Optimization and Nonlinear Equations

Gordon K. Smyth

May 1997

Optimization means to find that value of $\mathbf{x}$ which maximizes or minimizes a given function $f(\mathbf{x})$. The idea of optimization goes to the heart of statistical methodology, as it is involved in solving statistical problems based on least squares, maximum likelihood, posterior mode and so on.

A closely related problem is that of solving a nonlinear equation,

$$\mathbf{g}(\mathbf{x}) = 0$$

for $\mathbf{x}$ where $\mathbf{g}$ is a possibly multivariate function. Many algorithms for minimizing $f(\mathbf{x})$ are in fact derived from algorithms for solving $\mathbf{g} = \partial f/\partial \mathbf{x} = 0$, where $\partial f/\partial \mathbf{x}$ is the vector of derivatives of f with respect to the components of $\mathbf{x}$.

Except in linear cases, optimization and equation solving invariably proceed by iteration. Starting from an approximate trial solution, a useful algorithm will gradually refine the working estimate until a pre-determined level of precision has been reached. If the functions are smooth, a good algorithm can be expected to converge to a solution when given a sufficiently good starting value.

A good starting value is one of the keys to success. In general, finding a starting value requires heuristics and an analysis of the problem. One strategy for fitting complex statistical models, by maximum likelihood or otherwise, is to progress from the simple to the complex in stages. Fit a series of models of increasing complexity, using the simpler model as a starting value for the more complicated model in each case. Maximum likelihood iterations can often be initialized by using a less efficient moment estimator. In some special cases, such as generalized linear models, it is possible to use the data itself as a starting value for the fitted values.

An extremum (maximum or minimum) of f can be either *global* (truly the extreme value of f over its range) or *local* (the extreme value of f in a neighborhood containing the value). (See Figure 1.) Generally it is the global extremum that we want. (A maximum likelihood estimator, for example, is by definition the global maximum of the likelihood.) Unfortunately, distinguishing local extrema from the global extremum is not an easy task. One heuristic is to start the iteration from several widely varying starting points, and to take the most extreme (if they are not equal). If necessary a large number of starting values can be randomly generated. Another heuristic is to perturb a local extremum slightly to check that the algorithm returns to it. A relatively recent algorithm, simulated annealing, is often used successfully on problems where there are a large number of closely competing local extrema.

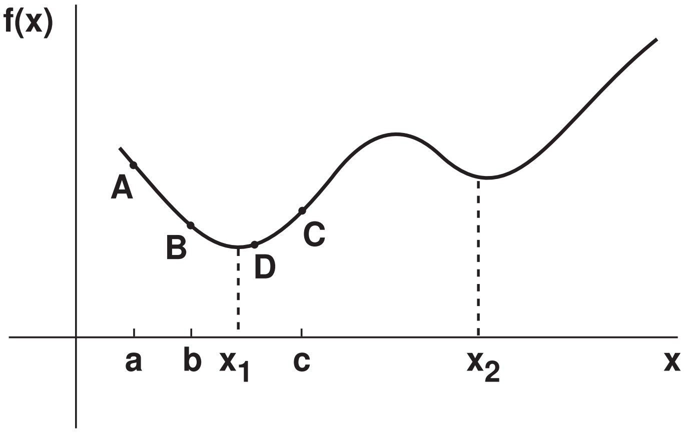

Figure 1: The function $f(x)$ has a local minimum at x_2 and a global minimum at x_1. The points $A = (a, f(a))$, $B = (b, f(b))$ and $C = (c, f(c))$ bracket the global minimum. The next point tried by a golden section search would be D.

This article discusses *unconstrained optimization*. Sometimes however $\mathbf{x}$ must satisfy one or more constraints. An example is some of the components of $\mathbf{x}$ being known a priori to be positive. In some cases the constraints may be removed by a suitable transformation ($x_i = e^{z_i}$ for example), or by use of Lagrange multipliers.

One must choose between algorithms which use derivatives and those which do not. In general, methods which use derivatives are more powerful. However, the increase in speed does not always outweigh the extra overheads in computing the derivatives, and it can be a great convenience for the user to not have to program them.

Algorithms are also distinguished by the amount of memory they consume. Storage requirements are typically order N or order N^2 where N is the dimension of $\mathbf{x}$. In most biostatistical applications, N is not usually so large that storage becomes an issue.

If you can calculate first and second derivatives of f, then the well-known *Newton's method* is simple and works well. It is crucially important though to check the function value $f(\mathbf{x})$ at each iteration, and to implement some sort of backtracking strategy, to prevent the Newton iteration from diverging to distant parts of the parameter space from a poor starting value. If second derivatives are not available, then *quasi-Newton* methods, of which Fisher's method of scoring is one, can be recommended. General purpose quasi-Newton algorithms build up a working approximation to the second derivative matrix from successive values of the first derivative. If computer memory is very critical, then *conjugate gradient* methods make the same assumptions as quasi-Newton methods but require only order N storage [8, Section 10.6]. If even first derivatives are not available, the Nelder-Mead *downhill simplex* algorithm is compact and reasonably robust. However the slightly more complex *direction-set* methods or Newton methods with finite difference approximations to the derivatives should minimize most functions with fewer function evaluations. Whilst all the above comments apply generally, the one-dimensional problem is something of a special case. In one dimension, once one can provide an interval which contains the solution, there exist efficient "low-tech" algorithms robust enough to take on all problems.

A practical introduction to root finding and optimization is given in Chapters 9, 10 and 15 (Sections 15.5 and 15.7) of Numerical Recipes [8]. More specialist texts are Dennis and Schnabel [2], Fletcher [3] and Gill et al [4]. A classic but technical text on solving nonlinear equations is Ortega and Rheinboldt [6]. A survey of available software is given by Moré and Wright [5].

One Dimension

The case where x is one-dimensional is not just a special case, it is qualitatively simpler than the multidimensional case. This is because a solution can be trapped between bracketing values, which are gradually brought together. A root of $g(x)$ is bracketed in the interval (a, b) if $f(a)$ and $f(b)$ have opposite signs. A minimum of $f(x)$ is bracketed by a triplet of values, $a < b < c$, if $f(b)$ is less than both $f(a)$ and $f(c)$.

The simplest and most robust method for finding a root in a bracketing interval is *bisection*. That is, we evaluate the function g at the midpoint of (a, b) and examine its sign. The midpoint then replaces whichever endpoint has the same sign. After k iterations, the root is known to lie in an interval of length $(b-a)/2^k$.

The equivalent method for function minimization is the *golden section search*. Given a bracketing triplet of points, the next point to be tried is that which is a fraction 0.38197 of the way from the middle point of the triplet to the farther end point (Figure 1). One then drops whichever of the endpoints is farthest from the new minimum. The strange choice of step size ensures that at each iteration the middle point is always the same fraction of the way from one endpoint to the other (the so-called golden ratio). After k iterations, the minimum is bracketed in an interval of length $(c-a)\cdot 0.61803^k$.

Bisection and golden section search are linear methods, in that the amount of work required increases linearly with the number of significant figures required for x. There are a number of other methods, such as the *secant method*, the *method of false position*, *Muller's method* and *Ridder's method*, which are capable of *superlinear convergence*, where-in the number of significant figures liberated by a given amount of computation increases as the algorithm converges. The basic idea is that g should be roughly linear in the vicinity of a root. These methods interpolate a line or a quadratic polynomial through two or three previous points, and use the root of the polynomial as the next iterate. They therefore converge more rapidly than bisection or golden search when the function g is smooth, but can converge slowly when g is not well approximated by a low order polynomial. They also require modification if they are not to risk throwing the iteration outside the bracketing interval known to contain the root.

It is an advantage to use one of the higher order interpolating methods when the function g is nearly linear, but to fall back on the bisection or golden search methods when necessary. In that way a rate of convergence at least equal to that of the bisection or golden section methods can be guaranteed, but higher order convergence can be enjoyed when it is possible. Brent [1] [8] has published methods which do the necessary bookkeeping to achieve this, and which can be generally recommended for root finding or minimizing in one dimension. Brent's algorithms do not require the derivatives of f or g to be supplied. However the method for minimizing a function can be easily modified to make use of the derivative when it is available [8].

Newton's Method

The most celebrated of all methods for solving a nonlinear equation is *Newton's method*, also called *Newton-Raphson*. Newton's method is based on the idea of approximating $\mathbf{g}$ with its linear Taylor series expansion about a working value $\mathbf{x}_k$. Let $G(\mathbf{x})$ be the matrix of partial derivatives of $\mathbf{g}(\mathbf{x})$ with respect to $\mathbf{x}$. Using the root of the linear expansion as the new approximation gives

$$\mathbf{x}_{k+1} = \mathbf{x}_k - G(\mathbf{x}_k)^{-1}\mathbf{g}(\mathbf{x}_k)$$

(Figure 2.)

The same algorithm arises for minimizing $f(\mathbf{x})$ by approximating f with its quadratic Taylor series expansion about $\mathbf{x}_k$. In the minimization case, $\mathbf{g}(\mathbf{x})$ is the derivative vector (gradient) of $f(\mathbf{x})$ with respect to $\mathbf{x}$ and the second derivative matrix $G(\mathbf{x})$ is symmetric. Beware though that Newton's method as it stands will converge to a maximum just as easily as to a minimum.

If f is a log-likelihood function, then $\mathbf{g}$ is the score vector and $-G$ is the observed information matrix. Newton's method for maximizing the likelihood is based on the same quadratic expansion which underlies asymptotic maximum likelihood theory.

Newton's method is powerful and simple to implement. It will converge to a fixed point from any sufficiently close

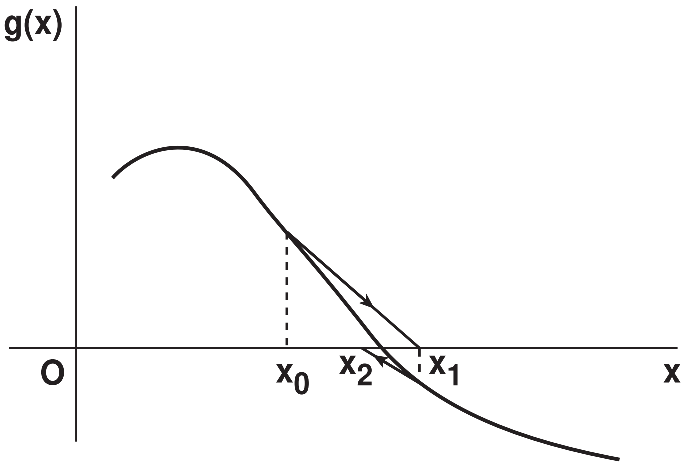

Figure 2: Newton's method converges quadratically from the starting value x_0.

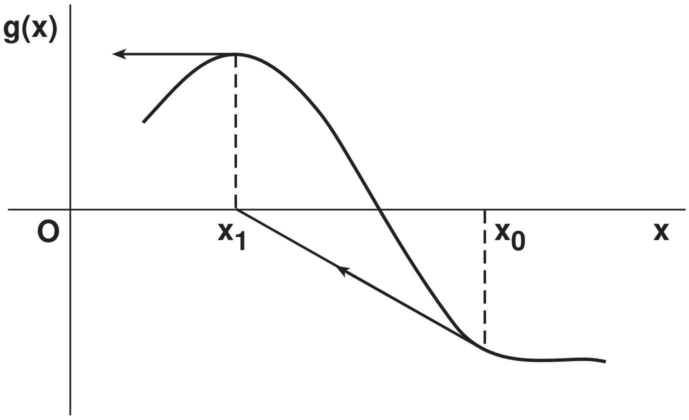

Figure 3: Newton's method diverges from the starting value x_0.

starting value. Moreover, once it starts to home in on a root, the convergence is quadratic. This means that if the error is ϵ, the error after one more iteration is of order ϵ^2. In other words, the number of significant places eventually doubles with each iteration. However its global convergence properties are poor. If $\mathbf{x}_k$ is unlucky enough to occur near a turning point of $\mathbf{g}$, the method can easily explode, sending the next estimate far out into the parameter space (Figure 3.) In fact, the set of values for which Newton's method does and doesn't converge can produce a fractal pattern [8].

The problems with Newton's method are (i) inability to distinguish maxima from minima, and (ii) poor global convergence properties. Both problems can be solved effectively through a *restricted step* sub-optimization [3]. Suppose we want to minimize $f(\mathbf{x})$. A condition for a minimum is that $G(\mathbf{x})$ be positive definite. We therefore add a diagonal matrix to G to ensure that it is positive definite:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [G(\mathbf{x}_k) + \lambda_k I]^{-1}\mathbf{g}(\mathbf{x}_k)$$

It is always possible to choose λ_k sufficiently large so that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$. A simple algorithm then is to choose λ_k just large enough to ensure a descent step. As the iteration converges to a minimum, λ_k can be decreased towards zero so that the algorithm enjoys superlinear convergence. This is the algorithm of choice when derivatives of f are available.

Solving $\mathbf{g}(\mathbf{x}) = 0$, when $\mathbf{g}$ is not the gradient of some objective function f, is slightly more difficult. One can manufacture a stand-in objective function by defining

$$f(\mathbf{x}) = \mathbf{g}(\mathbf{x})^T\mathbf{g}(\mathbf{x}).$$

Then the root of $\mathbf{g}$ occurs at a minimum of f. Note however that $\mathbf{g}$ is not the derivative of f, so that the above restricted step strategy is not available. In this case we can replace the Newton step with the *line search* strategy,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k G(\mathbf{x}_k)^{-1}\mathbf{g}(\mathbf{x}_k)$$

where $0 < \alpha_k < 1$. It is always possible to choose α_k sufficiently small that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$. The line search idea is to implement a one-dimensional suboptimization at each step, minimizing $f(\mathbf{x}_{k+1})$ approximately with respect to α_k.

Both the restricted step and the line search algorithms have global convergence properties. They can be guaranteed to find a local minimum of f and a root of $\mathbf{g}$ if such exist.

Quasi-Newton Methods

One of the drawbacks of Newton's method is that it requires the analytic derivative G at each iteration. This is a problem if the derivative is very expensive or difficult to compute. In such cases it may be convenient to iterate according to

$$\mathbf{x}_{k+1} = \mathbf{x}_k - A_k^{-1}\mathbf{g}(\mathbf{x}_k)$$

where A_k is an easily computed approximation to $G(\mathbf{x}_k)$. For example, in one dimension, the *secant method* approximates the derivative with the difference quotient

$$a_k = \frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}}$$

Such an iteration is called a *quasi-Newton* method. If A_k is positive definite, as it usually is, an alternative name is *variable metric* method.

One positive advantage to using an approximation in place of G is that A_k can be chosen to be positive definite, ensuring that the step will not be attracted to a maximum of f when one wants a minimum. Another advantage is that $A_k^{-1}\mathbf{g}(\mathbf{x}_k)$ is a descent direction from $\mathbf{x}_k$, allowing the use of line searches.

The best known quasi-Newton method in statistical contexts is Fisher's method of scoring, which is treated in more detail below. Among general purpose quasi-likelihood algorithms, the best is probably the *Broydon-Fletcher-Goldfarb-Shanno* (BFGS) algorithm. BFGS builds upon the earlier and similar *Davidon-Fletcher-Powell* algorithm. BFGS starts with a positive definite matrix approximation to $G(\mathbf{x}_0)$, usually the identity matrix. At each iteration it makes a minimalist (rank two) modification to A_k^{-1} to gradually approximate $G(\mathbf{x}_k)^{-1}$. Both DFP and BFGS are robust algorithms showing superlinear convergence.

Statisticians might fall into the trap of thinking that the final approximation A_k^{-1} is a good approximation to $G^{-1}(\mathbf{x}_k)$ at the final estimate. Since A_k is chosen to approximate $G(\mathbf{x}_k)$ only in the directions needed for the Newton step, it is useless for the purpose of providing standard errors for the final estimates.

Fisher's Method of Scoring

Of frequent interest to statisticians is the case where $f(\mathbf{x})$ is a log-likelihood function and $\mathbf{x}$ is the vector of unknown parameters. Then $\mathbf{g}$ is the score vector and $-G$ is the observed information matrix. For many models (curved exponential families are the major class) the Fisher information, $\mathcal{I}(\mathbf{x}) = E[-G(\mathbf{x})]$, is much simpler in form than $-G(\mathbf{x})$ itself. Furthermore, since $\mathcal{I}(\mathbf{x}) = \mathrm{var}[\mathbf{g}(\mathbf{x})]$, $\mathcal{I}(\mathbf{x})$ is positive definite for any parameter value $\mathbf{x}$ for which the statistical model is not degenerate. The quasi-Newton method which replaces $-G(\mathbf{x})$ with $\mathcal{I}(\mathbf{x})$ is known as *Fisher's method of scoring* [7, Section 5g]. Fisher scoring is linearly convergent, at a rate which depends on the relative difference between observed and expected information [10].

Consider the special case of nonlinear least squares, in which context Fisher scoring has a very long history and is known as the *Gauss-Newton* algorithm. The objective function is

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i - \mu(\mathbf{t}_i, \boldsymbol{\beta})]^2$$

where the y_i are observations and μ is a general function of covariate vectors $\mathbf{t}_i$ and the vector of unknown parameters $\boldsymbol{\beta}$. Write $\mathbf{y}$ for the vector of y_i, $\boldsymbol{\mu}$ for the vector of $\mu(\mathbf{t}_i, \boldsymbol{\beta})$ and $\dot{\boldsymbol{\mu}}$ for the derivative matrix of $\boldsymbol{\mu}$ with respect to $\boldsymbol{\beta}$. The Fisher scoring iteration becomes

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + (\dot{\boldsymbol{\mu}}^T \dot{\boldsymbol{\mu}})^{-1} \dot{\boldsymbol{\mu}}^T (\mathbf{y} - \boldsymbol{\mu})$$

where all terms on the right-hand size are evaluated at $\boldsymbol{\beta}_k$. The updated estimate is obtained by adding to $\boldsymbol{\beta}_k$ the coefficients from the multiple regression of the residuals $\mathbf{y} - \boldsymbol{\mu}$ on the derivative matrix $\dot{\boldsymbol{\mu}}$. Gauss-Newton therefore solves the nonlinear least squares problem by way of a series of linear regressions.

The Gauss-Newton algorithm can be speeded-up considerably in the special case that some of the $\boldsymbol{\beta}$ appear linearly in μ. For example if

$$\mu(t_i; \boldsymbol{\beta}) = \beta_1 e^{-\beta_3 t_i} + \beta_2 e^{-\beta_4 t_i}$$

then β_1 and β_2 are linear parameters. In such cases, the Gauss-Newton iteration can be restricted to the nonlinear parameters, β_3 and β_4. This idea is known as *separable* least squares. See for example Seber and Wild [9, Section 14.7]. Smyth [10] discusses the same principle in the context of maximum likelihood estimation.

Perhaps the most important application of Fisher scoring is to generalized linear models (GLMs). GLMs extend the idea of nonlinear regression to models with non-normal error distributions, including logistic regression and log-linear models as special cases. GLMs assume that y_i is distributed according to a probability density or mass function of the form

$$p(y; \theta_i, \sigma^2) = a(y, \sigma^2) \exp\{\frac{1}{\sigma^2}[y\theta_i - b(\theta_i)]\}$$

for some functions b and a (a curved exponential family). We find that $E(y_i) = \mu_i = b'(\theta_i)$ and $\mathrm{var}(y_i) = \sigma^2 v(\mu_i)$ where $v(\mu_i) = b''(\theta_i)$. If the mean μ_i of y_i is as given above for the nonlinear least squares, then the Fisher scoring iteration for $\boldsymbol{\beta}$ is a slight modification of the Gauss-Newton iteration:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + (\dot{\boldsymbol{\mu}}^T V^{-1} \dot{\boldsymbol{\mu}})^{-1} \dot{\boldsymbol{\mu}}^T V^{-1} (\mathbf{y} - \boldsymbol{\mu})$$

where V is the diagonal matrix of the $v(\mu_i)$. The update for $\boldsymbol{\beta}$ iteration is still obtained from a linear regression of the residuals on $\dot{\boldsymbol{\mu}}$, but now the observations are weighted inversely according to their variances.

Classical GLMs assume a link-linear model of the form

$$h(\mu_i) = \mathbf{x}_i^T \boldsymbol{\beta}$$

for some link function h. In that case the Fisher scoring update can be re-organized as

$$\boldsymbol{\beta}_{k+1} = (X^T W X)^{-1} X^T W \mathbf{z}$$

where $\mathbf{z}$ is a working vector with components $z_i = h'(\mu_i)(y_i - \mu_i) + h(\mu_i)$ and W is a diagonal matrix of working weights $1/[h''(\mu_i)^2 v(\mu_i)]$. The updated $\boldsymbol{\beta}$ is obtained from weighted linear regression of the working vector $\mathbf{z}$ on X. Since X remains the same throughout the iteration, but the working weights change, this iteration is known as *Iteratively Reweighted Least Squares* (IRLS).

When the observations y_i follow an exponential family distribution, observed and expected information coincide so that Fisher scoring is the same as Newton's method. For GLMs this is so if h is the *canonical link*. We can conclude from this that IRLS is quadratically convergent for logistic regression and log-linear models, but linearly convergent for binomial regression with a probit link for example. In practice, rapid linear regression is difficult to distinguish from quadratic convergence.

Non-Derivative Methods

The Nelder-Mead *downhill simplex algorithm* is a popular derivative-free optimization method. It is based on the idea of function comparisons amongst a simplex of $N+1$ points.

Depending on the function values, the simplex is reflected or shrunk away from the maximum point. Although there are no theoretical results on the convergence of the algorithm, it works very well on a range of practical problems. It is a good choice when you want a once-off solution with minimum programming effort.

If you are prepared to use a more complex program, the best performing methods for optimization without derivatives are quasi-Newton methods with difference approximations for the gradient vector. These programs require only the objective function as input, and compute difference approximations for the derivatives internally. Note that this is different from computing numerical derivatives and inputing them as derivatives to a program designed to accept analytic derivatives. Such a strategy is unlikely to be successful, as the numerical derivatives are unlikely to show the assumed analytic behavior.

A close competitor to the finite-difference methods are *direction set methods*. These methods perform one-dimensional line searches in a series of directions which are chosen to be approximately *conjugate*, that is orthogonal with respect to the second derivative matrix. The best current implementation is given by Brent [1].

EM Algorithm

The EM algorithm is not an optimization method in its own right, but rather a statistical method of making optimization easier. The idea is the possibility that the log-likelihood $\ell(\mathbf{y};\boldsymbol{\theta})$ might be easier to maximize if there were additional observations or information. Let $\mathbf{z}$ be the completed data, and let $\ell(\mathbf{z};\boldsymbol{\theta})$ be the log-likelihood for $\mathbf{z}$. Maximizing the *incomplete* likelihood $\ell(\mathbf{y};\boldsymbol{\theta})$ is equivalent to maximizing the conditional expectation of the *complete* likelihood given $\mathbf{y}$, $E[\ell(\mathbf{z};\boldsymbol{\theta})|\mathbf{y})$. In most cases, the EM is applied when the complete likelihood can be maximized in one-step. However the conditional expectation changes with the updated estimate of $\boldsymbol{\theta}$. So the optimization proceeds by alternate steps of expectation and maximization, hence the name "EM".

The EM algorithm is linearly convergent, at a rate which depends on the proportion of observed to unobserved Fisher information. Let ρ be the fraction of the Fisher information for a particular parameter in the complete log-likelihood $\ell(\mathbf{z};\boldsymbol{\beta})$ which is not actually observed. Then the error in the estimate for that parameter after each iteration is $\epsilon_{k+1} \approx \rho\epsilon_k$. The proportion ρ can in applications be very close, or even equal, to one for some parameters, so that convergence can be very slow. On the other hand, the EM algorithm normally converges even from poor starting values. The iteration can often be speeded up by *Aitkin acceleration*, which attempts to convert linear convergence into quadratic convergence [8, page 92].

Software

Optimization software is included in the commercial subroutine libraries IMSL and NAG, and in many statistical programs such SAS, S-Plus, MATLAB and Gauss. Publicly available software can be obtained by searching the NETLIB online library at

`http://www.netlib.org/`

The guides and software provided by the Optimization Technology Center at the Argonne National Laboratory at the URL

`http://www.mcs.anl.gov/home/otc/`

are also worth considering. Less elaborate programs suitable for user modification can be found in Numerical Recipes [8].

References

[1] Brent, R. P. (1973). *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ.

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

[3] Fletcher, R. (1987). *Practical Methods of Optimization.* Second Edition, John Wiley, New York.

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

[5] Moré, J. J., and Wright, W. J. (1993). *Optimization software guide.* Society for Industrial and Applied Mathematics, Philadelphia.

[6] Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative solution of nonlinear equations in several variables.* Academic Press, New York.

[7] Rao, C. R. (1973). *Linear Statistical Inference, Second Edition.* Wiley, New York.

[8] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical recipes in Fortran.* Cambridge University Press. (Also available for C, Basic and Pascal.)

[9] Seber, G. A. F., and Wild, C. J. (1989). *Nonlinear Regression*, John Wiley, New York.

[10] Smyth, G. K. (1996). Partitioned algorithms for maximum likelihood and other nonlinear estimation. *Statist. Comput.*, **6**, 201–216.

Related Articles

Numerical Analysis