A User's Guide
To
Nonlinear Optimization Algorithms

by

J.E. Dennis, Jr.[1]

Technical Report 83-19, August 1983.
(Revised February 1984)

---

[1]Mathematical Sciences Department, Rice University, Houston, Texas 77251. Research sponsored by DOE DE-AS05-82ER13016, ARO DAAG-79-C-0124, NSF MCS81-16779. This paper was prepared by invitation for delivery at The 22nd IEEE Conference on Decision and Control, San Antonio, December 1983.

Abstract

The purpose of this paper is to provide a user's introduction to the basic ideas currently favored in nonlinear optimization routines by numerical analysts. The primary focus will be on the unconstrained problem because the main ideas are much more settled. Although this is not a paper about nonlinear least squares, the rich structure of this important practical problem makes it a convenient example to use to illustrate the ideas we will discuss. We will make most use of this example in the first three sections which deal with the helpful concept of a local modeling technique and the attendant local convergence analysis. Stress will be put on ways used to improve a poor initial solution estimate since this is one of the keys to choosing the most suitable routine for a particular application. This material is covered in the rather long Section 4. The discussion of the constrained problem in Section 5 will be a brief outline of the current issues involved in deciding what algorithms to implement. Section 6 is devoted to some concluding remarks including sparse comments on large problems.

Key words

Numerical methods, optimization.

1. Introduction.

The problem of estimating parameters by nonlinear least-squares is well suited to illustrate the general ideas common to all the algorithms that we will discuss. In particular, we might be given a model and some data

$$y=x_1+x_2\cos(x_3 t), \text{ and } (t_i,y_i),\ i=1,2,\ldots n.$$

Our problem is to determine the parameter vector $x=(x_1,x_2,x_3)^T$ to best fit the data. We define R to be the residual vector function that maps parameter space $\mathbf{R}^p$ into residual space $\mathbf{R}^n$, and whose ith component is

$$R_i(x) = x_1+x_2\cos(x_3 t_i) - y_i.$$

The unconstrained problem is to minimize

$$f(x) \equiv \frac{1}{2}R(x)^T R(x) = \frac{1}{2}\sum_{i=1}^{n}[R_i(x)]^2. \tag{1.1}$$

Most practical problem have solutions and we can usually find one, say x_*. One point is that the user often claims to want a global minimizer of f, a literal solution to (1.1), and there is really no way to write a library program to handle general problems of this class, use only finitely many function evaluations and no expensive derivative bounds, and be sure of finding the global minimizer. Instead, the algorithms discussed here try to find a local minimizer, which is what it sounds like: there is a ball around x_* in which no x gives a smaller value of f. We will also speak of strong local minimizers x_*. This just means that there is a ball in which no x gives a value of f as small as $f(x_*)$. Many of the routines will return a warning if the problem is sufficiently ill-conditioned so that x_* is not a strong local minimizer. This doesn't sound very satisfactory, especially since optimizers usually state the definition of a local minimizer very formally and use ε, which connotes a small number, to denote the radius of the ball. In practice, the size of the ball is most affected by the appropriateness of the model, the conditioning is dependent on how nearly redundant the parameterization is, and the particular x_* found is usually the most closely related to, if not the closest to, the initial or nominal estimate of the parameter vector x. Users only worry about whether x_* is the global minimizer if it is not the answer they expected to find. Some

interesting algorithms are being developed to approximate global minimizers, and the continuation methods can be a real help in finding the *right* local minimizer. See [1], [20], [39], [42], [78], [79], [86].

We will generate a sequence of iterates $\{x_k\}$ by the following technique: At each iteration, we build an easily solved problem that models (1.1) for x near the current iterate x_c. We then solve the model problem and hope that its solution will be an acceptable next iterate. If not, then we use one of the strategies of Section 4 to modify that solution to provide the next iterate x_+.

The standard form of the model for unconstrained minimization is a quadratic function written in terms of the step s from x_c:

$$q_c(x_c+s)=f(x_c)+g_c^Ts+\frac{1}{2}s^TH_cs. \tag{1.2}$$

Here, $g_c=\nabla q_c(x_c)\in\mathbf{R}^p$ is the gradient at x_c of the quadratic q_c. The gradient of a real-valued functional φ is the column of partial derivatives of φ; its ith element is $\partial_i\varphi$. Strictly speaking, it is the transpose of the derivative φ', which is a row vector. The Hessian matrix consists of the second derivatives of a functional φ; its i,jth element is $\partial_{ij}\varphi$. The model Hessian H_c is a $p\times p$ symmetric matrix. If H_c is positive definite, $v^TH_cv>0$, then the model problem has a unique global minimizer at the quasi-Newton point x_+ obtained from solving the $p\times p$ linear system:

$$H_cs^{qN} = -g_c, \tag{1.3}$$

and then

$$x_+ = x_c+s^{qN}.$$

If the model Hessian is taken to be positive definite, then the Cholesky factorization technique can be used on (1.3). This algorithm uses half the work and storage of Gaussian elimination, and it is completely stable. Furthermore, all the methods take g_c to be either $\nabla f(x_c)$, or a finite-difference approximation to it, and so a positive-definite model Hessian ensures that the quasi-Newton step is at least a step in a descent direction, a direction for which f has a negative directional derivative at x_c.

$$\frac{df(x_c+ts)}{dt}|_{t=0}=g_c^Ts=-g_c^TH_c^{-1}g_c|_{s=s^N}<0.$$

In Section 4, we will see ways of dealing with singular or indefinite model Hessians.

Another useful terminology is that of the Jacobian of a vector-valued function of a vector variable. The Jacobian J_c of R at x_c is just the $n\times p$ matrix whose ith row is $R_i^i(x_c)$. In general, the Jacobian of the gradient is the Hessian. The chain rule gives

$$\nabla f(x_c) = f(x_c)^T = J_c^TR(x_c). \tag{1.4}$$

1.1. Derivatives and Noise.

There are two basic assumptions that underlie the choice of methods that we will discuss. The first is that the objective function is expensive to compute, and the second is that the user doesn't want to furnish any more information than necessary to solve his problem. To illustrate the first, if the functions involved are cheap to evaluate, then why not lay down a fine grid and just find the minimizer on the grid in the feasible set? To illustrate the second, if routine A asks for only objective function values and B asks for function, gradient, and Hessian values, then B will get very little use. On the other hand, if A fails too often, then it will also get very little use. The numerical analysts' client groups are just like all clients, they want something available that does the job and is cheap and easy to use.

Most users are understandably reluctant to find and to code the partial derivatives required for the Jacobian of R or the gradient of f; this is true even in problems for which the derivatives are straightforward. It is clear from the discussion before this subsection that the algorithms will need partial derivatives. It is standard practice now for optimization routines to furnish subroutines that calculate derivatives by finite differences. Given the option, most users will take it, and it is probably the safest thing to do, since most people are likely to make an error in either taking or coding the relevant partials. In fact, if the user does furnish his own derivatives, many library routines will begin the computation by using finite differences to check for errors in the user supplied routine. There is work underway to use symbolic differentiators to furnish derivatives, but there do not seem to be

any widely available minimization codes of this type. [63]

We will mention variable-metric methods in the section on local models. These fascinating generalizations of the secant method furnish cheap derivative approximations. They are excellent for Hessians but they are probably not accurate enough to be used in gradient approximations, since $\nabla f(x_*)=0$ is a necessary condition for a solution. Broyden's method [6] is a well-known version of these methods for approximating Jacobians, but it is dangerous to use it for R' in the nonlinear least-squares problem since (1.4) can be interpreted as saying that at x_*, the residual vector must be orthogonal to the column space of R'. If the problem is $p\times p$ and of full rank, then there is no worry since the column space is all of $\mathbf{R}^p=\mathbf{R}^n$. Otherwise, the column space is at most a p-dimensional subspace of $\mathbf{R}^n$ and needs to be known accurately to avoid converging to a nonminimizer. This objection is irrelevant if the user can realistically use a convergence test based only on the final residuals.

Thus, for all gradient-related calculations, we strongly recommend using the finite-difference subroutine if one is provided, but there can be a problem if the user-provided function routine is less accurate than the finite-difference subroutine expects. The proper perturbation step h_i in the finite-difference calculation

$$\frac{f(x+h_i\varepsilon_i)-f(x)}{h_i} \simeq \partial_i f(x) \qquad (1.1.1)$$

is for h_i to perturb half as many digits in x as are accurate in f(x). (Here, we are using ε_i to denote the ith unit vector in x-space.) If h_i is too small relative to the accuracy of f, then evaluating the numerator of (1.1.1) will result in cancellation of all but the noise in the trailing digits of the two f-values. Division by h_i then makes this small noise louder by the magnitude of h_i. If a routine asks for the accuracy of your function evaluations, then it will probably want to use the accuracy information in this way. If it does not ask, then beware because it probably assumes that f is accurate to full precision.

If the user can provide only a few digits of accuracy in the function, then it is very useful to provide derivatives if possible. The derivatives need not even be as accurate as

the function, just more accurate than the finite differences will be. Users sometimes combine a long-precision version of a finite-difference optimization code with a short-precision function, use the default tolerances, and as a consequence, observe an expensive drawn-out lack of convergence. This is especially common in estimating parameters for differential equation models where the cost of the residual calculation increases dramatically with the accuracy requested. To alleviate this, it is well to remember that many of these problems can be manipulated to provide the relevant derivatives as a part of the residual calculation. See [73], p.470, for an example.

Most of the algorithms can be modified to need only a few significant digits, if the user is realistic about when to stop. A notable exception is the Nelder-Mead simplex algorithm ([3], [52], [32]) which really only needs enough accuracy to be able to trust the sign of the difference of objective function values at the iterates. Unfortunately, this algorithm is not efficient for more than about five variables, but it works surprisingly well for small problems. Glad and Goldstein [34] seems to be the only alternative if the problem has only one or two significant digits in the objective function. Some coworkers and the author are analyzing ways to modify the algorithms considered here to make them more robust for low noise levels. It does not seem to be a very difficult problem, but it is much too interesting to be finished quickly because it is leading us to a deeper understanding of the underlying concepts. See [17], [18], [84], [85].

2. Local Models

This section contains an explanation of the Newton, Gauss-Newton, and some quasi-Newton methods for (1.1) in terms of the local modeling ideas introduced above. The use of nonlinear least squares as an example is partly based on the number of interesting local models available. The Levenberg-Marquardt method will not appear until Section 4 since we view it as a natural modification of the Gauss-Newton local model to deal with singularity and poor initial guesses.

The Gauss-Newton model is only incidentally based on a local quadratic model. The actual philosophy behind the method is applicable to any parameter-selection problem for a nonlinear model in which the corresponding problem for a linear model is cheap enough to act as a local model. This way to view the Gauss-Newton method is to think of it as building a local affine model of the residual function R and then solving the linear least-squares problem for the Gauss-Newton step:

$$s^{GN} = argmin\frac{1}{2}[R(x_c)+J_cs]^T[R(x_c)+J_cs], \tag{2.1}$$

which is

$$= -[J_c^TJ_c]^{-1}J_c^TR(x_c)\ ,$$

if one chooses to use the normal equations to solve the linear problem. Here we have used argmin (expression) to denote the argument s that minimizes the expression.

Expanding (2.1) and rearranging terms, gives us the Gauss-Newton quadratic model as (1.2) with

$$g_c^{GN} = J_c^TR(x_c) = \nabla f(x_c), \text{ and } H_c^{GN} = J_c^TJ_c.$$

Note that H_c^{GN} is always at least positive semi-definite, and that it is available from derivative information already gathered for the gradient.

Newton's method is the essence of the quadratic modeling idea. It chooses the model derivatives to match the gradient and Hessian of the objective function f at x_c. The problem is that the Hessian $f''(x_c)$ is often expensive to compute or to approximate by finite differences. The Hessian of the least-squares problem is:

$$f''(x_c) = H_c^N = J_c^TJ_c+\sum_{i=1}^{n}R_i(x_c)R_i''(x_c) \equiv J_c^TJ_c+S(x_c).$$

Thus we see that the difference between the local models used by the Gauss-Newton and Newton methods is just $S(x_c)$, the sum of the second-order information weighted by the corresponding residual. The theory tells us that if $f''(x_*)$ is nonsingular, then f''(x) will be positive definite near a strong local minimizer of f, but it may not be positive definite early in the iteration. Thus, Newton's method has the disadvantages of computing the most

expensive part of the Hessian and of having to worry about negative as well as zero eigenvalues. Since the Newton model of f is probably more accurate, one would expect better absolute performance, even if it were not enough better to justify the extra difficulties. As we will see in Section 4, negative curvature and singularity at iteration points are not such difficult problems as they once were (see [30], [51]).

The Gauss-Newton method pays a penalty for ignoring $S(x_c)$ when it is applied to nonlinear problems with large residuals. There is a simple example in [12] of a problem for which x_+ is always further than x_c from x_* for choices of x_c close enough to x_*. For such problems, it may never be possible to avoid the expense of using the techniques for proceeding from a poor solution estimate. In some applications areas, the Gauss-Newton method is almost never used for this reason. Some alternatives are the so-called variable-metric methods [16], [25], [32]. These very effective and convenient Hessian approximation techniques are generalizations of the way the 1-dimensional secant method approximates derivatives in that the new model Hessian H_+ is chosen to satisfy the secant condition (also called the quasi-Newton equation): $H_+(x_+-x_c)=g_+-g_c$. Of course, H_+ is not determined by this condition alone for $p>1$. The different methods can be viewed as resolving this ambiguity for H_+ by choosing it as near as possible to H_c in some norm. The two most natural norms that try to mimic the scaling in the problem give the DFP, or Davidon [10], Fletcher-Powell [22] method, and the BFGS, or Broyden [7], [8], Fletcher [23], Goldfarb [35], Shanno [68] method. For these methods, it is not hard to maintain positive-definite model Hessians, and the Cholesky factors of H_+ can be gotten from those of H_c in $O(n^2)$ operations. See [16], [32].

The DFP and BFGS do get around the local convergence problems that the Gauss-Newton method may have, but they are not nearly as effective as that method for small residual problems. This seems reasonable because they don't make use of the structure of f''(x). Several people have considered ways to make use of the availability of H_c^{GN} to maintain a model Hessian of the form

$$H_c^A = H_c^{GN} + S_c, \tag{2.2}$$

where S_c is a variable-metric approximation to $S(x_c)$. We will refer to this as an augmented Gauss-Newton method. See [13], [14], [32].

3. Local Convergence.

In this section, we will consider the following question for each of the local modeling techniques of the last section:

If x_c is near enough to x_* and x_* is a strong local solution to the problem, then can we be sure that x_+ defined by the solution to the model problem will be a better approximate solution than x_c?

This is a basic question that effects the way the algorithm is implemented to improve poor initial guesses. Also, the analysis that answers this question usually tells something about the rate of improvement that can be expected in the last few iterations. Someone once said that this local convergence property is not important for Newton's method because most of the work in carrying out the iteration is expended in getting close enough to be able to take the full step to x_+. This is somewhat like saying that jet travel is not an important part of a two-week trip to Europe because it occupies so little of the time.

It is not in keeping with our purpose here to state the theorems that have been given to answer this question for the methods of interest. Instead, we will give an informal comparison. We will consider only the case where x_* is a strong local minimizer in the sense that $f''(x_*)$ is positive-definite. The Gauss-Newton method requires the least smoothness to give an answer. It requires that R' be Lipschitz continuous, but the method is locally convergent only if the residuals are small enough or linear enough, and it is more technical than we wish to discuss here. Newton's method is locally convergent if R'' is continuous, but this degree of smoothness only ensures that the convergence is superlinear rather than the second-order convergence for which Newton's method is known. A better comparison can be made if we assume that R'' is Lipschitz continuous. In this case, the Newton

method, the BFGS, DFP, and the augmented Gauss-Newton given in [13] are all locally convergent, regardless of residual size. The Gauss-Newton method converges locally if $S(x_*)$ is smaller than $J(x_*)^TJ(x_*)$. Loosely speaking, if $e_k=\|x_*-x_k\|$ and e_{k+1} is defined similarly, then $e_{k+1}\simeq re_k$, where r is the size of $S(x_*)$ relative to $J(x_*)^TJ(x_*)$. [4], [80]. If $S(x_*)$ is the zero matrix, then the Gauss-Newton method is as fast as Newton's method, i.e., $e_{k+1}=O(e_k^2)$, which doubles the number of significant digits at each step. The other methods are all superlinear, i.e., $e_{k+1}=o(e_k)$.

4. Improving a Poor Solution Estimate.

This section is to discuss the question of what to do if we follow the procedure of building a local model, say by one of the techniques given above, solve for the corresponding quasi-Newton step, and then find that x_+ gives a worse approximate solution than x_c. In fact, it is clearly not sufficient in theory just to be sure that $f(x_+)<f(x_c)$, since is easy to construct a nicely convex 1-dimensional example to show $f(x_k)$ converging monotonically down to $2f(x_*)$ and $\{x_k\}$ converging to a point between x_0 and x_*. To avoid this, we need to know how x_+ must improve on x_c. After that, the rest is usually straightforward; if the quasi-Newton step, or some subsequent trial step, does not give sufficient improvement, then we usually assume that our trial step was too long, and we try a shorter step. There are some contexts in which we actually try a longer step if a trial step fails in a way that indicates that the function is still decreasing rapidly in the direction of the trial step.

The backtracking strategy of always taking a shorter step if a trial step is unsuccessful, implies the following implicit assumption, which we will state in terms of a local quadratic model, on the compatibility of the test for sufficient improvement and the local modeling technique: the method provides $q_c(x_c+s)$ accurate enough near x_c so that for a sufficiently small step s, if x_c+s gives sufficient improvement over x_c as an approximate minimizer of q_c, then it will also give sufficient improvement as an approximate minimizer of f.

It is worth noting that if f is allowed to increase for some iterations, then much faster convergence is occasionally observed. In the author's experience, such strategies are slower and less reliable over the long haul. Also, intuition suggests a greater likelihood of finding the wrong local minimizer to go with the user-supplied initial estimate.

4.1. Step Acceptance Tests.

The Goldstein [37]-Armijo [2] conditions are the most basic tests used to determine whether a step is a sufficient improvement to warrant taking it. There are really only three concerns: the step must be in a good direction, it must not be too long, and it must not be too short. We only require the direction of s to be a descent direction, $g_c^T s<0$. The condition we impose that the step s not be too long is really a requirement that a longer step must result in a proportionally greater reduction in f. This is to rule out the case in which we stand on the side of a valley and take a step so long that we go up the other side of the valley almost to the same altitude as the point we left. The problem here is that if we repeat the process making less and less downward progress each time back and forth across the valley, then we can flatten out by converging to two limit points, one on each side of the valley. The test to prevent this is

$$f(x_c+s)\leqq f(x_c)+\alpha g_c^T s,\ 0<\alpha<1. \tag{4.1.1}$$

This says that $f(x_+)$ should be below a line between the horizontal and the line tangent to f at x_c in the direction of s. Notice that if f is continuous along a descent direction from x_c, then there is always an interval of steps s in that direction that all satisfy (4.1.1). The constant α is usually taken to be .0001, so this is not a very stringent requirement. It is interesting to note that if f is a quadratic along the direction of s, then $\alpha=.5$ goes right through the minimizer of f along the direction. Thus, one would probably never use a larger value than .5, since for the model problem, any step slightly longer than the best possible would be unacceptable.

There are several ways to make sure the step is not too short. Here we are trying to guard against our valley walker repeatedly going half the remaining distance down toward a

fixed spot on his side of the valley, so that he satifies (4.1.1), but he converges because the sum of the lengths of his steps converges, not because he reaches the bottom of the valley. The most straightforward test is

$$f(x_c+s)\geqq f(x_c)+\beta g_c^T s,\ 0<\alpha<.5<\beta<1. \tag{4.1.2}$$

This just inserts another line below (4.1.1) and above the line tangent to f at x_c. Since f(x) starts out below both lines (4.1.1&2), if f is continuous, then f either decreases forever in the given direction, or else it eventually goes above the top line, in which case there must be an interval of steps s that all satisfy (4.1.1&2). The reason for choosing $\beta>.5$ is the same as the reason given above for choosing $\alpha<.5$.

It is easy to see which way to adjust the length of a step in a fixed direction using these tests: if (4.1.1) fails, then there is certainly a shorter step that will satisfy both tests, and if (4.1.1) passes and (4.1.2) fails, then there is a longer step that satisfies both tests. It is (4.1.2) that is the most troublesome to satisfy and to my knowledge, (4.1.2) is not implemented in any generally available software. Still the pair of conditions taken together have the very interesting property that they are totally independent of the way s is chosen, as long as it is in a descent direction. It can be shown that if $\{s_k\}$ satisfies both conditions, then under very weak conditions on f,

$$\lim_{k\to\infty}\nabla f(x_k)^T s_k/\|s_k\|=0. \tag{4.1.3}$$

See [16], [37], [54], [82], [83]. If the local method will just take responsibility for bounding the step and gradient directions away from orthogonality, then (4.1.3) implies

$$\lim_{k\to\infty}\nabla f(x_k)=0. \tag{4.1.4}$$

Furthermore, if the level set of x_0 defined by

$$L(x_0)=\{x\in R^p: f(x)\leqq f(x_0)\},$$

is bounded and f' is continuous, then $\{x_k\}$ has a subsequence that converges to a zero of $\nabla f(x)$. We could go on adding conditions and drawing ever stronger conclusions, but if f''(x) is positive definite on $L(x_0)$, i.e., f is strictly convex there, then $\{x_k\}$ converges to x_*, a local minimizer of f.

4.2. Alternate Step Tests.

In the previous subsection, we saw tests for sufficient improvement that didn't depend at all on the algorithm used to choose the step, as long as the step was made in a descent direction. The argument could be made that we may as well use more information from the algorithm, if it will help. The simplest scheme is a safeguarded backtrack. If the full step to the solution of the model problem satifies (4.1.1), then we take the step and ignore (4.1.2). If (4.1.1) is not satisfied, then we know that there is a shorter step that is neither too long or too short. We decrease the unsuccessful step by a factor ρ, usually $.1<\rho<.5$, and test (4.1.1). This process is continued until (4.1.1) is satisfied. Thus the step is not too long, and we trust the process we used to decrease the step to insure that it is not too short. See [2], [16], [32], [54].

This procedure is even more interesting when we replace (4.1.1) by another test more closely tied to the local modeling idea. First, we can rewrite the test in the following way:

$$\frac{f(x_c+s)-f(x_c)}{g_c^T s} \geqq \alpha. \tag{4.2.1}$$

The numerator is the reduction in f that results from the step s. Now think of the denominator as the reduction in an affine model of f in taking the step s from x_c, i.e., the predicted reduction from (1.2) with H_c taken to be the zero matrix. Thus, (4.2.1) suggests another test to replace (4.1.1):

$$\alpha \leqq \frac{ared}{pred} \equiv \frac{f(x_+)-f(x_c)}{q_c(x_+)-q_c(x_c)} = \frac{f(x_+)-f(x_c)}{g_c^T s+\frac{1}{2}s^T H_c s}. \tag{4.2.2}$$

See [24], [64]. A seductive interpretation of this test is that, since the predicted reduction is always going to be relatively large, if this test passes with say $\alpha=.1$, then we must be inside the region about x_c where the model adequately represents f. Since s was not too short for the model, we assume that it is not be too short for f, and we ignore (4.1.2). If the actual reduction, ared, is so small that (4.2.2) fails, then we assume that the step is too long and we use a safeguarded backtracking scheme.

It is natural to make the same sort of analog to (4.1.2), (but we state it in negative form)

$$\frac{ared}{pred} \geq \beta > .1, \tag{4.2.3}$$

as an indication of whether a longer step should have been tried. In fact, if the prediction is much more conservative than the actual reduction, then it might be that the modeling technique is not really valid in the region in question and much larger reductions are possible by taking larger steps. For example, f might have significant negative curvature in a step direction in which the model curvature is positive. We will say more later about what use some algorithms make of this information.

Finally, there is one more type of replacement for (4.1.2) that is used in some algorithms. To motivate it, notice that an intuitive interpretation of the test is that a step is not too short if the function has already started to turn up enough so that $f(x_c+s)$ is above the line defined by the right-hand side of (4.1.2). The new version is especially intended for those problems where the computation of f' is cheap if f is being computed at the same time. In this test, we measure whether the function has started to turn up at x_c+s by comparing the directional derivative of f along s at that point to its value at x_c. If

$$\beta f(x_c)s \leq f(x_c+s)s, \quad \beta = \frac{1}{2}, \tag{4.2.4}$$

then the new directional derivative is at least twice as large as before, but it may even be zero or positive. If the new directional derivative is zero, then we have stepped to a critical point for f along s. In most practical cases, the way s has been chosen will make the critical point be a local minimizer of f along s. This is sometimes called a *perfect step* and the only reason we don't try to take perfect steps is that they are too expensive to justify. For some algorithms, like conjugate-direction methods [32], [38], they may be worth the expense. In order to control the degree of perfection of the step, Gill, Murray, and Wright [32] suggest the following test:

$$|f(x_c+s)s| \leq -\beta f(x_c)s.$$

This test also includes an implicit test for s too long, but using it with (4.1.1) is more

assurance that x_c+s is a suitable critical point.

Finally, there is one more point to make about (4.2.4). We said in Section 2 that maintaining positive-definite model Hessians is not hard for the DFP and BFGS methods. If H_c is symmetric and positive definite, then a standard easy result is that H_+ will be also if, and only if, (4.2.4) holds for $\beta=1$. See [16]. This is less stringent than for $\beta=.5$, and some variable-metric routines enforcement it. An alternative to enforcement is to keep the same model Hessian for another iteration when the test fails.

4.3. Backtracking

In this subsection, we will discuss briefly the adjustment of the lengths of the trial steps. We mentioned at the beginning of Subsection 4.2 that if the quasi-Newton step, s^{qN} in (1.3), fails to satisfy (4.1.1), then we decrease the length of s by a factor $.1\leqq\rho\leqq.5$. The choice of ρ is again by the local modeling idea.

When we attempt a step s and (4.1.1) fails, then we have at least one new piece of function information, $f(x_c+s)$, which enables us to correct the model along the direction of s to reflect the fact that the surface is higher at the trial point than we thought it was. It seems sensible to try next a step the length of the step to the minimizer of the corrected directional model. If the failure was on the step to the minimizer of the original quadratic model, then we correct the quadratic model along the line. However, if the next step fails, then it is common practice to use the three function values and one directional derivative to build a cubic model along s for use in choosing the next value of ρ. Cubic models have a real advantage in modeling negative curvature. In fact, if the gradient is being computed along with the function, then the extra directional derivative obtained at failed points allows the use of a cubic model after the first failure. Of course, more sophisticated models can be used to adjust the steplength, and they may be worthwhile for some problems.

The candidate value of ρ provided by the above schema can easily be seen to be much too extreme sometimes, and so it is common to take the nearest point to ρ in the interval $[.1, .5]$. If one were using one of the tests that sometimes call for a longer step than s^{qN},

then the interval could be adjusted to say [.1, 5].

Finally, some routines use step-length information from the current step to preset an initial trial step-length for the next step before its direction is even computed. The rationale is that if we have just completed a successful step, then the agreement between the actual and predicted reduction can be used to forecast a good step length to try next time in order to reduce the number of unsuccessful trials needed to find the next step. For example, the rule might be that if ared/pred is near the lower limit in (4.2.3), then we were lucky to pass the test and we should not try such a long step next time. If the local model seems to be doing well in this region as evidenced by a ratio near 1, then perhaps we should allow ourselves the freedom to take a step up to twice as long as we would have allowed ourselves this time. The same increase might be reasonable if the ratio is large enough to indicate that the modeling technique isn't very accurate in this region because the function decreased much more than predicted. These ideas seemed to have been developed for the Harwell library [24], [55], [64]. They are universally used in the trust-region step choice algorithms discussed in the next subsection, but they have not been used to the author's knowledge with the line-search algorithms also discussed below. The disadvantage of such a scheme is that steps shorter than s^{qN} might be taken on some iterations where a quasi-Newton step would have passed all the tests. All algorithms make some provision for limiting the length of a trial step; this is especially important early in the computation when raw steps s^{qN} are often large enough to cause overflow in the user-supplied function routine.

4.4. Step Direction.

So far, s^{qN} is the only specific search direction we have mentioned and a negative directional derivative at x_c is the only assumption we have made on s. In this subsection, we will outline the line search method, the locally constrained optimal step, and the dogleg step. Each has its advantages and is to be found in standard libraries. The locally constrained optimal step and dogleg step are often called trust-region methods.

The line search method makes all its trial steps in a fixed direction. The simplest and most widely known way to choose the step direction is to use s^{qN}. This requires only the $p \times p$ linear system (1.3) to be solved at each iteration, which is a big advantage for the variable-metric methods and for problems with large sparse Hessian matrices. It has certain theoretical advantages as well. In particular, if the solutions of the local models are independent of a particular class of basis changes, then so is $\{x_k\}$. This is a useful property since many times in practice, problems are difficult computationally only because of the units in which the variables are given. In addition, the best convergence theorems for the variable-metric methods without any explicit assumptions on the nearness of the initial estimate are for line search methods [58]. Line search methods have no intrinsic way to deal with singularity and they can require many function evaluations to find a successful step, if the initial step direction happens to be a bad one. The routines from the NAg Library are based on this technique; there are a full suite of optimization codes including finite difference and variable metric versions.

The locally constrained optimal step, lcos, is a name that can be applied to the step choice used by a class of methods that includes the Levenberg [41]-Marquardt [44] method. The rationale is simple: if we agree to try a step of length λ, then the optimal step subject to that local constraint is the solution to:

$$\min f(x_c+s) \ , \ \text{for } \|s\| \leq \lambda. \tag{4.4.1}$$

Of course, we can't solve (4.4.1) as a step in (1.1), so following the local model philosophy, we model (4.4.1) by

$$\min q_c(x_c+s) \ , \ \text{for } \|s\| \leq \lambda. \tag{4.4.2}$$

This step is not so difficult to compute as one might expect and it works well in practice, but a problem with the premise of (4.4.1) is the formulation of the local constraint $\|s\| \leq \lambda$. The bound λ came from looking at the behavior of the problem in one direction, and so the particular norm chosen needs to reflect the corresponding problem-dependent notion of length in all other directions as well in order to make the bound relevant. One could argue that if H_c is positive definite, then it reflects our best knowledge about the proper scaling of

the problem, but then the solution to (4.4.2) with the norm defined by the innerproduct $s^T H_c s$ is the step of length λ in the quasi-Newton direction, the line-search step. A counter argument could be that using the model Hessian in both roles is putting all our eggs in the same basket, or at least in baskets of the same shape. Moré suggests implementing the algorithm using a step bound in a diagonally-scaled Euclidean norm chosen so that the solution to (4.4.2) is invariant with respect to positive diagonal scalings of x. See [46].

For simplicity, let us use the Euclidean norm for the step bound. Then a straightforward application of Lagrange multipliers says that the solution to (4.4.2) is

$$(\mu I + H_c)s(\mu) = -g_c, \tag{4.4.3}$$

where $\mu=0$ if H_c is positive definite and $\|s^{qN}\| \leq \lambda$, or else $\mu>0$, for which $\|s(\mu)\|=\lambda$, and $\mu I + H_c$ is positive semidefinite. It is not necessary to solve (4.4.2) very accurately. Notice that singularity or indefiniteness of the model Hessian is automatically handled by this framework. It makes the implementation more complex to worry about these cases, but it is conceptually simple and the user can regard it as implementational detail. See [30], [51]. An interesting point is that as μ increases, the step is a shorter and shorter step more and more nearly along the negative gradient direction.

For the Gauss-Newton method, $H_c=H_c^{GN}$, we see that the Icos step choice algorithm is just the Levenberg-Marquardt algorithm. We have indicated ways to select λ, and so (4.4.3) gives the optimizer's point of view on how to choose the Levenberg- Marquardt parameter, a question users of older Levenberg-Marquardt codes often ask. We haven't gone into the extremely clever and effective iterative techniques used to find μ given λ. This material can be found in [16] and [46]. The user can regard it as detail, since the goal is clear. Minpack [47] contains an implementation of the Levenberg-Marquardt algorithm and [14] is an implementation of the augmented Gauss-Newton method based on these ideas. The general form of the idea seems to be from [36], and [31] contains an implementation for Newton's method.

It is clear from (4.4.3) that lcos needs $O(p^3)$ operations for dense problems, or in general the solution of at least one linear system. This is competitive for all except the variable-metric methods which need only $O(p^2)$ operations for a linesearch implementation. However, early in the computation, it is sometimes necessary to do inner iterations on the choice of μ, and each one costs a $p \times p$ linear system. This is a disadvantage also for large sparse problems. The dogleg step choice algorithms require no more algebra than the line search methods, and they have some of the flavor of the lcos method in that the direction of the step depends on λ. Powell [55] originally suggested the idea in connection with the Broyden update method, a variable-metric type algorithm for solving $p \times p$ systems of nonlinear equations, and Steihaug put it in the proper context of the conjugate-gradient algorithms in [72]. Powell's version was to approximate $x_c + s(\lambda)$ by a straight-line segment from x_c to the minimizer of $q_c(x_c - tg_c)$ and then on to $x_c + s^{qN}$. The dogleg point is just the point on this set of distance λ from x_c. Minpack [47] contains a Broyden method with a dogleg step choice for square nonlinear systems. Experience indicates that the dogleg often does better than line searches far from the solution, they are the same near the solution, but in the middle ground, there is some indication that an almost full step along s^{qN} is more effective than the same length dogleg step. Dennis and Mei [15] suggested a way to use this experience. Gay [31] has the best implementation of their double dogleg scheme using the BFGS variable-metric update.

For H_c positive definite, one way to view the dogleg is that it is exactly the segmented conjugate-gradient iteration applied to minimizing the quadratic model on the two dimensional subspace spanned by $x_c - g_c$ and $x_c + s^{qN}$. This view leads to a natural extension suggested in [72] that should be especially useful for large sparse problems. We simply begin applying conjugate gradients to solve (1.3) and treat this p-dimensional polygonal arc as the full dimensional dogleg. The theory for the lcos and dogleg techniques is very strong for Newton's method, but much remains to be learned. See [51], [56], [57], [61], [67], [70].

5. Constraints.

In the proceeding, we used the nonlinear least-squares problem as an example because its interesting structure was useful in making the fundamental concept of a local model more clear. The interesting structure derives from the fact that the objective function our algorithms try to minimize is a composition of a vector-valued residual function of a parameter vector with a real function of the residuals. We try, but don't expect to be able, to choose a parameter vector to make the residuals zero, and so as an explicit part of the problem statement, we have a rule for deciding which of a set of parameters vectors is the most meritorious. This least-squares rule often comes from statistical reasoning about the process being modeled and the data collection method, but not from the residual equations themselves. It is ironic that a much more difficult problem is the apparent subproblem of finding a root of a square $p \times p$ nonlinear system of equations. The difficulty is that since we expect to be able to make the residuals zero, there is no prior agreement between the algorist and the user as to what makes a proposed next iterate a better approximate solution than the current one. The criterion function that makes this decision is often called a merit function. We will see below that the choice of merit function is one of the unresolved issues in numerical methods for constrained optimization.

Nonlinear equations and nonlinear least squares are especially interesting to compare, because the Gauss-Newton method for (1.1) and the Newton method for square systems both use exactly the same local model for the residual function. The difference is that in one case, we plug that residual model into the merit function provided as part of the definition of the problem and that serves as our local model of the problem. We just try to reduce the merit function in making the next step; if we can't do that by going to the global solution to the local model problem as our next iterate, then we backtrack with a clear idea of what we are trying to do. The Newton method for the square problem is not nearly so well posed. It is straightforward to want to solve the model problem, $J_c s + R(x_c) = 0$, but what do we do if this problem has no solution, or if the solution gives a point that we are certain must be a worse approximation? The answer is that we use the least-squares merit

function to define our backtracking strategy. In other words, we make up an equivalent minimization problem and use the structure of the original problem to help build and solve the local model problems. Later, we will see a class of merit functions made up for the constrained problem. The unconstrained minimization problem can be viewed as the problem in which we are only given the merit function and no other structure; we build a local model of the merit function and apply the step choice strategy of our choice to the model.

For the sake of notation, we pause to define the constrained problem as:

$$\begin{aligned} &\min f(x), \\ &\text{for } g_i(x)=0,\ i=1,2,..,n \\ &\text{and } h_i(x)\leqq 0,\ i=1,...,m, \end{aligned} \tag{5.1}$$

where $g_i, h_i: \mathbf{R}^p \rightarrow \mathbf{R}$ denote the constraints. A point is called *feasible* if it satisfies all n+m constraints.

Returning to the discussion above, the key point that makes constrained minimization more difficult computationally is the absence of a natural way to choose between two infeasible approximate solutions. This means that, since we don't know how to measure improvement, it is unclear how to improve a poor initial estimate that does not satisfy the constraints. If we have two feasible points, x_1 and x_2, then it is easy to choose between them by comparing $f(x_1)$ to $f(x_2)$. Thus, some algorithms always generate iterates $\{x_k\}$ that satisfy the constraints. In practice, this strategy has the added advantage that the user is always given a feasible point even if the algorithm has failed to find a local optimum. They also have an advantage for those problems in which some constraints may have to be satisfied in order for the objective function to even be defined. Of course, the user can't be expected to provide a feasible initial point, and so a two-phase computation is necessary. Feasible-point algorithms are reasonable for linear constraints, but there are definite problems in using them for nonlinear constraints because the part of the iteration that deals with decreasing f will usually cause the raw iterate to become infeasible. Thus, feasibility would have to be restored as a second part of each iteration.

The likelihood of losing feasibility can be seen by considering the popular sequential quadratic programming approach to providing a local model of (5.1):

$$\begin{aligned} &\min\ q_c(x_c+s), \\ &\text{for } g_i(x)+g_i'(x)s=0,\ i=1,2,..,n, \\ &\text{and } h_i(x)+h_i'(x)s\leqq 0,\ i=1,...,m. \end{aligned} \tag{5.2}$$

In other words, we model the general nonlinear programming problem by a quadratic programming problem. Let s^{qN} denote the solution in terms of s to (5.2). Posed in this way, we will satisfy the linearized constraints at x_c+s^{qN}, but if H_c still just models the Hessian of f as it did for the unconstrained problem, then there would be no provision for following any curvature in the constraints. This can be viewed as a reason that usually the algorithms take

$$H_c=f''(x_c)+\sum_{i=1}^{n}\gamma_c^i g_i''(x_c)+\sum_{i=1}^{m}\eta_c^i h_i''(x_c), \tag{5.3}$$

where the vectors $\gamma_c\in\mathbf{R}^n$ and $\eta_c\in\mathbf{R}^m$ are the current estimates of the Lagrange multipliers, introduced below.

Just as $\nabla f(x_*)=0$ was very useful for the unconstrained problem, the Kuhn-Tucker necessary conditions for x_* to be a solution to (5.1) are useful in the constrained problem. See [32], [43]. In fact, geometrically these conditions have a similar flavor. The necessary condition for an unconstrained minimizer x_* is that there be no descent direction from x_*. The constrained problem has a more relaxed necessary condition; it says that there should be no descent direction from x_* that would not force violation of a constraint active at the putative solution x_*. An active constraint C_i at a point x is one for which $C_i(x)=0$, even if the constraint is $C_i(x)\leqq 0$. Thus all the g_i's are active at every feasible x, and those h_i are active for which $h_i(x)=0$. It will be handy to denote the vector function of constraints active at x_* by $A_*(x)$, and for definiteness, we will assume that there are n_* constraints active at x_*. Of course, we don't know in advance which constraints h_i will be in A_*, but it is standard practice to try to find out as the iteration proceeds. A particular heuristic scheme for doing this is called an "active set strategy". See [26], [32].

It would lead us to unnecessary complication to state the general Kuhn-Tucker necessary conditions for (5.1). The algorithms most often use an active set strategy to choose a set A_c of n_c constraints, and then try to satisfy the Kuhn-Tucker conditions for the corresponding equality constrained version of (5.1), min f(x) subject to $A_c(x)=0$. In this context, the necessary conditions for a point x to be a local solution to (5.1) with A_c contained in its active set are that under appropriate differentiability assumptions and constraint qualifications, there is a vector of Lagrange multipliers $\lambda \in \mathbf{R}^{n_c}$ such that

$$\begin{aligned} &\nabla f(x)+A_c'(x)^T\lambda=0 \\ &A_c(x)=0. \end{aligned} \tag{5.4}$$

Taken together, (5.4) is a square system of at most $p+n_c$ equations in the unknowns x,λ. One approach would be to treat (5.4) as an unstructured system of nonlinear equations whose solution should be near x_*,λ_*, the solution to (5.4) with A_* in place of A_c. Notice that since we are trying to solve (5.1), it is not enough to satisfy (5.4), a point also must be feasible for (5.1). It is less obvious, but not hard to see, that any component of λ_* should be positive if it corresponds to an active inequality constraint, i.e., if it is an η_*^i. It is also reasonable that at a local solution x_*, if $h_i(x_*)<0$, then $\eta_*^i=0$. Of course, a constraint might be inactive at one local solution and active at another.

A more rigorous derivation of the local quadratic program as a model leads to the local convergence results for the SQP technique given independently in [33], [40], and [74]. If we linearize the nonlinear system (5.4) about x_c,λ_c and set up the linear system for s and σ, the step in λ, then we get

$$\begin{aligned} &\Big[f''(x_c)+\sum_{i=1}^{n_c}\lambda_c^i(A_c)_i''(x_c)\Big]s+A_c'(x_c)^T\sigma=-\nabla f(x_c)-A_c'(x_c)^T\lambda_c \\ &A_c'(x_c)s=-A_c(x_c). \end{aligned} \tag{5.5}$$

This can be shown equivalent to solving a model quadratic program where the objective function is a straightforward Taylor expansion in x of the Lagrangian function,

$$L(x,\lambda_c)=f(x)+\lambda_c^TA_c(x).$$

This might be regarded as somewhat unnatural because it gives the multipliers the same

status as x. We could eliminate σ from (5.5) by substituting $\lambda_+ - \lambda_c$ and then eliminating the common term $A_c'(x_c)^T\lambda_c$ from both sides of the first block of equations. Notice that it is not necessary to distinguish between the Lagrange function with the current set A_c and with the whole set of constraints since our current estimate is 0 of any η_i whose h_i is not in the current active set. Thus, (5.5) is equivalent to (5.2) with H_c given by (5.3); in other words, the objective function in the model quadratic program has the gradient of the original objective function f and the Hessian with respect to x of the Lagrangian function. The new Lagrange mutipliers λ_+ can be interpreted from the local model problem. Implementation details of some importance have to do with the choice of active constraints. For example, do we apply an active set strategy to the building of the model, or do we linearize all the constraints and then apply an active set strategy to the model problem? Other concerns have to do with how to adapt from unconstrained minimization and nonlinear equations approximation schemes for all the derivative matrices.

As we mentioned, SQP has been shown to be locally convergent and recent work [27], [28] has the local theory approaching the same advanced state as for the unconstrained problem. However, the main problem still remains: how to use the solution to the model problem to improve a poor initial estimate. Han [40] suggested that poor initial guesses be improved by a line search along s^{qN} using the merit function

$$\vartheta(x) = f(x) + r[\sum_{i=1}^{n} g_i(x)_+ + \sum_{i=1}^{m} h_i(x)_+], \qquad (5.6)$$
$$\textit{where } y_+ \equiv \max\{0,y\} \text{ for } \textit{any real } y.$$

Here, r must be a positive number larger than the Lagrange multipliers at the solution. He proved that s^{qN} defines a descent direction for ϑ from x_c. See also [59]. Maratos [45] later showed that x_c could be arbitrarily close to x_*, a local solution to (5.1), but $x_c + s^{qN}$ might not reduce ϑ. Other authors [5], [19] have given a merit function that depends on two parameters for which there are values that admit the full step to the model problem solution. The choice of merit function is an important implementation decision that can be thought of as assigning weights to the competing objectives of getting more feasible and

more minimal.

The merit function ϑ is one of a class of merit functions applied to (5.1) which are called *penalty functions*. All of them involve a penalty parameter like r. If the penalty function is *exact* in the sense that there is an r sufficiently large for which x_*, a local solution to (5.1), is a local minimizer of the penalty function, then they have the advantage of allowing (5.1) to be turned into an unconstrained problem. Penalty functions without this property generally have the property that the minimizers $\{x(r)\}$ of the penalty function with parameter r, converge to a local solution to (5.1) with any sequence $\{r^k\}$ that converges to ∞. The problem is that the Hessians of the unconstrained problems become infinitely badly conditioned at the same time. Still, penalty functions have been useful tools, and SUMT, which is based on this approach, has been helpful to many users. See [21].

In the augmented Lagrangian technique, a penalty term is added to the Lagrangian function to help attain feasibility and to convexify the function in x. This again contributes to bringing the curvature of the constraints into the model problem, but it adds the complication of another parameter to estimate. Moreover, it is not clear whether this additional parameter can be used to improve the algorithm. See Tapia [75].

The reduced gradient is another useful approach for which reliable programs are available. This can be thought of as an attempt to change the variables so that the constraints are eliminated to give an unconstrained problem in fewer variables. The implementation of this idea is much more clear for linear constraints than for the general problem where a two-phase iteration is usually required to restore feasibility at each step. An excellent exposition of this complicated area is given in [32].

6. Concluding Remarks.

There has been dramatic progress in all of computational mathematics during the past twenty years. Numerical nonlinear optimization certainly lagged behind numerical linear algebra, approximation theory, and quadrature, but we are catching up fast. We now

have a number of excellent young researchers in the area and more and more of the research effort is going into ways to incorporate constraints into the conceptual framework developed in the proceeding sections. This is not an indication that research in unconstrained minimization is over; rather, it is more of an indication that the problems there have gotten sufficiently hard that it is more immediately productive to apply some of what has been learned there to improve the algorithms available for constrained problems.

We have seen ample evidence that there are many thoughtful algorithms available for unconstrained optimization. In fact, MINPACK [47], NL2SOL [14], the routines mentioned by Gay [31], and any routines that appear in TOMS, the ACM Transactions on Mathematical Software, are available for under $100 through one of the IMSL distribution services. The Harwell library is a bargain, especially at current exchange rates, but it is only partially portable from the IBM 370. NAg has always been the commercial leader in optimization routines, but IMSL has recently made a commitment, and has already begun, to improve this part of its library. Neither is especially expensive for commercial customers or university computer centers, although university research groups will probably find them less economically feasible as we get our own research machines and need only specialized portions of the libraries. One suspects however, that NAg and IMSL will find ways to accommodate this potential market.

NAg has always had special routines available for the case in which the only constraints are simple bounds on the variables. In this author's experience, a majority of the constraints encountered in practice are of this type; for example, certain of the variables might be constrained to be nonnegative. In general, it is more efficient to use a program that handles no constraints more complex than those in the problem to be solved. Of course, this does require familiarity with a larger library.

Recently, there have been several good surveys of nonlinear programming software. See [65], [77], [81]. The proceedings of the NATO conference held at Cambridge in 1981 [60] contains surveys on all of the topics mentioned here. With this paper as an introduc-

tion, the reader will probably be able to follow all the contributions without difficulty; the section on "The Current State of Software" will be of special interest. The books [16], [25], [26], [32] should also be accessible.

We have not talked about optimization over function spaces. The reason for the omission is that the reduction of such problems to the type studied here is application dependent and is not a subarea of numerical optimization. In other words, the author does not know as much about that as the reader, and he elects not to use his limited number of pages displaying a proof of this fact.

Finally, we have only made passing remarks about large sparse problems. There was a strong effort in that direction until about 1980. We will certainly return to it later because progress continues on solving sparse linear problems, and numerical linear algebra is the mother's milk of numerical optimization. Perhaps the major sticking point now is the superficial nature of sparsity as a property of the problem. In work on small dense problems, we discovered that it is a good thing to have our algorithms be independent to the greatest possible extent of the basis used to represent the problem. Thus, our best variable-metric methods are based on fundamental properties of the Hessian like symmetry and positive definiteness that are linearly invariant. Sparsity is a property only of the pair consisting of the problem and the basis used to represent it. This means that we lose some powerful derivative approximation methods and so Newton's method is the only quadratic modeling technique that carries over from the small dense case. See [66], [76]. There has been some important work done in savings for finite-difference approximations to the Hessian. See [9], [62].

The lcos approach to the problem of improving a poor initial guess for a large sparse problem is not really practical because of the expense of solving $p \times p$ systems. The dogleg approach based on its derivation from the conjugate gradient algorithm is the only alternative to line searches at this time. See [72]. The inexact-Newton approach [11] helps to understand the issue of how accurately to solve the system (1.3) in order not to inhibit the

convergence rate of the underlying quadratic modeling scheme. The work of [51], and [67] will be useful in deciding how accurately to solve the model problem.

Sometimes, the problem is so large that it is impossible to store Hessian information. In this case, it is possible to apply a conjugate-direction [69] algorithm to an underlying quadratic model that is defined by low dimensional snapshots taken as needed. For example, if the algorithm requires $H_c s$, then this is treated as an approximate directional derivative and approximated by finite differences on the gradient. These algorithms are very interesting from a theoretical point of view, since it is not clear how to analyze them. More work is needed in this area because the algorithms are so convenient for large problems. See [29], [53].

The author would like to thank Trond Steihaug and Richard Tapia for commenting on the manuscript.

References.

(1) E.Allgower and K.Georg (1980), "Simplicial and continuation methods for approximating fixed points and solutions to systems of equations", *SIAM Review 22*, pp.28-85.

(2) L.Armijo (1966), "Minimization of functions having Lipschitz-continuous first partial derivatives", *Pacific J. Math. 16*, pp.1-3.

(3) M.Avriel (1976), *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

(4) P.T.Boggs and J.E.Dennis Jr. (1976), "A stability analysis for perturbed nonlinear iterative methods", *Math. Comp. 30*, pp.1-17.

(5) P.T.Boggs and J.W.Tolle (1980), "Augmented Lagrangians which are quadratic in the multiplier", *J.Opt.Theory Appl. 8*, pp.17-26.

(6) C.G.Broyden (1965), "A class of methods for solving nonlinear simultaneous equations", *Math. Comp. 19*, pp.577-593.

(7) C.G.Broyden (1969), "A new double-rank minimization algorithm", *AMS Notices 16*, p.670.

(8) C.G.Broyden (1970), "The convergence of a class of double-rank minimization algorithms", Parts I & II, *J.I.M.A. 6*, pp. 76-90 & 222-236.

(9) T.F.Coleman and J.J.Moré (1982), "Estimation of sparse Hessian matrices and graph coloring problems", Cornell U. Computer Science TR 82-535.

(10) W.C.Davidon (1959), "Variable metric methods for minimization", Argonne National Labs Report ANL-5990.

(11) R. S. Dembo, S. C. Eisenstat, and T. Steihaug (1982), Inexact Newton methods, *SIAM J. of Numer. Anal.*, 19, pp.400-408.

(12) J.E.Dennis Jr. (1977), "Nonlinear least squares and equations", in *The State of the Art in Numerical Analysis*, D.Jacobs, Ed., Academic Press, London, pp.269-312.

(13) J.E.Dennis Jr., D.M.Gay, and R.E.Welsch (1981a), "An adaptive nonlinear least-squares algorithm", *TOMS 7*, pp.348-368.

(14) J.E.Dennis Jr., D.M.Gay, and R.E.Welsch (1981b), "Algorithm 573 NL2SOL - An adaptive nonlinear least-squares algorithm [E4]", *TOMS 7*, pp.369-383.

(15) J.E.Dennis Jr. and H.H.W.Mei (1979), "Two new unconstrained optimization algorithms which use function and gradient values", *J. Optim. Theory Appl. 28*, pp.453-482.

(16) J.E.Dennis Jr. and R.B.Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs,NJ.

(17) J.E. Dennis, Jr. and Homer F. Walker (1983a), "Inaccuracy in quasi-Newton methods: local improvement theorems", (Department of Mathematical Sciences, TR83-11, Rice University). To appear in *Math. Prog. Studies*.

(18) J.E. Dennis, Jr. and Homer F. Walker (1983b), "Least-change secant update methods with inaccurate secant conditions", Department of Mathematical Sciences, TR83-26B,

Rice University.

(19) G.DiPillo and L.Grippo (1979), "A new class of augmented Lagrangians in nonlinear programming", *SIAM J. Control Optim. 17*, pp.618-628.

(20) L.C.W.Dixon and G.P.Szego (1975,1978), *Towards Global Optimization, Vols 1,2*, North-Holland, Amsterdam

(21) A.V.Fiacco and G.P.McCormick (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, NY.

(22) R.Fletcher and M.J.D.Powell (1963), "A rapidly convergent descent method for minimization", *Comput. J. 6*, pp.163-168.

(23) R.Fletcher (1970), "A new approach to variable metric algorithms", *Comput. J. 13*, pp.317-322.

(24) R.Fletcher (1971), "A modified Marquardt subroutine for non-linear least squares", A.E.R.E. Harwell report TP 476.

(25) R.Fletcher (1980), *Practical Methods of Optimization, Vol 1, Unconstrained Optimization*, John Wiley and Sons, New York.

(26) R.Fletcher (1981), *Practical Methods of Optimization, Vol 2, Constrained Optimization*, John Wiley and Sons, New York.

(27) R.Fonticilla (1983), "A general convergence theory for a class of quasi-Newton methods for constrained optimization", Ph.D. Thesis, Department of Mathematical Sciences, Rice University.

(28) R.Fonticilla, T.Steihaug, R.A.Tapia (1983), "A convergence theory for a class of quasi-Newton methods for constrained optimization", Department of Mathematical Sciences, TR83-15, Rice University.

(29) N.K.Garg and R.A.Tapia (1980), "QDN: A variable storage algorithm for unconstrained optimization", Department of Mathematical Sciences, Rice University.

(30) D.M.Gay (1981), "Computing optimal locally constrained steps", *SIAM J. Sci. Stat. Comp. 2*, pp.186-197.

(31) D.M.Gay (1983), "Subroutines for unconstrained minimization using a model/trust-region approach", *TOMS* (to appear).

(32) P.E.Gill, W.Murray and M.H.Wright (1981), *Practical Optimization*, Academic Press, London.

(33) T.Glad (1979), "Properties of updating methods for the multipliers in augmented Lagrangians", *J. Optim. Theory Applics. 28*, pp.135-156.

(34) T.Glad and A.Goldstein (1977), "Optimization of functions whose values are subject to small errors", *B.I.T.,17* pp.160-169.

(35) D.Goldfarb (1970), "A family of variable metric methods derived by variational means", *Math. Comp. 24*, pp.23-26.

(36) S.M.Goldfeldt, R.E.Quandt, and H.F.Trotter (1966), "Maximization by quadratic hill-climbing", *Econometrica 34*, pp.541-551.

(37) A.A.Goldstein (1967), *Constructive Real Analysis*, Harper & Row, New York.

(38) G.Golub and C.Van Loan (1983), *Matrix Computations*, The Johns Hopkins University Press.

(39) S.Gomez and A.V.Levy (1982), "The tunneling method for solving the constrained global optimization problem with several non-connected feasible regions", *Numerical Analysis*, Proceedings, Cocoyoc, Mexico 1981, edited by J.P.Hennart, Springer Lecture Notes in Mathematics 909, pp34-47.

(40) S.P.Han (1977), "A globally convergent method for nonlinear programming", *J. Opt. Theory Appli. 22*, pp.297-309.

(41) K.Levenberg (1944), "A method for the solution of certain problems in least squares", *Quart. Appl. Math. 2*, pp. 164-168.

(42) A.V.Levy, A.Montalvo, S.Gomez, and A.Calderon (1982), "Topics in global optimization", *Numerical Analysis*, Proceedings, Cocoyoc, Mexico 1981, edited by J.P.Hennart, Springer Lecture Notes in Mathematics 909, pp18-33.

(43) O.L.Mangasarian (1969), *Nonlinear Programming*, McGraw-Hill, NY.

(44) D.Marquardt (1963), "An algorithm for least-squares estimation of nonlinear parameters", *SIAM J. Appl. Math. 11*, pp.431-441.

(45) N.Maratos (1978), "Exact penalty function algorithms for finite-dimensional and control optimization problems", Ph.D. Thesis, University of London.

(46) J.J.Moré (1977), "The Levenberg-Marquardt algorithm: implementation and theory", in *Numerical Analysis*, G.A.Watson, Ed., Lecture Notes in Math. 630, Springer Verlag, Berlin, pp.105-116.

(47) J.J.Moré, B.S.Garbow, and K.E.Hillstrom (1980), "User guide for MINPACK-1", Argonne National Labs Report ANL-80-74.

(48) J.J.Moré, B.S.Garbow, and K.E.Hillstrom (1981a), "Testing unconstrained optimization software", *TOMS 7*, pp.17-41.

(49) J.J.Moré, B.S.Garbow, and K.E.Hillstrom (1981b), "Fortran subroutines for testing unconstrained optimization software", *TOMS 7*, pp.136-140.

(50) J.J.Moré and D.C.Sorensen (1979), "On the use of directions of negative curvature in a modified Newton method", *Math. Prog. 16*, pp.1-20.

(51) J.J.Moré and D.C.Sorensen (1982), "Newton's method", Argonne Nat. Labs. Mathematical and Computer Sciences ANL-82-8.

(52) J.A.Nelder and R.Mead (1965), "A simplex method for function minimization", *Comput. J. 7*, pp.308-313.

(53) D.P.O'Leary (1982), "A discrete Newton algorithm for minimizing a function of many variables", *Math. Prog. 23*, pp.20-33.

(54) J. M. Ortega and W. C. Rheinboldt (1970), *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York.

(55) M.J.D.Powell (1970a), "A hybrid method for nonlinear equations", in *Numerical Methods for Nonlinear Algebraic Equations,* P.Rabinowitz, Ed., Gordon and Breach, London, pp.87-114.

(56) M.J.D.Powell (1970b), "A new algorithm for unconstrained optimization", in *Nonlinear Programming,* J.B.Rosen, O.L.Mangasarian and K.Ritter, Eds., Academic Press, New York, pp.31-65.

(57) M.J.D.Powell (1975), "Convergence properties of a class of minimization algorithms", in *Nonlinear Programming 2,* O.Mangasarian, R.Meyer, and S.Robinson, Eds., Academic Press, NY, pp.1-27.

(58) M.J.D.Powell (1976), "Some global convergence properties of a variable metric algorithm without exact line searches", in *Nonlinear Programming,* R.Cottle and C.Lemke, Eds., AMS, Providence,R.I., pp.53-72.

(59) M.J.D.Powell (1976), "Algorithms for nonlinear constraints that use Lagrangian functions", presented at the Ninth Int. Symp. on Math. Prog., Budapest, available as DAMTP report, DAMTP, University of Cambridge, Silver Street, CB3 9EW, England.

(60) M.J.D.Powell (1982a), *Nonlinear Optimization 1981,* NATO Conference Series, Academic Press, London.

(61) M.J.D.Powell (1982b), "On the global convergence of trust region algorithms for unconstrained minimization", available as DAMTP report, DAMTP, University of Cambridge, Silver Street, CB3 9EW, England.

(62) M.J.D.Powell and Ph.L.Toint (1979), "On the estimation of sparse Hessian matrices", *SIAM J. Numer. Anal. 16,* pp.1060-1074.

(63) L.B.Rall (1981), *Automatic Differentiation: Techniques and Applications,* Springer Lecture Notes in Computer Science 120.

(64) J.K.Reid (1973), "Least squares solution of sparse systems of non-linear equations by a modified Marquardt algorithm", in *Proc. NATO Conf. at Cambridge, July 1972*, North Holland, Amsterdam,pp.437-445.

(65) K.Schittkowski (1980), *Nonlinear Programming Codes*, Springer Lecture Notes in Economics and Mathematical Systems 183.

(66) R.B.Schnabel and Ph.L.Toint (1983), "Forcing sparsity by projecting with respect to a non-diagonally weighted Frobenius norm", *Math. Prog. 25*, pp.125-129.

(67) G.A.Shultz, R.B.Schnabel, and R.H.Byrd (1982), "A family of trust region based algorithms for unconstrained minimization with strong global convergence properties", Department of Computer Science tech. rpt. CU-CS-216-82.

(68) D.F.Shanno (1970), "Conditioning of quasi-Newton methods for function minimization", *Math. Comp. 24*, pp.647-657.

(69) D.F.Shanno (1978), "Conjugate-gradient methods with inexact searches", *Math. of Oper. Res. 3*, pp.244-256.

(70) D.C.Sorensen (1982), "Newton's method with a model trust region modification", *SIAM J. Numer. Anal. 19*, pp.409-426.

(71) T.Steihaug (1981), "Quasi-Newton methods for large scale nonlinear problems", Ph.D. Thesis Yale University.

(72) T.Steihaug (1981), "The conjugate gradient method and trust regions in large scale optimization", Department of Mathematical Sciences, TR81-1, Rice University.

(73) J.Stoer and R.Bulirsch (1980), *Introduction To Numerical Analysis*, Springer, NY.

(74) R.A.Tapia (1977), "Diagonalized multiplier methods and quasi-Newton methods for constrained optimization", *J. Optim. Theory Appl. 22*, pp.135-194.

(75) R.A.Tapia (1980), "Augmented Lagrangian methods for constrained optimization: the role of the penalty constant", *Ottimizzazione Non Lineare e Applicazioni*, edited by S.Incerti and G.Treccani, *Quaderni dell'Unione Matematica Italiana, 17*, pp.161-186.

(76) Ph.L.Toint (1981), "A sparse quasi-Newton update derived variationally with a non-diagonally weighted Frobenius norm", *Math. Comp. 37*, pp.425-434.

(77) A.D.Waren and L.S.Lasdon (1979), "The status of nonlinear programming software", *Operations Research 27*, pp.431-456.

(78) L.T.Watson (1979), "An algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problems", *SIAM J. Numer. Anal. 16*, pp.394-401.

(79) L.T.Watson (1981), "Engineering applications of the Chow-Yorke algorithm" *Appl. Math. and Comp. 9*, pp.111-133.

(80) P.-A.Wedin (1974), "On the Gauss-Newton method for the non-linear least squares problem", ITM Arbetsrapport nr.24, Inst. för Tellämpad Matematik, Stockholm.

(81) M.H.Wright (1978), "A survey of available software for nonlinearly constrained optimization", Technical Report SOL 78-4, Dept. of Operations Research, Stanford University.

(82) P.Wolfe (1969), "Convergence conditions for ascent methods", *SIAM Review 11*, pp.226-235.

(83) P.Wolfe (1971), "Convergence conditions for ascent methods. II: some corrections", *SIAM Review 13*, pp.185-188.

(84) P.Young (1976), "Optimization in the presence of noise--a guided tour", *Optimization in Action*, L.C.W.Dixon,ed., Academic Press, London.

(85) T.J.Ypma (1983), "The effect of rounding errors on Newton-like methods", *Inst.Maths.Applics.J.Numer.Anal.* (to appear).

(86) F.Zirilli (1982), "The solution of nonlinear systems of equations by second order systems of o.d.e. and linearly implicit A-stable techniques", *SIAM J. Numer. Anal. 19*, pp.800-816.