient onto the coordinate direction.

(6) k = N, gradient direction-decreasing sequence on variables within each iteration.

(7) k = N, coordinate direction-constant step size.

(8) k = N, gradient direction-decreasing sequence of step sizes.

(9) k = N, coordinate direction where the step sizes are proportional to the scalar product of the hessian inverse times the gradient onto the coordinate direction.

(10) k = N, same as 9 but in the gradient direction.

(11) k = N, quadratic approximation with steps determined by dynamic programming.

(12) k = N, quadratic approximation with steps determined by partial solution of simultaneous linear equations.

Each of these methods are then tried on Rosenbrock's and Wood's functions with varying degrees of success. Of the first 10 methods,

4 gives the best results requiring 123 and 192 iterations, respectively. Somewhat surprisingly, method 2, a single-step method, also does quite well requiring 495 and 316 iterations, respectively. These are less than one half the iterations for method 1 (the classical gradient method) which required 840 for Rosenbrock's function and did not reach optimum in 1000 iterations on Wood's function. Method 3 also does quite well terminating in 363 and 585 iterations, respectively.

Methods 5 through 10 have one thing in common. That is, they all find a good point by a search procedure and then alter the step some way to obtain a different point. In the test function with smooth contours this works quite well; but these two functions do not have smooth contours. In fact they are non-convex, possess valleys and ridges with very steep sides, and Wood's has local optima. Altering the step size can, therefore, cause a great deal of trouble. Tables XXX through XXXV in Chapter II demonstrate this very dramatically. Most of the time the methods flounder in a valley, terminate because of exponent overflow, or converge to a local optimum. These methods must then be handled with extreme care when dealing with non-convex functions. It is interesting to note that, when they do work, they work quite well. Method 10, for example, minimized Rosenbrock's function in 69 iterations. This was the best so far.

Methods 11 and 12 seem to offer a great deal of potential. Method 11, for example, minimized Rosenbrock's function in only 5 iterations. In fact, the error at the final point was 9×10^{-8}. Although

generalized results for four variable problems were not obtained for method 11, the three variable approximation was used on Wood's and it terminated in 177 iterations. The four variable approximation should do much better when results are available.

Method 12 completely minimized Rosenbrock's function in 62 iterations (the best except for method 11) and converged to a local minimum for Wood's function in only 10 iterations. Methods 11 and 12 are essentially the same methods except 12 imposes a cruder approximation than 11 and the results for 11 are not available for N greater than 3. Both are very promising, therefore.

Since all the research through Chapter II is on unconstrained optimization, Chapter III attempts to show how these results can also be applied to constrained optimization. In the first section, the discussion concentrates on the application of these results to some of the well known optimization procedures. Those discussed are the ricocheting gradient of Greenstadt [11], the methods of feasible directions of Zoutendijk [31], the projected gradient method of Rosen [26][27], and the penalty formulations of Fiacco and McCormic [9], and Courant [6]. The discussion shows that the incomplete relaxation and multistep concepts have application to constrained optimization any time a direction following algorithm is used.

Next, a new algorithm is suggested that bears strong resemblance to several of the above methods. It imposes a penalty formulation quite like Fiacco and McCormic [9] and Courant [6] discuss; but inside the feasible region it behaves much like the methods of Zoutendijk [31]

and Rosen [26][27]. Finally, by minimizing the penalty formulation, it ricochets back into the feasible region from a constraint similar to what the method of Greenstadt [11] does. The concept is presented as a step-by-step algorithm and is discussed to some depth.

Next, to demonstrate the applicability of nonlinear programming to "real world" problems and to investigate the efficiency of the proposed algorithm, a typical management problem is attacked. Basically there are five products that can be produced that have quadratic effects on the corporate profit. There are processing units that linearly impose constraints on the amount of each product produced. The objective of course is to maximize the profit while not violating the constraints. The problem, which qualified as a quadratic programming problem, is solved using the suggested algorithm. The algorithm is found to be quite effective and efficient terminating at an "acceptable" solution rather quickly. Optimality is assumed when the gradient is nearly normal to an active constraint or when it is equal to 0.

The method does have the disadvantage that it cannot "zero in" on the exact solution. As the gradient approaches being normal to the hyperplane, the method has trouble finding a direction that leads back into the feasible region. Suggestions are made to improve this. For example the use of a projected gradient as a terminating procedure should help a great deal. In general, the concept appears to be quite promising.

This dissertation shows that incompleterelaxation and multistep

concepts do indeed have applciations both to unconstrained and constrained optimization problems. Further, it shows how these concepts can be used to improve the efficiency of some well known optimization procedures to the point of showing that some of these procedures already employ these techniques. Finally, and perhaps most importantly, it demonstrates that nonlinear programming is not just a mathematical toy for exercising the mind; but it is a field that can and should be used to solve many real world problems as it often can model the real world quite accurately.

In pursuit of these goals, many fertile areas are discovered, mentioned or discussed and then bypassed. These are areas that should be explored in further research. Some of the more important areas are:

(1) The algorithm for the solution of simultaneous equations developed in Chapter II is quite promising and is being explored further. A research proposal is now in the hands of the Air Force Office of Scientific Research for funds to aid in further research.

(2) In Chapter II, quite often k = 2 was a better method than k = 3. This was attributed to the fact that for two steps the directions were always orthogonal but not for three steps or to the fact that in quadratic problems the search may tend to be planar. This should be resolved so more accurate determination of an optimal k may be obtained.

(3) The second suggestion for the determination of multistep multipliers was $C_i \alpha(H^{-1}(X)\nabla(X), u_i)$. The problem here was the determination of an inverse at each stage. There are several other

things that can be done. The use of H(X) instead of $H^{-1}(X)$, for example, was mentioned. The use of one or two steps of an iterative type procedure for inverting a matrix could also be used. This suggestion seems to offer much potential, so this should be explored.

(4) In researching the second suggestion, it was found that all these methods tend to go slowly at first and then make a tremendous jump to the optimum. This should be studied to see if it can be used as an ending routine for another method.

(5) In Chapter II under the third suggestion, it was proposed that incomplete relaxation or multistep concepts be used on sectioning or one at a time search procedure. Good intuitive arguments can be developed for its success so this should be examined.

(6) In Chapter II under the fourth suggestion, reference is made quite often to the comparison of searching instead of doubling in the acceleration step of pattern search; but no actual comparison is made. This should be explored, as it could give good insight into the advantages and disadvantages of such a search.

(7) The fifth suggestion mentions assuming the step size constant for two or more iterations in the search stage to conserve experiments, when costly experiments are necessary. This should be studied.

(8) The sixth suggestion discusses in great depth the application of dynamic programming methods and coordinate directions to unconstrained optimization. Gradient directions are only mentioned in passing because of algebraic difficulty. This should be explored

further and generalized results developed as with coordinate directions.

(9) In dealing with the acceleration mechanism of multistep techniques (especially on non-convex problems), one problem is efficiently determining the step to an optimum. An arbitrary bound on the search region was used in this research, but a reverse Fibonacci search or some other procedure may be more efficient. This should be examined further.

(10) The last method discussed in Chapter II involved one step toward the solution of the simultaneous set of linear equations $HC = -\nabla(X)$. Sometimes there is a very large step at first and sometimes it is not as large as it should be. A heuristic rule could be developed that would allow 2 steps when profitable, but only one if the first is a large one. This should be examined.

(11) Chapter III opens by discussing the application of incomplete relaxation and multistep concepts to classical constrained optimization procedures but never does any real applying. Some practical experience should be obtained in this area.

(12) The penalty algorithm proposed in Chapter III was left in a rather crude state. Suggestions were made for improvement such as acceleration mechanisms and zeroing in routines, but nothing was actually done. This should be polished into a complete algorithm as it shows considerable potential.

(13) In Chapter II, the underrelaxation multiplier was always chosen as ·8. A study could be made to attempt to find if there is

a better multiplier.

These thirteen areas are, of course, only a partial list of subjects that merit further research. The results as discussed along with these future areas should make a substantial contribution toward better usage of optimization techniques in real world problems which naturally is the real test the research must eventually face.

Finally, of the many areas compared, there are some that presently seem to be the most important. These are:

(1) The discovery that underrelaxation is better than complete relaxation.

(2) The discovery that "good" multistep procedures are better than any one step procedure.

(3) The development of a dynamic programming approach for the solution of successive quadratic approximations.

(4) The development of a procedure for the solution of near singular simultaneous linear equations.

(5) The development of a new procedure for constrained optimization.

BIBLIOGRAPHY

1. Ackoff, Russell L. and Sasieni, Maurice W., Fundamentals of Operations Research, John Wiley and Sons, Inc., New York, 1968.

2. Beltrami, Edward J., An Algorithmic Approach to Non Linear Analysis and Optimization, Academic Press, New York, 1970.

3. Beveridge, Gordon S. and Schnechter, Robert S., Optimization: Theory and Practice, McGraw-Hill Book Company, New York, 1970.

4. Canon, Michael D., Cullum, Clifford D., and Polak, Elijah, Theory of Optimal Control and Mathematical Programming, McGraw-Hill Book Company, New York, 1970.

5. Chung, An-min: "Linear Programming," Charles E. Merrill Books, Inc., Columbus, Ohio, 1963.

6. Courant, R., "Variational Methods for the Solution of Problems of Equilibrium and Vibrations," Bull. Am. Math. Soc., Vol. 49, 1943.

7. Craig, E. E. and Levy, A. V., "Study on a Supermemory Gradient Method for the Minimization of Functions," Journal of Optimization Theory and Applications, Vol. 4, No. 3, March, 1970.

8. Fadeev, D. K. and Fadeeva, V. N., Computational Methods of Linear Algebra, Translated by Robert C. Williams, W. H. Freeman and Company, San Francisco, 1963.

9. Fiacco, Anthony V. and McCormic, Garth P., Non-Linear Programming: Sequential Unconstrained Minimization Technique, John Wiley and Sons, Inc., New York, 1968.

10. Fletcher R. and Powell, M. J. D., "A Rapidly Convergent Descent Method for Minimization," Comp. J., Vol. 6, No. 2, 1963.

11. Greenstadt, J. L., "A Ricocheting Gradient Method for Non-Linear Optimization," Journal of SIAM Applied Mathematics. Vol. 14, No. 3, May, 1966.

12. Ghare, P. M., "Multi-Step Gradient Methods for Non-Linear Programming - Part I - Unconstrained Optimization," A paper presented to the 7th Mathematical Programming Symposium. The Hague, Netherlands, September, 1970.

13. Ghare, P. M., and Turner W. C., "An Efficient Method for the Solution of Near Singular Simultaneous Linear Equations," Unpublished working paper, IEOR Dept., VPI & SU, Blacksburg, Virginia.

14. Gass, S. I., Linear Programming: Methods and Applications, McGraw-Hill Book Company, New York, 1958.

15. Geoffrion, A. M., "Strictly Concave Parametric Programming Part I: Basic Theory," Management Science. Vol. 13, 1966.

16. Gue, Ronald L. and Thomas, Michael E., Mathematical Methods in Operations Research, The Macmillan Company, London, 1968.

17. Hadley, G., Linear Programming. Addison-Wesley Publishing Company, Reading, Mass., 1962.

18. Hadley, G., Nonlinear and Dynamic Programming, Addison Publishing Company, Reading, Mass., 1964.

19. Hestens, M. R. and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," J. Res. Natl. Bur. Standards, Vol. 49, 1952.

20. Hillier, Frederick S., and Lieberman, Gerald J., Introduction to Operations Research, Holden-Day, Inc., San Francisco, 1967.

21. Hooke, R. and Jeeves, T. A., "'Direct Search' Solution of Numerical and Statistical Problems," J. Assn. Comp. Mach., Vol. 8, No. 2, April, 1961.

22. Kunzi, Hands Paul, Frelle, Wilhelm and Oettli, Werner, translated by Levin, Frank, Nonlinear Programming, Blaisdell Publishing Company, Waltham, Mass., Toronto, 1966.

23. Llewelyn, Robert W., Linear Programming, Holt Rinehart and Winston, New York, 1966.

24. Mangasarian, Olvi L., Nonlinear Programming, McGraw-Hill Book Company, New York, 1969.

25. Nemhauser, George L., Introduction to Dynamic Programming, John Wiley and Sons, Inc., New York, 1966.

26. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints," J. Soc. Indust. Appl. Math ., Vol. 8, No. 1, March, 1960.

27. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints," J. Soc. Indust. Appl. Math., Vol. 9, No. 4, December, 1961.

28. Schmidt, J. W., Functional Analysis in Management Science and Systems Engineering, Richard D. Irwin, Inc., In Press.

29. Shah, B. V., Buehler, R. J., and Kempthorne, O., "Some Algorithms for Minimizing a Function of Several Variables," J. Soc. Ind. Appl. Math., Vol. 12, No. 1, March, 1964.

30. Wilde, Douglas J. and Beightler, Charles S., Foundations of Optimization, Prentice Hall, Inc., Englewood Cliffs, N. J., 1967.

31. Zoutendijk, G., Methods of Feasible Directions. A Study in Linear Programming, Elsevier, New York, 1960.

32. Smith, James R., The Development of Search by Successive Quadratic Approximation and Cost-Related Termination Criteria for Application to Simulation Models, Doctoral Dissertation, IEOR Dept., VPI & SU, Blacksburg, Va., 1971.

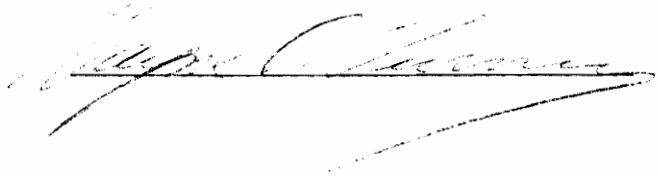
VITA

The author was born on May 7, 1942, in West Point, Virginia, to Mr. and Mrs. Percy Turner. He attended West Point High School until graduation in June of 1960.

Due to the encouragement and guidance of his brother, Mr. Kenneth L. Turner, the author entered Virginia Polytechnic Institute in the fall of 1960. He received his BSIE from Virginia Polytechnic Institute in June 1964 and began working for Modine Manufacturing Company of Racine, Wisconsin, as a manufacturing engineer. In February 1966, the author was appointed Manager of Manufacturing Engineering and Maintenance for the Buena Vista, Virginia, plant of Modine.

In September, 1967, the author reentered Virginia Polytechnic Institute to work toward his MSIE. Since that time, he has worked under a research grant, taught as an assistant, and taught for two years as an Instructor. He received his MSIE from VPI in 1970.

The author is a member of Alpha Pi Mu honorary fraternity, the American Institute of Industrial Engineers, and the Operations Research Society of America.



DEVELOPMENT AND APPLICATION OF MULTISTEP COMPUTATIONAL TECHNIQUES FOR CONSTRAINED AND UNCONSTRAINED MATHEMATICAL FUNCTIONS

by

Wayne C. Turner

ABSTRACT

The application of incomplete relaxation and multistep concepts in the usage of steepest descent methods for the solution of simultaneous linear equations is recognized in the literature. There has been research in the area with very favorable results. Only recently, however, has there been any recognition of the fact that these concepts can be extended to unconstrained optimization problems, and by penalty formulations, also to constrained optimization problems. This holds true not only for steepest descent methods but also for any other "improving direction."

In the discussion of the application of incomplete relaxation and multistep concepts to mathematical functions, very little has been accomplished in the mechanical applying of these ideas. The primary goal of this research, therefore, is to study these concepts and learn a significant amount about them. Algorithms are developed demonstrating the usage of incomplete relaxation and multistep concepts for unconstrained optimization on two directions - coordinate and gradient directions. Discussion of the performance of these methods follows in an attempt to choose some of the better ones. Ten such promising methods are selected and are applied to some

complicated unconstrained functions to investigate the adaptability of these methods.

Next, the application of these methods to constrained optimization is examined. Some well known constrained procedures are discussed to show how these applications can be made. A new algorithm for constrained optimization is then developed and used to solve a real world problem both to demonstrate the use of this algorithm and to show that nonlinear programming does have applications to the "real world." Finally, some areas that need further research are mentioned and discussed.

The results, in general, are quite promising. For unconstrained optimization, underrelaxation yields faster convergence than did complete relaxation for all the problems examined. The difference is highly significant; but this is not startling as the same is true in the solution of simultaneous linear equations. Multistep methods are also quite efficient providing some way of efficiently determining or approximating the multistep multipliers can be obtained. In fact, an efficient multistep method is better than any one step method for these problems.

This research shows that out of the many algorithms tried only a few seem to offer the efficiency and adaptability that is needed. These methods are isolated so that their usage may be facilitated. Each of these requires its own development and to some extent stands alone.

complicated unconstrained functions to investigate the adaptability of these methods.

Next, the application of these methods to constrained optimization is examined. Some well known constrained procedures are discussed to show how these applications can be made. A new algorithm for constrained optimization is then developed and used to solve a real world problem both to demonstrate the use of this algorithm and to show that nonlinear programming does have applications to the "real world." Finally, some areas that need further research are mentioned and discussed.

The results, in general, are quite promising. For unconstrained optimization, underrelaxation yields faster convergence than did complete relaxation for all the problems examined. The difference is highly significant; but this is not startling as the same is true in the solution of simultaneous linear equations. Multistep methods are also quite efficient providing some way of efficiently determining or approximating the multistep multipliers can be obtained. In fact, an efficient multistep method is better than any one step method for these problems.

This research shows that out of the many algorithms tried only a few seem to offer the efficiency and adaptability that is needed. These methods are isolated so that their usage may be facilitated. Each of these requires its own development and to some extent stands alone.