Optimization

Gordon K. Smyth

Volume 3, pp 1481–1487

in

Encyclopedia of Environmetrics
(ISBN 0471 899976)

Edited by

Abdel H. El-Shaarawi and Walter W. Piegorsch

Optimization

Optimization is the process by which one finds that value of a vector $\mathbf{x}$, say, that 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,

$$\mathbf{g}(\mathbf{x}) = \mathbf{0} \tag{1}$$

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 invariably proceeds by iteration. Starting from an approximate trial solution, a useful algorithm will gradually refine the working estimate until a predetermined level of precision has been reached. If the function is smooth, a good algorithm can be expected to converge to a maxima or minima 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 in stages from the simple to the complex. 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 (GLMs), it is possible to use the datum point itself as a starting value for the fitted values.

An extremum (maxima or minima) 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. **Simulated annealing** and **genetic algorithms** are relatively recent types of algorithms which are often used successfully on problems where there are a large number of closely competing local extrema.

This entry 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 Lagrangian multipliers (*see* **Constrained optimization**).

One must choose between algorithms that use derivatives and those that do not. In general, methods that use derivatives are the more powerful. However, the increase in speed does not always outweigh the extra overhead in computing the derivatives, and it can be a great convenience for the user not to have to program them. In some special cases it is possible to generate analytical derivatives directly from the code defining the function by using **automatic differentiation**.

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

If one can calculate first and second derivatives of f, then the well-known Newtonian 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-Newtonian methods, of which Fisher's method of scoring is one, can be recommended. General purpose quasi-Newtonian 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-Newtonian methods but require only storage of order N [6, 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 Newtonian methods with finite-difference approximations to the derivatives should minimize most functions with fewer function evaluations. Although 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 that contains the solution, there exist efficient 'low-tech' algorithms robust enough to take on all problems.

A practical introduction to optimization is given in Chapters 10 and 15 (Sections 15.5 and 15.7) of [6]. More specialist texts include [2]–[4]. A survey of available software is given by [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 minimum of $f(x)$ is bracketed by a triplet of values, $a < b < c$, if $f(b)$ is less than both $f(a)$ and $f(c)$ (see Figure 1).

The simplest and most robust 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 midpoint of the triplet to the farther endpoint (point D in 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 midpoint 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 $(c - a) \times 0.61803^k$.

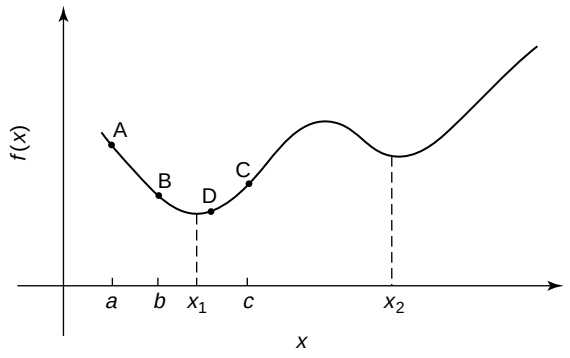

Figure 1 The golden search method. The function $f(x)$ has a local minimum at x_2 and a global minimum at x_1. The points A, B, and C bracket the global minimum. The next point tried by a golden section search would be D

The golden section search is a linear method 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 (see below), the method of false position, Muller's method, and Ridder's method, which are capable of superlinear convergence, wherein 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 golden search when the function g is smooth, but they may 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 search methods can be guaranteed, but higher-order convergence can be enjoyed when it is possible. Brent [1] has published methods that do the necessary bookkeeping to achieve this and that can be generally recommended for root finding or minimizing in one dimension [6]. 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 [6].

Newton's Method

The most celebrated of all methods for solving a nonlinear equation is Newton's method, also called the Newton–Raphson method. 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 $\mathbf{G}(\mathbf{x})$ be the matrix of second derivatives of $f(\mathbf{x})$ with respect to $\mathbf{x}$. Using the root of the linear expansion

as the new approximation gives

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{G}(\mathbf{x}_k)^{-1}\mathbf{g}(\mathbf{x}_k) \qquad (2)$$

Newton's method is illustrated in Figure 2(a). The same algorithm arises by approximating f with its quadratic Taylor series expansion about $\mathbf{x}_k$ and finding the local extrema of the quadratic. 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 $-\mathbf{G}$ is the observed **information matrix**. Newton's method for maximizing the likelihood is based on the same quadratic expansion that 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 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 2b). In fact, the set of values for which Newton's method does and does not converge can produce a fractal pattern [6] (*see* **Fractal dimensions**).

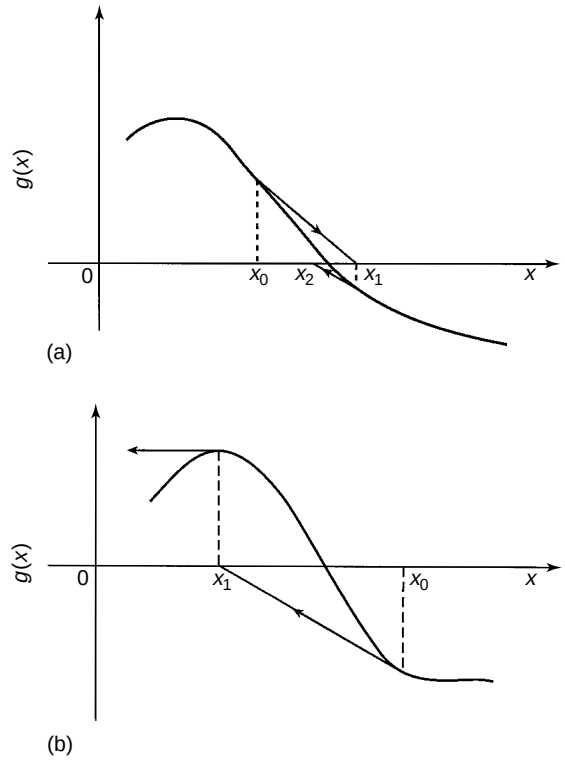

Figure 2 Newton's method: (a) quadratic convergence from starting point x_0; (b) divergence from starting value x_0

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

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{G}(\mathbf{x}_k) + \lambda_k \mathbf{I}]^{-1}\mathbf{g}(\mathbf{x}_k) \qquad (3)$$

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.

If $\mathbf{G}(\mathbf{x}_k)$ can be guaranteed to be positive definite without the addition of a diagonal matrix, then an alternative and popular strategy is to use a line search suboptimization. In this case we can replace the Newton step with

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{G}(\mathbf{x}_k)^{-1}\mathbf{g}(\mathbf{x}_k) \qquad (4)$$

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 analytical derivative $\mathbf{G}$ 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

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1}\mathbf{g}(\mathbf{x}_k) \tag{5}$$

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

$$a_k = \frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}} \tag{6}$$

Such an iteration is called a quasi-Newton method. If $\mathbf{A}_k$ is positive definite, as it usually is, an alternative name is variable metric method.

One positive advantage to using an approximation in place of $\mathbf{G}$ is that $\mathbf{A}_k$ 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 $\mathbf{A}_k^{-1}\mathbf{g}(\mathbf{x}_k)$ 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-Newton algorithms, the best is probably the Broydon–Fletcher–Goldfarb–Shanno (BFGS) algorithm. The BFGS algorithm builds upon the earlier and similar Davidon–Fletcher–Powell (DFP) algorithm. The BFGS algorithm starts with a positive definite matrix approximation to $\mathbf{G}(\mathbf{x}_0)$, usually the identity matrix. At each iteration it makes a minimalist (rank two) modification to $\mathbf{A}_k^{-1}$ to gradually approximate $\mathbf{G}(\mathbf{x}_k)^{-1}$. DFP and BFGS are both robust algorithms showing superlinear convergence.

Statisticians might fall into the trap when using algorithms such as BFGS or DFP of thinking that the final approximation $\mathbf{A}_k^{-1}$ is a good approximation to $\mathbf{G}^{-1}(\mathbf{x}_k)$ at the final estimate. Since $\mathbf{A}_k$ is chosen to approximate $\mathbf{G}(\mathbf{x}_k)$ only in the directions needed for the Newtonian step, however, 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 $-\mathbf{G}$ is the observed information matrix. For many models (curved exponential families are the major class) the Fisher information, $\mathcal{I}(\mathbf{x}) = E[-\mathbf{G}(\mathbf{x})]$, is much simpler in form than $-\mathrm{G}(\mathbf{x})$ 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 $-\mathbf{G}(\mathbf{x})$ with $\mathcal{I}(\mathbf{x})$ is known as Fisher's method of scoring [7, Section 5g]. Fisher scoring is linearly convergent, at a rate that depends on the relative difference between observed and expected information [9].

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

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{n}[y_i - \mu(\mathbf{t}_i, \boldsymbol{\beta})]^2 \tag{7}$$

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

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + (\dot{\boldsymbol{\mu}}^{\mathrm{T}}\dot{\boldsymbol{\mu}})^{-1}\dot{\boldsymbol{\mu}}^{\mathrm{T}}(\mathbf{y} - \boldsymbol{\mu}) \tag{8}$$

where all terms on the right-hand side 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 $\dot{\boldsymbol{\mu}}$. The Gauss–Newton algorithm 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

$$\mu(t_i, \boldsymbol{\beta}) = \beta_1 \mathbf{e}^{-\beta_3 t_i} + \beta_2 \mathbf{e}^{-\beta_4 t_i} \tag{9}$$

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 [8, Section 14.7]. The same principle is discussed in the context of maximum likelihood estimation in [9].

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 (*see* **Categorical data**) as special cases. GLMs assume that y_i is distributed according to a probability density or mass function of the form

$$\Pr(y;\theta_i,\sigma^2) = a(y,\sigma^2)\exp\left\{\frac{1}{\sigma^2}[y\theta_i - b(\theta_i)]\right\} \quad (10)$$

for some functions b and a (a curved exponential family). We find that $\mathrm{E}(y_i) = \mu_i = b'(\theta_i)$ 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 nonlinear least squares, then the Fisher scoring iteration for $\boldsymbol{\beta}$ is a slight modification of the Gauss–Newton iteration

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + (\dot{\boldsymbol{\mu}}^{\mathrm{T}}\mathbf{V}^{-1}\dot{\boldsymbol{\mu}})^{-1}\dot{\boldsymbol{\mu}}^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{y}-\boldsymbol{\mu}) \quad (11)$$

where $\mathbf{V}$ is the diagonal matrix of the $v(\mu_i)$. The update for $\boldsymbol{\beta}$ 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

$$h(\mu_i) = \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta} \quad (12)$$

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

$$\boldsymbol{\beta}_{k+1} = (\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{z} \quad (13)$$

where $\mathbf{z}$ is a working vector with components $z_i = h'(\mu_i)(y_i - \mu_i) + h(\mu_i)$, and $\mathbf{W}$ is the 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 $\mathbf{X}$. Since $\mathbf{X}$ 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 that is defined by $h(\mu_i) = \theta_i$. We can conclude from this that IRLS is quadratically convergent for logistic regression and log-linear models but is linearly convergent for binomial regression with a probit link (*see* **Probit model**), for example. In practice, rapid linear convergence is difficult to distinguish from quadratic convergence.

Nonderivative Methods

The Nelder–Mead downhill simplex algorithm is a popular derivative-free optimization method. It is based on the idea of function comparisons among 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 a one-off solution is wanted with minimum programming effort. It can also be used to minimize functions that are not differentiable.

If the user is 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 analytical derivatives. Such a strategy is unlikely to be successful, as the numerical derivatives are unlikely to show the assumed analytical 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 that are chosen to be approximately orthogonal with respect to the second-derivative matrix. The best current implementation is given by Brent [1].

Another optimization strategy that usually does not require derivatives is the expectation–maximization **(EM) algorithm**. Properly speaking, this is not an optimization method in its own right but rather is a statistical method of making optimization easier. The idea is to view the dataset as an incompletely observed version of a larger dataset for which the optimization would be very easy. One maximizes the expectation of the log-likelihood of the larger data-set, given the observed data. Since the expectation itself depends on the unknown parameters, and these are updated by the maximization, it is necessary to iterate between the maximization and the expectation until convergence. The EM algorithm converges linearly at a rate determined by the proportion of the completed data that is actually observed. Compared with derivative-based optimization methods, the EM algorithm tends to converge slowly but surely.

Simulated annealing and genetic algorithms are designed to minimize functions that are not smooth and that may have many local minima. Simulated annealing algorithms introduce a random element into the iteration process, giving the algorithm a chance to escape from a local extremum. Genetic algorithms carry information about multiple candidates for the global extremum that are simultaneously refined as the iteration proceeds.

Software

Optimization software is included in the commercial subroutine libraries IMSL and NAG, and in many statistical programs such as **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

`http://www-fp.mcs.anl.gov/otc/Guide/`

are also worth considering. Less elaborate programs suitable for user modification can be found in [6]; see

`http://www.nr.com/`

An alternative decision tree for choosing software, with special attention to global optimization, is given at

`http://plato.la.asu.edu/guide.html/`

References

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

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

[3] Fletcher, R. (1987). *Practical Methods of Optimization*, 2nd Edition, Wiley, New York.

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

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

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

[7] Rao, C.R. (1973). *Linear Statistical Inference*, 2nd Edition, Wiley, New York.

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

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

GORDON K. SMYTH