Minimization Algorithms for Variational Data Assimilation

M Fisher

Abstract

An introduction to the main minimization algorithms used in variational data assimilation is presented. For the important case of a strictly quadratic cost function, the conjugate gradient algorithm is the natural choice. The algorithm is discussed in some detail, and a statistical interpretation of its well-known optimality property is given. The connection between conjugate gradients and the Lanczos algorithm is exploited to show how the minimization of quadratic cost functions may be accelerated. For nearly-quadratic functions, Newton methods are considered, and a new generalized truncated Newton algorithm is presented. The new algorithm is shown to be competitive with a widely-used quasi-Newton code, and provides a way to guarantee the convergence of incremental assimilation.

Introduction

Many problems in meteorology may be expressed in terms of the minimization of a function. However, it is the development of variational data assimilation which has given particular emphasis to the need for efficient minimization algorithms. Variational data assimilation, especially 4dVar, is computationally expensive. Moreover, the computational cost is directly related to the efficiency of the algorithm used. There is consequently a strong incentive to develop efficient algorithms which are adapted to the particular characteristics of the analysis problem.

This paper is intended partly as an introduction to the main minimization algorithms which are used in variational data assimilation. In the next section, the main characteristics of the problem, are discussed. Following this, the paper is divided into two main parts. The first part considers purely quadratic cost functions. These are particularly important, since quadratic cost functions appear as the functions minimized during the inner iterations of the incremental method (Courtier *et al.*, 1994), and whenever the analysis depends linearly on the observed and background data. The conjugate gradient method is the natural algorithm for quadratic functions, and is discussed in some detail. An interpretation of the well-known convergence properties of conjugate gradients is given in terms of the likelihood of the analysis error at each iteration. Preconditioning is discussed, and the connection between the Lanczos algorithm and conjugate gradients is examined. Following this, the effect of rounding error on the convergence of the minimization is considered. The main result of this part of the paper is that the convergence of conjugate gradients may be significantly accelerated by suppressing the effects of rounding error.

The next part of the paper considers nearly quadratic cost functions. A brief introduction is given to Newton algorithms in general, and to quasi-Newton algorithms in particular. However, most of the section is given over to a discussion of truncated Newton methods. These are less well known than quasi-Newton methods, and are particularly well suited for use in variational data assimilation. A new, generalized truncated Newton algorithm is presented. The algorithm is similar in spirit to the incremental method, but has the advantage of guaranteeing convergence of the outer iterations. The algorithm is evaluated in the ECMWF 4dVar analysis system, and is shown to be competitive with the currently-used quasi-Newton algorithm.

Characteristics of Variational Data Assimilation

Four dimensional variational data assimilation (4dVar) defines the analysis as the initial model state x_0 which minimizes a cost function:

$$J(x_0) = (x_0 - x^b)^T B^{-1} (x_0 - x^b) + \sum_{i=0}^{M} (H_i(x_i) - y_i) R_i^{-1} (H_i(x_i) - y_i) + J_c(x_0) \tag{1}$$

where the sum is over a sequence of "time windows" of length a few timesteps of the forecast model. The model state for each time window, x_i, is calculated by a short integration of the forecast model with initial conditions x_{i-1}. (Three dimensional variational data assimilation (3dVar) may be regarded as a special case of 4dVar for which $M = 0$.) For each time window, there is a vector of observations, y_i. The observations are compared with corresponding values calculated from the model variables by the application of the operators H_i. In general, these "observation operators" are non-linear.

The first term on the right hand side of equation 1 measures the discrepancy between the initial model state x_0, and a "background" state, typically provided by the state x_M of the previous analysis cycle. The weight matrix B is the covariance matrix of errors in the background state.

The sum in equation 1 represents a weighted sum of squares of the discrepancy between the model states and the observations for each time window. The weights are provided by the matrices R_i, which are covariance matrices for the errors in $(H_i(x_i^t) - y_i)$, where x_i^t is the discretized representation of the true state of the atmosphere.

The third term on the right hand side of equation 1 represents prior information, and typically penalizes departures from a balanced state.

In practice, minimization of the cost function defined in equation 1 is computationally very expensive. Courtier *et al.* (1994) introduced a method which greatly reduces the computational cost. Their "incremental" formulation involves two approximations. First, the observation operators and the forecast model are linearized about a model trajectory $\{x_i^n; i = 0 \dots M\}$. This allows equation 1 to be replaced by a quadratic cost function for the analysis increments δx_0^n. That is, an approximation to the analysis is given by $x_0^n + \delta x_0^n$, where δx_0^n minimizes

$$J(\delta x_0^n) = (\delta x_0^n + x_0^n - x^b)^T B^{-1} (\delta x_0^n + x_0^n - x^b) + \sum_{i=0}^{M} (H_i' \delta x_i^n - d_i^n) R_i^{-1} (H_i' \delta x_i^n - d_i^n) + J_c(\delta x_0^n) \tag{2}$$

Here, δx_i^n is given by an integration of the tangent linear model with initial conditions δx_{i-1}^n, $d_i^n = y_i - H_i(x_i^n)$, and H'_i is the linearization of H_i about x_i^n.

The second approximation in deriving the incremental formulation of 4dVar is to introduce a simplification operator S. Typically, this is a linear operator such as spectral truncation which reduces the spatial resolution of the model state. The linearized observation operators, background error covariance matrix, J_c, and the tangent linear model are replaced by low resolution versions (indicated below by hats) to give the cost function

$$\hat{J}(\delta\hat{x}_0^n) = (\delta\hat{x}_0^n + S(x_0^n - x^b))^T \hat{B}^{-1}(\delta\hat{x}_0^n + S(x_0^n - x^b)) \tag{3}$$

$$+ \sum_{i=0}^{M} (\hat{H}_i\delta\hat{x}_i^n - d_i^n) R_i^{-1} (\hat{H}_i\delta\hat{x}_i^n - d_i^n) \quad + \quad \hat{J}_c(\delta x_0^n)$$

The approximate analysis is defined as $x_0^n + S^+\delta\hat{x}_0^n$, where S^+ is the pseudo-inverse of the simplification operator. Once the approximate analysis has been determined, it may be used to define a new trajectory, $\{x_i^{n+1}; i = 0 \ldots M\}$, about which the cost function may be linearized and a new analysis increment $\delta\hat{x}_i^{n+1}$ determined. The process may be repeated to give an iterative algorithm. Note, however, that there is no guarantee that the iterations will converge!

There are two main advantages of the incremental formulation. First, the ability to use a reduced resolution tangent linear model during the minimization greatly reduces the computational cost of the analysis. Second, the cost function becomes strictly quadratic. We will see below that significant increases in the convergence rate of the minimization are possible if advantage is taken of this property. In practice, it may be convenient to forego the advantages of a quadratic cost function and to allow some non-linearities to remain in the simplified cost function. In the current ECMWF implementation of 4dVar (Rabier *et al.*, 1999, Mahfouf and Rabier, 1999; Klinker *et al.*, 1999), the cost function is not strictly quadratic due to the non-quadratic cost function used for scatterometer observations (Stoffelen and Andersson, 1997), and the use of variational quality control. Nevertheless, the behaviour of the cost function remains approximately quadratic in the vicinity of the analysed model state.

The remainder of this paper will ignore many of the details of incremental 4dVar analysis, and consider the production of the analysis purely as a minimization problem. From this point of view, the main characteristics of the problem are as follows. First, the dimension of the control vector $\delta\hat{x}_i^n$ is large (typically 10^5 to 10^6 elements). Second, the function to be minimized is quadratic or nearly quadratic. Third, nearly all the computational expense is taken up in the calculation of the cost function and its gradient. Finally, the matrices $\hat{B}$ and R_i are covariance matrices. As a consequence, the Hessian matrix of the cost function for the strictly quadratic problem is positive definite.

These characteristics have implications for the sorts of algorithms one should use to minimize the cost function. The quadratic or nearly-quadratic nature of the cost function suggests that algorithms based on quadratic models, such as conjugate gradient and Newton methods, should be successful. However, the large dimension of the control vector makes explicit storage and manipulation of the Hessian matrix impossible. The high computational cost of the function and gradient calculations means that algorithms which extract as much information as possible about the shape of the cost function from relatively few function and gradient evaluations are preferred.

Strictly Quadratic Cost Functions

For strictly quadratic cost functions, it makes sense to replace the problem of minimizing J by the equivalent problem of solving the linear system of equations $\nabla J(x) = 0$. There is a vast literature on this subject. The book by Barrett *et al.* (1994) is one of many good introductions to the field, and has the advantage of being available on the World Wide Web. The methods available may be divided into direct and iterative techniques.

Direct Solution Methods

Direct methods are out of the question unless the linear system is modified to split the problem into a set of smaller problems. One way to achieve this is to divide the physical domain of the problem into regions which are approximately independent. This approach leads to analysis algorithms which resemble Optimal Interpolation (Hollingsworth, 1987). The disadvantage of this approach (and of OI) is that it introduces artificial boundaries and data selection into the problem. The resulting analysis is consequently non-global and sub-optimal. Nevertheless, the approach may lead to particularly efficient preconditioners for use with iterative solution methods. This method of preconditioning is particularly attractive in versions of variational data assimilation in which the control variable is defined in observation space, rather than in model variable space. In these systems, the analysis equation takes the form

$$x^a = x^b + BH^T z \tag{4}$$

where

$$z = (R + HBH^T)^{-1}(y - Hx^b) \tag{5}$$

The matrix $(R + HBH^T)$ is essentially the same matrix which must be inverted in an Optimal Interpolation analysis system. Domain decomposition techniques for solving equation 5 using direct methods may be put to good use to precondition the iterative solution of this equation. The technique is used to good effect in the observation space 3dVar analysis system currently under development for use with the US Navy Operational Global Atmospheric Prediction System (Daley, personal communication).

Iterative Solution Methods

The Hessian of the cost function is symmetric and positive definite. This makes conjugate gradients the iterative method of choice. Schewchuk (1994) gives a good pedagogical introduction to the method, which is available in electronic form.

The conjugate gradient algorithm is remarkably simple. It requires an initial estimate $x^{(0)}$ of the solution, the gradient $g^{(0)} = \nabla J(x^{(0)})$ at the initial point, and a descent direction $d^{(0)}$, for which the usual choice is $d^{(0)} = -g^{(0)}$. Improved estimates of the solution are generated by iterating the following steps:

1. $\alpha_k = \dfrac{\langle g^{(k)}, g^{(k)} \rangle}{\langle d^{(k)}, J'' d^{(k)} \rangle}$ Step to the line minimum of J.
2. $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ New estimate for the solution.
3. $g^{(k+1)} = g^{(k)} + \alpha_k J'' d^{(k)}$ Gradient at the new point.
4. $\beta_k = \dfrac{\langle g^{(k+1)}, g^{(k+1)} \rangle}{\langle g^{(k)}, g^{(k)} \rangle}$
5. $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$ New descent direction.

Here $\langle .,. \rangle$ denotes the inner product with respect to which the gradient is defined.

At each iteration, we must calculate $J'' d^{(k)}$. This can be done using a single gradient calculation. One method is

$$J''\mathrm{d}^{(k)} = \frac{1}{\tau}(\nabla(J(\mathrm{x}^{(0)} + \tau\mathrm{d}^{(k)}) - \mathrm{g}^{(0)})) \tag{6}$$

where τ should be chosen large enough to avoid rounding error problems.

The importance of conjugate gradients stems from the following result (see for example Shewchuk, 1994). Let λ_i and v_i be the eigenvalues and corresponding eigenvectors of the Hessian matrix of the analysis cost function. For convenience of notation, define the Hessian norm $\|\mathrm{x}\|_{J''} = (\mathrm{x}^T J''\mathrm{x})^{1/2}$, and denote the polynomial of order k with real coefficients $c_1 \ldots c_k$, which takes the value one at the origin, by

$$P_k(x;c_1\ldots c_k) \equiv 1 + c_1 x + c_2 x^2 + \ldots + c_k x^k \tag{7}$$

We may expand the difference between the initial approximation to the analysis and the true solution of $\nabla J(\mathrm{x}) = 0$ in terms of the eigenvectors as:

$$\mathrm{e}^{(0)} \equiv \mathrm{x}^{(0)} - \mathrm{x}^a = \sum_j \xi_j \mathrm{v}_j \tag{8}$$

It is easy to show that the corresponding expression for the error at the kth iteration for the conjugate gradient algorithm, along with many other iterative methods, is given by

$$\mathrm{e}^{(k)} \equiv \mathrm{x}^{(k)} - \mathrm{x}^a = P_k(J'';c_1\ldots c_k)\mathrm{e}^{(0)} \tag{9}$$

$$= \sum_j \xi_j P_k(\lambda_j;c_1\ldots c_k)\mathrm{v}_j$$

for some polynomial $P_k(.;c_1\ldots c_k)$.

The Hessian norm of the error is

$$\|\mathrm{e}^{(k)}\|^2_{J''} = \sum_j \lambda_j \xi_j^2 [P_k(\lambda_j;c_1\ldots c_k)]^2 \tag{10}$$

The conjugate gradient algorithm has a certain optimality in reducing the error in the solution in that it minimizes the Hessian norm of the error over *all* polynomials $P_k(.;c_1\ldots c_k)$.

In variational assimilation, the Hessian norm has a natural interpretation since, provided the background and observation error covariance matrices are correctly specified, the Hessian matrix is the inverse of the covariance matrix of analysis error, A (Thépaut and Moll, 1990). If we assume that analysis errors are Gaussian, then the conditional probability of $\mathrm{x}^{(k)}$, assuming that the exact solution of the analysis equation coincides with the true state of the atmosphere, is

$$\log(p(\mathrm{x}^{(k)}|\mathrm{x}^a)) = -\frac{1}{2}(\mathrm{e}^{(k)})^T \mathrm{A}^{-1}\mathrm{e}^{(k)} + c \tag{11}$$

$$= -\frac{1}{2}\|\mathrm{e}^{(k)}\|^2_{J''} + c$$

There are two sources of error in a correctly tuned variational analysis system. The first is the inevitable error due to inaccuracies in the observations and in the background. If the covariance matrices are correctly specified, then the exact solution of the analysis equation maximizes the likelihood of the difference between the exact solution and the true state of the atmosphere (Daley, 1991).

The second source of error arises from the fact that the iterative algorithm is stopped before full convergence has been achieved. This additional error is the discrepancy $e^{(k)}$. Equation 11 shows that the conjugate gradient algorithm maximizes the conditional likelihood of this additional analysis error over all expressions of the form given by equation 9.

The rate of convergence of the conjugate gradient algorithm depends in a complicated way on the distribution of the eigenvalues of J''. However, an upper bound on the norm of the error, which depends only on the condition number κ of the Hessian matrix (i.e. the ratio of its largest and smallest eigenvalues), is given by (see e.g. Shewchuk, 1994)

$$\|e^{(k)}\|_{J''}^2 \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e^{(0)}\|_{J''}^2 \tag{12}$$

Preconditioning

To achieve acceptable convergence rates, it is necessary to precondition the conjugate gradient algorithm. There are at least four ways to do this: a suitable inner product may be chosen; an explicit change-of-variable may be employed; an explicit preconditioner (i.e. an approximation to the inverse of the Hessian matrix) may be used; or an implicit preconditioner (i.e an approximation to the Hessian matrix itself) may be used.

Preconditioning via the inner product requires us to choose an inner product which makes the Hessian matrix close to the identity matrix. Ideally, we should use the inner product defined by the Hessian matrix itself. It is easy to see why. Let us write the cost function in the form

$$J(x) = J(0) + b^T x + \frac{1}{2} x^T K x \tag{13}$$

The vector b and the matrix K are the gradient at $x = 0$ and the Hessian matrix *with respect to the Euclidean inner product.* In terms of a general inner product, we have

$$J(x) = J(0) + \langle \nabla J(0), x \rangle + \frac{1}{2} \langle x, J'' x \rangle \tag{14}$$

Equating the terms in these two expressions for the cost function, we see in particular that

$$x^T K x = \langle x, J'' x \rangle \tag{15}$$

This may be regarded as the definition of the Hessian matrix J'' for the inner product $\langle .,. \rangle$. If we now choose to define the inner product of any pair of vectors u and v by $\langle u,v \rangle \equiv u^T K v$, then we must have $J'' = I$. The condition number of the Hessian in this case is one, and the conjugate gradient algorithm converges in a single iteration.

To precondition via a change of variable, the entire cost function is expressed in terms of a new variable $\chi = L(x - x^b)$. (For simplicity, the non-incremental formulation is considered.) The background error covariance matrix and observation operators must be suitably modified. The analysis then becomes a two-stage process. The analysis equation is first solved to determine the analysis in terms of the new variable, after which the inverse of the change of variable matrix L is applied to determine the analysis in terms of model variables.

Provided the covariance matrix for the variable χ is correctly specified, the statistical optimality of the analysis is not altered by the change of variable. Indeed, the exact solution of the analysis equation is identical. Furthermore, since the probability of the iteration error $e^{(k)}$ is independent of the particular variable in which it is expressed, it is still the case that the likelihood of this additional source of analysis error is maximized over all expressions of the form given by equation 9. What changes, however, is the distribution of the eigenvalues of the Hessian. Consequently, a good choice of control variable can give faster convergence towards the exact solution of the analysis equation.

Preconditioning by a change of variable is the method currently used in the ECMWF analysis system. In this case, the change of variable is defined as $L = B^{-1/2}$. This has the particular advantage that the background error covariance matrix for the control variable is simply the identity matrix. However, the condition number of the Hessian with this change of variable is relatively large (a few thousand), so the convergence of the minimization is rather slow.

To use an explicit preconditioner, the simple conjugate gradient algorithm described above must be replaced by the slightly more complicated preconditioned algorithm (see, e.g. Axelsson, 1994). The preconditioner must be an approximation for the inverse of the Hessian matrix $(J'')^{-1} = (B^{-1} + H^T R^{-1} H)^{-1}$. To date, there is little practical experience to indicate how such an approximation might be generated. One possibility is to use the Shermann-Morrison-Woodbury formula (Sherman and Morrison, 1949) to rewrite the inverse Hessian as

$$(J'')^{-1} = B - BH^T(R + HBH^T)^{-1}HB \tag{16}$$

The matrix $(R + HBH^T)$ may then be inverted using domain decomposition techniques familiar from Optimal Interpolation schemes. Unfortunately, the presence of the minus sign in equation 16 makes it difficult to ensure that the resulting approximation to the inverse of the Hessian matrix is positive definite.

Implicit preconditioners are approximations to the Hessian matrix itself, rather than to its inverse. The algorithm requires the solution at every iteration of systems of linear equations involving the approximate Hessian matrix. If these equations are solved accurately, then the standard preconditioned conjugate gradient algorithm may be used. However, they may also be solved approximately using, for example, a few iterations of a conjugate gradient algorithm. In this case, the preconditioner may no longer be regarded as applying a fixed matrix at each iteration, and the usual preconditioned conjugate algorithm is no longer appropriate. Axelsson (1994) describes generalized conjugate gradient algorithms which may be used in this case.

Clearly, the equations involving the approximate Hessian matrix must be computationally inexpensive to solve. One possibility is to note that the matrix $H^T R^{-1} H$ is likely to be rather well approximated by a diagonal or banded matrix. Indeed, in 3dVar, the matrix is exactly diagonal if the observation error covariance matrix is itself diagonal, and if all observations are at model gridpoints and are observations of model variables. Of course, it is not generally straightforward to generate a suitable banded approximation to $H^T R^{-1} H$. Moreover, the presence of the inverse of the background error covariance matrix in the expression for the Hessian may make it computationally too expensive to give much benefit in reducing the computational cost of the minimization.

The Lanczos Connection

The connection between the Lanczos algorithm and conjugate gradient minimization is well known (e.g. Paige and Saunders, 1975). Specifically, we may combine the equations for steps 2, 3 and 5 of the conjugate gradient algorithm given above to eliminate both the descent directions, $d^{(k)}$, and the iterates $x^{(k)}$. The resulting equation involves only gradient vectors, and is usually expressed in terms of the normalized gradients $q^{(k)} = g^{(k)} / \|g^{(k)}\|$:

$$J''\mathrm{q}^{(k+1)} = \gamma_{k+1}\mathrm{q}^{(k+2)} + \delta_{k+1}\mathrm{q}^{(k+1)} + \gamma_k\mathrm{q}^{(k)} \tag{17}$$

The coefficients γ_k and δ_k are related to the coefficients α_k and β_k of the conjugate gradient algorithm by

$$\begin{aligned}\gamma_k &= \frac{\sqrt{\beta_{k+1}}}{\alpha_k} \\ \delta_{k+1} &= \frac{1}{\alpha_{k+1}} + \frac{\beta_{k+1}}{\alpha_k}\end{aligned} \tag{18}$$

Equation 17 may be written in matrix form as

$$J''\mathrm{Q}^{(k)} = \mathrm{Q}^{(k)}\mathrm{T}^{(k)} + \gamma_k\mathrm{q}^{(k+1)}(u^{(k)})^T \tag{19}$$

where $\mathrm{Q}^{(k)}$ is the matrix whose columns are the normalized gradients $\mathrm{q}^{(k)}$, $(\mathrm{u}^{(k)})^T = (0, \ldots, 0, 1)$ and $\mathrm{T}^{(k)}$ is the tridiagonal matrix

$$\mathrm{T}^{(k)} = \begin{pmatrix} \delta_1 & \gamma_1 & & & \\ \gamma_1 & \delta_2 & \gamma_2 & & \\ & \cdot & \cdot & \cdot & \\ & & \gamma_{k-2} & \delta_{k-1} & \gamma_{k-1} \\ & & & \gamma_{k-1} & \delta_k \end{pmatrix} \tag{20}$$

The eigenvalues of $\mathrm{T}^{(k)}$ converge to the eigenvalues of J'', and the eigenvectors of $\mathrm{T}^{(k)}$, when pre-multiplied by $\mathrm{Q}^{(k)}$ are approximate eigenvectors of J''.

The elements of the matrix $\mathrm{T}^{(k)}$, and the vectors $\mathrm{q}^{(k)}$, may be calculated from the coefficients and vectors generated during the minimization of the cost function. Consequently, good approximations to the eigenvalues and eigenvectors of the Hessian may be generated during the minimization at negligible additional computational expense. The eigenvectors and eigenvalues provide valuable information about the Hessian matrix. For example, since the extreme eigenvalues are the first to converge, it is possible to estimate the condition number of the Hessian. This allows good estimates of the expected convergence rate to be generated. (In the ECMWF system, the particular choice of change-of-variable means that the smallest eigenvalue of the Hessian matrix is equal to one. The condition number is therefore equal to the leading eigenvalue.) Explicit knowledge of the extreme eigenvectors and eigenvalues of the Hessian may also be useful in constructing preconditioners for subsequent minimizations of the cost function, and may help in the diagnosis of cases of excessively slow convergence.

The eigenvectors of the Hessian are eigenvectors of its inverse, the covariance matrix of analysis error. Eigenvectors corresponding to the largest eigenvalues of the Hessian are the patterns with the smallest variance of analysis error. Fisher and Courtier (1995) describe an algorithm (used in the current ECMWF analysis system.) which uses these eigenvectors to estimate the variance of analysis error.

In addition to its use for estimating eigenvectors and eigenvalues, the Lanczos algorithm may also be used for solving systems of linear equations. The method is based on the Newton equation

$$J''(\mathrm{x}^a - \mathrm{x}^{(0)}) = -\nabla J(\mathrm{x}^{(0)}) \tag{21}$$

If we look for solutions of the form $\mathrm{x}^{(k)} = \mathrm{x}^{(0)} + \mathrm{Q}^{(k)}\mathrm{z}^{(k)}$, then by equation 19, we have

$$\mathrm{T}^{(k)}\mathrm{z}^{(k)} \approx -(\mathrm{Q}^{(k)})^T \nabla J(\mathrm{x}^{(0)}) \tag{22}$$

This is easily solved to give an approximate value for $\mathrm{z}^{(k)}$ and hence for $\mathrm{x}^{(k)}$. The approximate solution generated in this way is identical with that generated by k iterations of conjugate gradients.

The Lanczos approach to minimizing the cost function suggests some intriguing possibilities. First, note that for a strictly quadratic cost function, the Hessian matrix depends only on the statistics of observation and background error and not on the particular observed and background values. Equation 19 may be regarded as providing an approximation to the Hessian matrix. This remains a valid approximation if the observations and the background fields are perturbed. Thus, once we have performed a single analysis, and generated the matrices $\mathrm{Q}^{(k)}$ and $\mathrm{T}^{(k)}$, we may use equation 22 to generate new analyses from perturbed data at the cost of a single gradient evaluation per additional analysis. This approach may provide a particularly efficient method for generating ensembles of analyses. Note however, that the optimality properties of the conjugate gradient algorithm apply only to the original analysis for unperturbed data. The solutions of equation 22 for perturbed backgrounds and observations might in practice be rather poor solutions of the analysis equation.

A second application of the Lanczos approach to minimizing the cost function is provided by the block version of the algorithm (see, for example O'Leary, 1980). This generalizes equation 17 by replacing the vectors $\mathrm{q}^{(k)}$ by rectangular matrices $\mathrm{V}^{(k)}$, with a small number of columns. The coefficients γ_k and δ_k are generalized to small square matrices, and the matrix $\mathrm{T}^{(k)}$ becomes block-tridiagonal. The advantage of the algorithm is that at each iteration, the Hessian matrix must be applied to all the columns of $\mathrm{V}^{(k)}$. These calculations may be performed simultaneously. The result is a parallel solution method which is well suited to modern computers. This approach to parallel minimization is used, for example, in the block conjugate gradient algorithm (O'Leary, 1987), and in the block truncated Newton algorithm (Nash and Sofer, 1989).

Rounding Errors in Conjugate Gradients

There has been surprisingly little investigation into the effects of rounding error on the conjugate gradient algorithm. Upper bounds for these effects were given by Greenbaum (1989), Strakos (1991) and Greenbaum and Strakos (1992).

The effect of rounding error in the Lanczos algorithm is better understood. It is well known that rounding error causes a loss of orthogonality among the vectors $\mathrm{q}^{(k)}$. This results in the convergence of spurious copies of already-converged eigenvalues. Notay (1993) showed that the number of extra iterations of conjugate gradients required to overcome the effects of rounding error is equal to the number of spurious copies of eigenvalues generated by the equivalent Lanczos algorithm.

In exact arithmetic, conjugate gradient minimization demonstrates super-linear convergence. That is, the convergence rate increases with the number of iterations. In practice, however, such behaviour may not materialize. According to van der Sluis and van der Vorst (1986), the super-linear convergence of conjugate gradients may be interpreted as follows. First, although the convergence rate depends in a rather complicated way on the distribution of eigenvalues, in many cases equation 12 gives a rather accurate estimate of the rate of convergence. That is, the convergence rate depends on the condition number, i.e. on the ratio of the largest to the smallest eigenvalue. At each iteration of conjugate gradients, the cost function is fully minimized in the space spanned by all the descent directions generated so far. Now, the approximate eigenvectors generated by the equivalent Lanczos process are simply linear combinations of these descent directions. So, the cost function must be fully minimized in the directions of all fully-converged eigenvectors. All subsequent descent directions will be orthogonal to the converged eigenvectors, and the minimization will behave as if it were minimizing a function in a space of reduced dimension with a Hessian from which these eigenvalues have been removed. The condition number for this reduced-dimension problem is the ratio of the largest to smallest unconverged eigenvalues. Since the Lanczos algorithm tends to converge the extreme eigenvalues first, the effective condition number decreases as the iterations proceed, and the convergence rate accelerates.

This interpretation also explains how rounding error may prevent the super-linear convergence predicted for exact arithmetic. The loss of orthogonality among the vectors $q^{(k)}$ is matched by a loss of orthogonality among the descent directions. In particular, these directions will not remain orthogonal to already-converged eigenvectors, and the effective condition number for the minimization will not decrease.

Paige (1971), Parlett and Scott (1979) and Simon (1984) demonstrate how to avoid the generation of spurious copies of eigenvalues in the Lanczos algorithm by orthogonalizing the vectors $q^{(k)}$. Notay's (*op. cit.*) identification of the number of spurious copies of eigenvalues with the number of iterations of conjugate gradients required to overcome the effects of rounding error suggests that a similar orthogonalization may reduce the number of iterations of conjugate gradients required to minimize the cost function. Furthermore, van der Sluis and van der Vorst's (*op. cit.*) interpretation of super-linear convergence suggests that explicit orthogonalization of the gradient vectors generated by the conjugate gradient algorithm may be sufficient to allow super-linear convergence to occur.

The thick solid curve in figure 1 shows the effect of explicitly orthogonalizing each new gradient vector against its predecessors using the modified Gramm-Schmidt method. This was done immediately after step 3 of the algorithm defined above. The cost function for this example was a strictly quadratic, reduced resolution version of the operational ECMWF 4dVar analysis. The resolution of the control variable in this case was T42 with 31 levels. All operationally-used observations were included. However, variational quality control was disabled, and a quadratic version of the observation cost function for scatterometer data was used. The abscissa shows the square of the Hessian norm of the difference between the control vector at each iteration, and its value at the 118th iteration of the orthogonalized algorithm. It was calculated using

$$\begin{aligned}\left\|\chi^{(k)}-\chi^{(118)}\right\|_{J''}^{2} &= (\chi^{(k)}-\chi^{(118)})^{T} J''(\chi^{(k)}-\chi^{(118)}) \\ &= (\chi^{(k)}-\chi^{(118)})^{T}(g^{(k)}-g^{(118)})\end{aligned} \tag{23}$$

Except for the last few iterations of each algorithm, this is a good estimate of the square of the Hessian norm of the true iteration error $\left\|e^{(k)}\right\|_{J''}^{2}$.

The thick dashed curve in figure 1 shows the convergence of the algorithm without orthogonalization. The curve is almost straight, indicating that the convergence rate is roughly constant in this case. The slope is similar to that of the thin straight line, which shows the upper bound on the convergence rate given by equation 12. By contrast, the orthogonalized algorithm clearly demonstrates super-linear convergence, with the slope of the curve steepening as the iterations proceed.

When the conjugate gradient algorithm is used to minimize non-quadratic functions, the coefficient β_k, which determines the descent direction is frequently calculated using the Polak-Ribiere equation (Polak and Ribiere, 1969)

$$\beta_k = \frac{\langle g^{(k+1)}, (g^{(k+1)} - g^{(k)}) \rangle}{\langle g^{(k)}, g^{(k)} \rangle} \tag{24}$$

For a quadratic function, and exact arithmetic, this gives an identical solution to the equation used in step 4 of the algorithm defined above, since the vectors $g^{(k+1)}$ and $g^{(k)}$ are orthogonal. In the presence of rounding error, however, these vectors are no longer orthogonal, resulting in a different value for the coefficient. The advantage of equation 24 for minimizing general (non-quadratic) functions is that if the minimization stagnates then β_k will be small. Consequently, the next descent direction will be close to the steepest descent direction. Effectively, the minimization is restarted.

The thin dashed line in figure 1 shows that use of equation 24 has little effect on the convergence rate for a quadratic 4dVar cost function. This is not surprising since, once orthogonality has been lost, the steepest descent direction is not orthogonal to the converged eigenvectors. Hence, equation 24 is of no assistance in preserving the orthogonality of the gradient vectors or the super-linear convergence of the minimization.

It is worth noting that all the algorithms shown in figure 1 reduce the Hessian norm of the iteration error by 6 orders of magnitude after 150 iterations. The iteration error in the solution is negligible in comparison with typical standard deviations of analysis error. For practical purposes, therefore, the conjugate gradient algorithm may be regarded as providing an *exact* solution of the analysis equation.

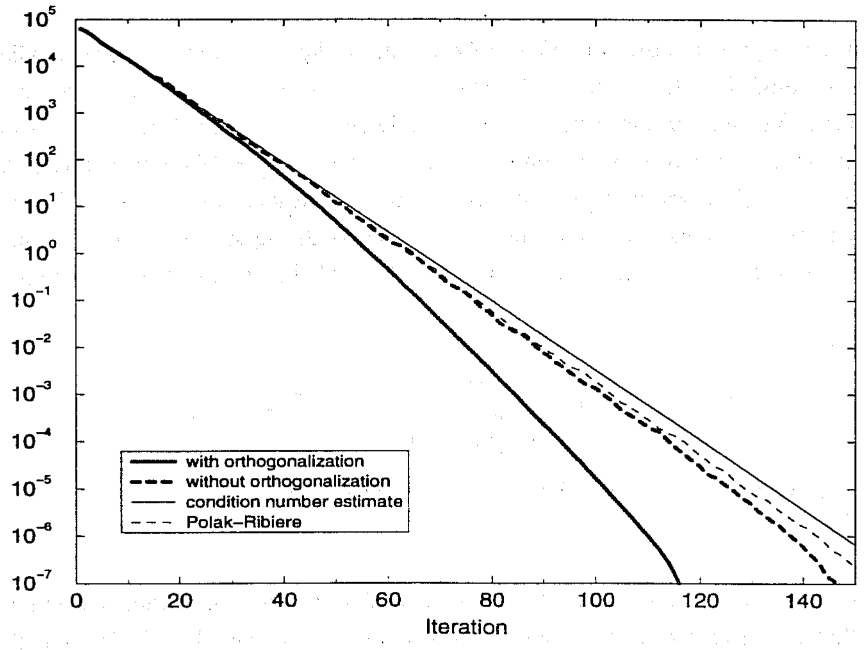

Figure 1: Convergence of conjugate gradients as a function of iteration for a 4dVar cost function. The abscissa is the square of the Hessian norm of the difference between the control vector at a given iteration and its value at the 118th iteration of the orthogonalized algorithm. The thick dashed curve is for the standard conjugate gradient algorithm. The thick solid curve is for the version of the algorithm in which each gradient is explicitly orthogonalized against its predecessors. The thin dashed curve uses the Polak-Ribiere method to determine the descent direction. The thin solid line is the upper bound on the convergence rate defined by equation 12.

Nearly Quadratic Cost Functions

Most minimization algorithms for well-behaved functions determine the descent direction at each iteration by fitting a quadratic model to the function. This approach is clearly appropriate in the case of variational data assimilation, for which a quadratic model is likely to be a rather accurate approximation of the cost function.

Given the success of the conjugate gradient method in the strictly-quadratic case, it is tempting to adapt it to the nearly-quadratic case. There are two problems to be addressed. First, it is no longer possible to determine the exact line minimum at each iteration with a single gradient calculation. Instead, it is necessary to search along the descent direction to find a point for which the function value is sufficiently reduced. A disadvantage of the conjugate gradient method is that, at the start of each iteration, it provides no indication of a likely step length. It may be necessary to perform several gradient calculations in order to determine a suitable step.

The second problem in applying the conjugate gradient algorithm to non-quadratic functions is that the algorithm has a tendency to stagnate. That is, after a few iterations the algorithm tends to produce rather poor search directions. Stagnation may be avoided by introducing a component of the steepest descent direction into the search direction. For example, the algorithm may be restarted every few iterations, or the Polak-Ribiere formula (equation 24) may be used to determine the descent direction. Navon and Legeler (1987) discuss other methods for preventing stagnation, and evaluate the conjugate gradient method for two meteorological applications.

Newton methods provide a second class of algorithms based on a quadratic model. These methods approximate the function to be minimized using a second-order truncated Taylor expansion in the vicinity of the most recent iterate:

$$J(\mathrm{x}) \approx J(\mathrm{x}^{(k)}) + \langle \nabla J(\mathrm{x}^{(k)}), (\mathrm{x} - \mathrm{x}^{(k)}) \rangle + \frac{1}{2}\langle (\mathrm{x} - \mathrm{x}^{(k)}), [J''(\mathrm{x}^{(k)})](\mathrm{x} - \mathrm{x}^{(k)}) \rangle \tag{25}$$

Applying the gradient operator gives

$$\nabla J(\mathrm{x}) \approx \nabla J(\mathrm{x}^{(k)}) + [J''(\mathrm{x}^{(k)})](\mathrm{x} - \mathrm{x}^{(k)}) \tag{26}$$

Since the $\nabla J(\mathrm{x})$ is zero at the minimum of the function, we may set the left hand side to zero and solve for x to determine the approximate location of the minimum.

This simple Newton method suffers from three shortcomings. First, it requires knowledge of the Hessian matrix. For large problems, the Hessian is too large and too computationally expensive to determine explicitly. The second shortcoming is that convergence is not guaranteed for all starting points $\mathrm{x}^{(0)}$. Moreover, the basin of convergence is extremely complicated, making it difficult to determine whether a given starting point will lead to convergence. Finally, even if the algorithm converges to a point with zero gradient, this point is not necessarily a minimum.

Quasi-Newton methods address the problems of the simple Newton method by replacing the Hessian matrix in equation 26 by a positive definite approximation to the Hessian matrix. The approximation is updated at each iteration. By requiring the matrix to be positive definite, quasi-Newton methods guarantee that the direction $(\mathrm{x} - \mathrm{x}^{(k)})$ is a descent direction. This means that there is always a point $\mathrm{x}^{(k)} + \alpha(\mathrm{x} - \mathrm{x}^{(k)})$ with $\alpha > 0$ for which the function is reduced. The requirement for the function to decrease at each iteration, together with some restrictions on the step size α are sufficient to guarantee convergence of the algorithm from any starting point.

The seminal paper on quasi-Newton methods is that of Dennis and Moré (1977). It provides a comprehensive review of the method, together with several theorems on convergence and a thorough discussion of the main approaches to approximating the Hessian matrix. The most popular implementations of the quasi-Newton method are the L-BFGS (Liu and Nocedal, 1989) and M1QN3 (Gilbert and Lemaréchal, 1989) algorithms. The latter is used in the current ECMWF analysis system. The two implementations are nearly identical, even at the code level (Navon, personal communication). Both are based on the limited memory version of the BFGS update formula due to Nocedal (1980).

Truncated Newton algorithms are less well known and less well developed than quasi-Newton algorithms. The essence of the method is to approximate the function explicitly at each iteration by the quadratic function

$$\begin{aligned}\phi(\mathrm{x}) &= J(\mathrm{x}^{(k)}) + \langle \nabla J(\mathrm{x}^{(k)}), (\mathrm{x} - \mathrm{x}^{(k)}) \rangle \\ &\quad + \frac{1}{2}\langle (\mathrm{x} - \mathrm{x}^{(k)}), [J''(\mathrm{x}^{(k)})](\mathrm{x} - \mathrm{x}^{(k)}) \rangle\end{aligned} \tag{27}$$

$\phi(\mathrm{x})$ may be minimized using, for example, conjugate gradients to give an estimate of the location of the minimum of $J(\mathrm{x})$. Since the function $\phi(\mathrm{x})$ is only an approximation to $J(\mathrm{x})$, there is no point in minimizing it fully. Therefore, the minimization of $\phi(\mathrm{x})$ is "truncated" by stopping before full convergence is achieved. The advantage of the truncated Newton approach is that the algorithm used to minimize $\phi(\mathrm{x})$ does not need an explicit representation of the full Hessian matrix. Rather, it needs to be able to calculate Hessian-vector products. One approach is to do this approximately using finite-differences:

$$J''\mathrm{d} = \frac{1}{\varepsilon}(\nabla J(\mathrm{x}^{(k)} - \varepsilon\mathrm{d}) - \nabla J(\mathrm{x}^{(k)})) \tag{28}$$

Other possibilities exist. For example, the second-order adjoint equation (Wang *et al.*, 1992) allows exact Hessian-vector products to be calculated, and has been used to good effect by Wang *et al.* (1995).

Truncated Newton methods frequently display impressive convergence rates when expressed as error reduction per iteration. However, each iteration is computationally expensive, since it requires the evaluation of several Hessian-vector products. As a result, truncated Newton methods are not, in general, markedly better than quasi-Newton methods. Nash and Nocedal (1989) compared the quasi-Newton L-BFGS algorithm (Liu and Nocedal, 1989) with the truncated Newton TN algorithm (Nash, 1984) for a set of 45 standard test problems. They concluded that neither algorithm is clearly superior, although both are superior to non-linear conjugate gradients for problems in which the cost of a function evaluation is large. For several of the test problems, the L-BFGS algorithm required fewer function-gradient evaluations. However, the TN algorithm was somewhat better than L-BFGS for quadratic and nearly-quadratic problems.

The truncated Newton algorithm is particularly well suited for use with trust-region methods. These methods define a region around $\mathrm{x}^{(k)}$ within which $\phi(\mathrm{x})$ is "trusted". That is, within the region $\phi(\mathrm{x})$ is a good approximation to $J(\mathrm{x})$. Steihaug (1983) showed that if the conjugate gradient algorithm is used to minimize $\phi(\mathrm{x})$, then convergence to the minimum of $\phi(\mathrm{x})$ within the trust region is guaranteed provided the minimization is stopped as soon as a direction of negative curvature is detected, or when the iteration steps outside the trust region. In both cases, the minimum is located on the boundary and is given by the intersection of the boundary of the trust region with the last search direction. The ability of trust region methods to follow directions of negative curvature is a particular advantage over methods based on positive definite approximations to the Hessian. The close connection between the Lanczos algorithm and conjugate gradients allows directions of negative curvature to be detected easily, since they correspond to negative eigenvalues of the tridiagonal matrix $\mathrm{T}^{(k)}$ (equation 20).

The truncated Newton algorithm has something of the flavour of incremental data assimilation. Like incremental assimilation, it consists of a pair of nested iterations in which the outer iteration minimizes a non-quadratic function, and the inner iteration minimizes a quadratic function. Since the incremental method gives a large reduction in computational cost, a truncated Newton method based on the incremental method is an attractive idea.

Recently, Li and Navon (1999) added a line search and a BFGS Hessian approximation to the outer loop of the incremental method. They demonstrate that their algorithm allows them to efficiently minimize a 4dVar cost function which includes physical parameterizations in the model, at a computational cost which is only 35% greater than that required for a cost function from which physical parameterizations are excluded. The computational efficiency of Li and Navon's algorithm, and of incremental assimilation, results from the minimization of a relatively inexpensive cost function during the inner iterations.

The main problem with the incremental method is that there is no guarantee that the outer iterations converge. In Li and Navon's algorithm (*op. cit*), this translates into the possibility that the search direction provided by the inner iteration may not be a descent direction. In practice, this possibility did not occur in Li and Navon's experiments. Nevertheless, the possibility of failure of the minimization is a serious shortcoming.

Let us consider a nested minimization algorithm, and denote the function minimized during the outer iterations by $J(\mathrm{x})$, and the function minimized during the inner iterations by $\hat{J}(\hat{\mathrm{x}})$, where $\hat{\mathrm{x}}$ is related to x through an affine transformation

$$\hat{\mathrm{x}} = \mathrm{S}_k \mathrm{x} + \hat{\mathrm{c}}_k \tag{29}$$

The inner iteration at step k of the outer loop consists of a partial minimization of $\hat{J}(\hat{\mathrm{x}})$ starting from the point $\hat{\mathrm{x}}^{(k)} = \mathrm{S}_k \mathrm{x}^{(k)} + \hat{\mathrm{c}}_k$. The result of this minimization is a point $\hat{\mathrm{x}}$ for which $\hat{J}(\hat{\mathrm{x}}) < \hat{J}(\hat{\mathrm{x}}^{(k)})$. In the incremental method, the point $\hat{\mathrm{x}}$ defines the new iterate at the outer level:

$$\mathrm{x}^{(k+1)} = \mathrm{x}^{(k)} + \mathrm{S}_k^{+}(\hat{\mathrm{x}} - \hat{\mathrm{x}}^{(k)}) \tag{30}$$

where S_k^{+} is the pseudo-inverse of S_k. In Li and Navon's method, $\mathrm{S}_k^{+}(\hat{\mathrm{x}} - \hat{\mathrm{x}}^{(k)})$ defines the initial search direction for the outer iteration. (They follow this initial search by a few steps of quasi-Newton minimization using a BFGS approximation to the Hessian matrix.)

For the direction $\mathrm{S}_k^{+}(\hat{\mathrm{x}} - \hat{\mathrm{x}}^{(k)})$ to be a descent direction, we require $\langle \nabla J(\mathrm{x}^{(k)}), \mathrm{S}_k^{+}(\hat{\mathrm{x}} - \hat{\mathrm{x}}^{(k)}) \rangle < 0$. In general, this cannot be guaranteed, since the inner iteration has no knowledge of the gradient $\nabla J(\mathrm{x}^{(k)})$ of the outer-level cost function. Fortunately, a minor modification to the inner cost function is sufficient to give a descent direction. Consider the following function

$$h_k(\hat{\mathrm{x}}) = \hat{J}(\hat{\mathrm{x}}) + \langle [(\mathrm{S}_k^{+})^{*} \nabla J(\mathrm{x}^{(k)}) - \nabla \hat{J}(\hat{\mathrm{x}}^{(k)})], (\hat{\mathrm{x}} - \hat{\mathrm{x}}^{(k)}) \rangle \tag{31}$$

where $(\mathrm{S}_k^{+})^{*}$ is the adjoint of S_k^{+}.

Note that the gradients which appear in the inner product on the right hand side of equation 31 are evaluated at the starting point, and are fixed throughout the inner iteration. The cost of a function or gradient evaluation for $h_k(\hat{\mathrm{x}})$ is therefore similar to that of $\hat{J}(\hat{\mathrm{x}})$. However, the need to calculate a gradient of $J(\mathrm{x})$ increases the cost of the outer iterations when compared with the standard incremental method.

Suppose we perform a few steps of minimization to find a point $\hat{\mathrm{x}}$ for which $h_k(\hat{\mathrm{x}}) < h_k(\hat{\mathrm{x}}^{(k)})$.

That is,

$$\hat{J}(\hat{x}) + \langle [(S_k^+)^* \nabla J(x^{(k)}) - \nabla \hat{J}(\hat{x}^{(k)})], (\hat{x} - \hat{x}^{(k)}) \rangle < h_k(\hat{x}^{(k)}) \tag{32}$$

Rearranging, and noting that $h_k(\hat{x}^{(k)}) = \hat{J}(\hat{x}^{(k)})$ gives

$$\langle [(S_k^+)^* \nabla J(x^{(k)})], (\hat{x} - \hat{x}^{(k)}) \rangle < \hat{J}(\hat{x}^{(k)}) + \langle [\nabla \hat{J}(\hat{x}^{(k)})], (\hat{x} - \hat{x}^{(k)}) \rangle - \hat{J}(\hat{x}) \tag{33}$$

The left hand side may be rearranged to give $\langle \nabla J(x^{(k)}), S_k^+(\hat{x} - \hat{x}^{(k)}) \rangle$. Thus $S_k^+(\hat{x} - \hat{x}^{(k)})$ is a descent direction if the right hand side of equation 33 is negative. Now, the first two terms on the right hand side of equation 33 define the tangent plane to $\hat{J}(\hat{x})$, which touches at $\hat{x}^{(k)}$. Consequently, the right hand side of equation 33 is negative by definition if $\hat{J}(\hat{x})$ is a convex function.

The preceding argument shows that any convex function $\hat{J}(\hat{x})$ may be used during the inner iterations to provide a descent direction for the outer iteration. However, this is not sufficient to ensure efficient minimization of the cost function. For this, the direction $S_k^+(\hat{x} - \hat{x}^{(k)})$ should be close to the direction implied by the Newton equation.

Taking the gradient of equation 31, gives

$$\nabla h_k(\hat{x}) = \nabla \hat{J}(\hat{x}) - \nabla \hat{J}(\hat{x}^{(k)}) + (S_k^+)^* \nabla J(x^{(k)}) \tag{34}$$

If $\hat{J}(\hat{x})$ is approximately quadratic, then the first term on the right hand side is approximately $\nabla \hat{J}(\hat{x}^{(k)}) + [\hat{J}''(\hat{x}^{(k)})](\hat{x} - \hat{x}^{(k)})$. Furthermore, if sufficient inner iterations have been performed, then the left hand side will be small. Applying these approximations gives

$$[\hat{J}''(\hat{x}^{(k)})](\hat{x} - \hat{x}^{(k)}) + (S_k^+)^* \nabla J(x^{(k)}) \approx 0 \tag{35}$$

Multiplying to the left by $S_k^+[\hat{J}''(\hat{x}^{(k)})]^{-1}$ gives

$$S_k^+(\hat{x} - \hat{x}^{(k)}) \approx -(S_k^+[\hat{J}''(\hat{x}^{(k)})]^{-1}(S_k^+)^*)\nabla J(x^{(k)}) \tag{36}$$

Thus, for the direction $S_k^+(\hat{x} - \hat{x}^{(k)})$ to be close to the solution of the Newton equation, we require that $[J''(x^{(k)})]^{-1}$ is well approximated by $S_k^+[\hat{J}''(\hat{x}^{(k)})]^{-1}(S_k^+)^*$.

The main result of this section is that the convergence of incremental assimilation may be guaranteed by replacing the simplified cost function minimized during the inner iterations by the function $h_k(\hat{x})$, and by performing a line search along the direction of the increments during the outer iteration. The algorithm is as follows:

The Generalized Truncated Newton Algorithm

Choose a starting point, $x^{(0)}$. Then, for k=0,1,2,... repeat the following steps:

1. Calculate $\nabla J(x^{(k)})$
2. Apply the simplification operator to $x^{(k)}$ to determine $\hat{x}^{(k)}$.
3. Calculate $\nabla \hat{J}(\hat{x}^{(k)})$.
4. Define the function $h_k(\hat{x}) = \hat{J}(\hat{x}) + \langle [(S_k^+)^* \nabla J(x^{(k)}) - \nabla \hat{J}(\hat{x}^{(k)})], (\hat{x} - \hat{x}^{(k)}) \rangle$.
5. Apply a few steps of a minimization algorithm to $h_k(\hat{x})$ to find a point $\hat{x}$ for which $h_k(\hat{x}) < h_k(\hat{x}^{(k)})$.
6. Perform a line search to find $\alpha > 0$ for which $J[x^{(k)} + \alpha S_k^+(\hat{x} - \hat{x}^{(k)})] < J(x^{(k)})$. Such a point must exist if $\hat{J}(\hat{x})$ is convex.
7. Define the next iterate as $x^{(k+1)} = x^{(k)} + \alpha S_k^+(\hat{x} - \hat{x}^{(k)})$.

The algorithm makes no assumptions about the minimization algorithm used for the inner iterations, or about the form of the operator S_k. This allows considerable flexibility. For example, a version of the multi-incremental formulation (Veersé and Thépaut, 1998) may be produced by using the generalized truncated Newton algorithm itself as the minimization algorithm for the inner iterations. The resulting recursive algorithm has much in common with multigrid methods.

The functions $h_k(\hat{x})$ minimized during the inner iterations are likely to have very similar (possibly identical) Hessian matrices. Information about the Hessian matrix of $h_k(\hat{x})$, which is generated during its minimization, may be saved and used to precondition subsequent minimizations. A convenient way to achieve this is to use the limited-memory BFGS update formula (Nocedal, 1980).

If the functions minimized during the inner iterations are strictly quadratic, then conjugate gradient minimization with explicit orthogonalization of the gradient vectors may be particularly efficient. However, the standard conjugate gradient algorithm assumes that the preconditioner is fixed throughout the minimization. This makes it difficult to use a limited-memory preconditioner. Axelsson (1994, algorithm 5a) presents a generalized conjugate gradient algorithm which may be used in this case. The algorithm includes explicit orthogonalization of the gradient vectors, so preserves super-linear convergence.

The generalized truncated Newton algorithm presented above is capable of demonstrating rapid convergence rates. However, the convergence is linear. To achieve super-linear convergence, it is necessary to make the approximate inverse Hessian matrix, which appears in equation 36, converge to $[J''(x)]^{-1}$ as the minimization proceeds. This may be achieved by incorporating information about $J''(x)$ generated during earlier iterations into the definition of S_k. The form of equation 36 suggests the product form of the BFGS or DFP updates could be used (Brodlie, Gourlay and Greenstadt, 1973). Alternatively, note that if $S_k^+(\hat{x} - \hat{x}^{(k)})$ is a descent direction, then so too is $S_k^+(\hat{x} - \hat{x}^{(k)}) - A_k \nabla J(x^{(k)})$ where A_k is any positive definite matrix. Nocedal (1980) notes that the standard form of the BFGS approximation to the inverse of the Hessian is of the form

$$H_k = V_k H_o V_k^* + A_k \tag{37}$$

where H_0 is an initial approximation to the inverse Hessian, and where A_k is positive definite.

Thus, if we define

$$S_k^+ = V_k S^+ \tag{38}$$

for some fixed simplification operator S, then equation 36 shows that the direction $S_k^+(\hat{x} - \hat{x}^{(k)}) - A_k \nabla J(x^{(k)})$ is approximately that which would be generated by a quasi-Newton algorithm using the BFGS approximation to the Hessian with $H_0 = S^+[\hat{J}''(\hat{x})](S^+)^*$. (Note that it is not necessary to form the matrices V_k and A_k explicitly. The recursive formula (Nocedal, 1980) may be used.)

Some Results

The generalized truncated Newton algorithm has been tested in the ECMWF 4dVar analysis system. So far, it has not been used to replace the incremental algorithm, since this would require the calculation of a gradient at high resolution, which is technically difficult. Instead, the algorithm has been used in place of the quasi-Newton minimization algorithm used in the inner iterations of the incremental algorithm. For the test, the cost function included all operationally-used observations, as well as the physical parameterizations of vertical diffusion, sub-grid scale orographic effects, large-scale precipitation, deep moist convection and longwave radiation (Mahfouf and Rabier, 1999). The incremental resolution was T63 with 31 levels.

The simplified cost function, $\hat{J}(\hat{x})$ minimized in the inner loops of the generalized truncated Newton algorithm was defined by removing the physical parameterizations and the balance constraint (J_c in equation 1). The computational cost of a gradient calculation for the simplified cost function is between 2 and 3 times less than the cost of a gradient calculation when the physical parameterizations and balance constraint are included. Note that the spatial resolution of the simplified cost function was the same as that of the cost function for the outer iterations. Consequently, the simplification operator, S, was simply the identity matrix.

The cost function for the inner incremental iterations is not strictly quadratic due to the inclusion of variational quality control (Andersson, 1997) and a non-quadratic cost function for scatterometer observations (Stoffelen and Andersson, 1997). For the simplified cost function, variational quality control was disabled, and a quadratic cost function was used for scatterometer observations. The simplified cost function was therefore quadratic.

The inner iterations of the generalized truncated Newton algorithm used a generalized conjugate gradient algorithm (Axelsson, 1994, algorithm 5a) together with a limited-memory BFGS preconditioner. The memory for this preconditioner was 200 pairs of vectors. No experimentation was performed to investigate the impact of the memory size on the efficiency of the minimization. Each inner iteration consisted of a fixed number of iterations of the generalized conjugate gradient algorithm.

The outer iterations of the generalized truncated Newton algorithm used the standard form of the limited memory BFGS approximation. The memory was 10 pairs of vectors. Again, no investigation of the impact of the memory size on the efficiency of the minimization was carried out. The line search algorithm was identical to that used in M1QN3.

Figure 2 shows the convergence of the generalized truncated Newton algorithm as a function of the number of outer iterations performed. Also shown is the convergence of the M1QN3 quasi-Newton algorithm. As is typical of truncated

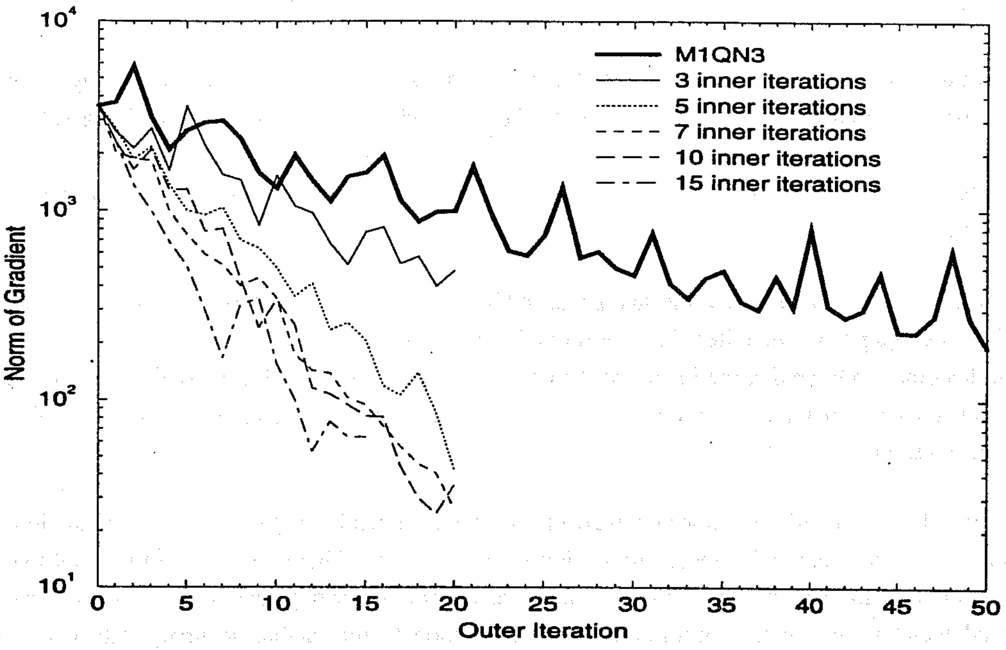

Figure 2: Convergence of the Generalized truncated Newton algorithm as a function of outer iteration.

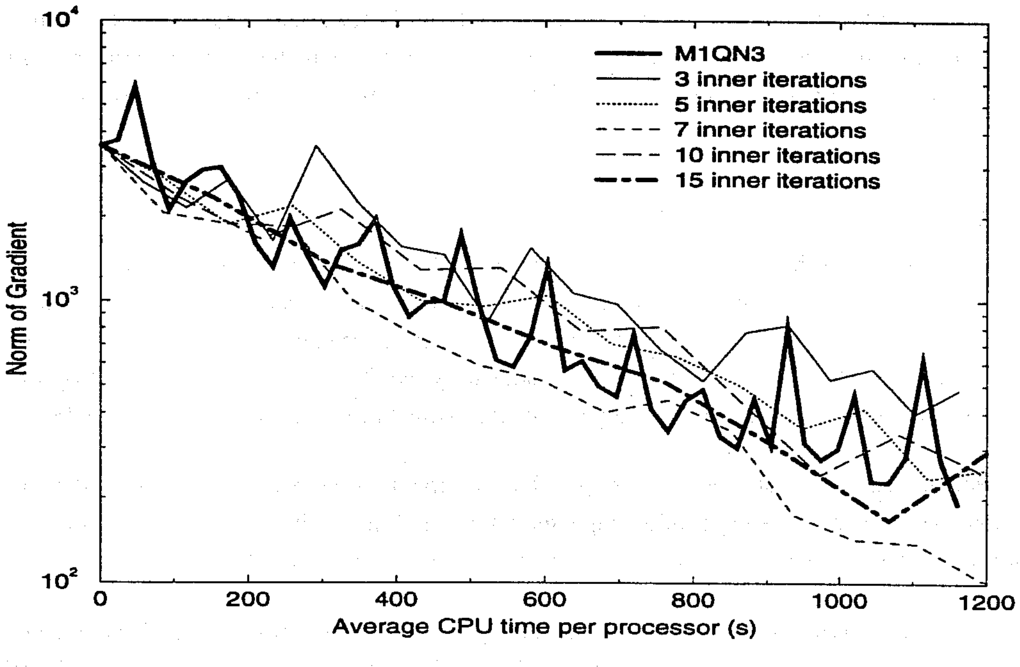

Figure 3: Convergence of the Generalized truncated Newton algorithm as a function of CPU time.

Newton algorithms, the convergence per outer iteration is rapid. The convergence rate increases monotonically with the number of inner iterations.

Figure 3 shows the convergence rate as a function of the average CPU time when run on 16 processors of a Fujitsu VPP700 computer. By this measure, the performance of the generalized truncated Newton algorithm is much more similar to that of the quasi-Newton algorithm. With just three inner iterations, the convergence rate of the truncated Newton algorithm is marginally slower than the quasi-Newton algorithm, With 7 inner iterations, the convergence rate is marginally faster.

With 7 or fewer inner iterations, the line search algorithm always accepted the initial point provided by the inner iterations. However, with 10 inner iterations, there were 3 outer iterations for which the cost function at the initial point was larger than that at the end of the previous iteration. In these cases, a line search was required to determine a suitable point. With 15 inner iterations, most outer iterations required a line search. Each line search found a suitable point after just one additional gradient evaluation.

Steihaug (1983) showed that the length of the step produced by conjugate gradient minimization increases monotonically with the number of conjugate gradient iterations performed. It is likely that, as the number of inner iterations is increased, the step length becomes so large that the inner problem can no longer be regarded as a good approximation to the outer problem. If this is the case, then a trust-region approach could provide a means to automatically determine the optimum number of inner iterations.

The performance of the generalized truncated Newton algorithm depends on the relative computational cost of the functions minimized in the outer and inner iterations. For the tests presented here, the ratio of these costs was around 2.5. In future, it is likely that this ratio will increase as parameterizations of more physical processes are included. The ability of the truncated Newton algorithm to minimize the cost function to reasonable accuracy with between 10 and 15 evaluations of the full cost function, rather than the 50 required by the quasi-Newton algorithm, makes the truncated Newton algorithm increasingly attractive as the cost of the physical parameterizations increases.

There is considerable scope for reducing the cost of the inner iterations. For example, a 3dVar cost function could be used, or the number of observations could be reduced. If the effect of these changes on the Hessian matrix is small, they would not adversely affect the convergence rate per outer iteration, but would significantly reduce the computational cost of the algorithm.

Conclusions

Minimization algorithms are at the heart of variational data assimilation, and the efficiency of the algorithm used has a direct impact on the computational cost of the analysis. For this reason, developments in the field of numerical optimization are likely to have important consequences for numerical weather prediction. Of particular interest are developments in truncated Newton algorithms since, as shown by Nash and Nocedal (1989) and by the generalized truncated Newton algorithm presented here, such algorithms are particularly well suited to nearly quadratic problems.

Real improvements in computational efficiency are possible by exploiting the particular characteristics of the analysis problem. An example is given in this paper of a simple modification of the conjugate gradient algorithm which reduces the number of iterations of conjugate gradients required to minimize a 4dVar cost function by about 20%.

The connection between the Lanczos algorithm and conjugate gradients provides several promising directions for future research. The ability to determine eigenvectors and eigenvalues of the Hessian matrix, and thus of the covariance matrix of analysis error, is already exploited in the ECMWF system to estimate variances of analysis error. An example of a further application of this connection, is provided by the algorithm of Golub and Meurant (1994), who show how the Lanczos algorithm may be used to determine upper and lower bounds on the quadratic form $u^T f(A) u$, where f is a well behaved function of a symmetric positive definite matrix A. Applying their algorithm to the Hessian matrix allows, for example, good estimates to be calculated for the mean variance of analysis error ($f(J'') = (J'')^{-1}$, and **u** a vector whose elements take the values ± 1 at random), or the determinant of the analysis error covariance matrix

($f(J'') = \log[(J'')^{-1}]$). In addition, Golub and von Matt (1996) show how the algorithm may be exploited to calculate the generalized cross-validation function (Golub, Heath and Wahba, 1979).

Preconditioning of the minimization in variational data assimilation is an area which has yet to receive the attention it deserves. There is the potential for very large reductions in computational cost. It is amusing that one of the best ways to precondition variational data assimilation may be to use the domain decomposition techniques which were developed for use in OI.

References

Andersson E., 1997, Implementation of variational quality control, Proc. ECMWF workshop on non-linear aspects of data assimilation, Reading 9-11 September, 1996.

Axelsson O., 1994. Iterative Solution Methods, Cambridge University Press.

Barrett R., M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst 1994, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition, Pub. SIAM, Philadelphia. http://www.netlib.org/templates/Templates.html

Brodlie K.W., A.R. Gourlay and J. Greenstadt, 1973, Rank-one and Rank-two Corrections to Positive Definite Matrices Expressed in Product Form. J. Inst. Maths Applics, 11, pp 73-82.

Courtier P., J-N. Thépaut, A. Hollingsworth 1994, A strategy for operational implementation of 4D-Var, using an incremental approach. Quart. J. Roy. Meteor. Soc. 120, pp 1367-1388.

Daley R., 1991, Atmospheric Data Analysis, Cambridge Atmospheric and Space Science Series, Cambridge University Press.

Dennis J.E. Jr. and J.J. Moré, 1977, Quasi-Newton Methods, Motivation and Theory. SIAM Review, Vol. 19, No. 1, pp 46-89.

Fisher M. and P. Courtier 1995, Estimating the Covariance Matrices of Analysis and Forecast Error in Variational Data Assimilation, ECMWF Research Dept. Tech. Memo 220.

Gilbert J.C. and C. Lemaréchal, 1989, Some numerical experiments with variable storage quasi-Newton algorithms, Math. Prog 45, pp 407-436.

Greenbaum, A. 1989, Behaviour of slightly perturbed Lanczos and conjugate-gradient recurrences, Linear Algebra Appl., 113, pp 7-63.

Golub G.H., M. Heath and G. Wahba, 1979, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics, 21, pp 215-223.

Golub G.H. and U. von Matt, 1996, Generalized cross-validation for large scale problems, Technical report SCCM-96-08, Stanford University, http://www-sccm.stanford.edu/reports.html

Golub G.H. and G. Meurant, 1994, Matrices, moments and quadrature, Technical report SCM-93-07, Stanford University, http://www-sccm.stanford.edu/reports.html

Greenbaum A. and Z. Strakos, 1992, Predicting the behaviour of finite precision Lanczos and conjugate gradient computations, SIAM J. Matrix An. Appl., 13, pp 121-137.

Hollingsworth A., 1987, Objective analysis for numerical weather prediction, Special volume J. Meteor. Soc. Japan "Short and medium range numerical weather prediction", Matsuno T. ed., pp 11-60.

Klinker E., F. Rabier, G. Kelly and J-F. Mahfouf, 1999, The ECMWF operational implementation of four dimensional variational data assimilation Part III: Experimental results and diagnostics with operational configuration (submitted to Quart. J. Royal Meteor. Soc.).

Li Z. and I.M. Navon, 1999, Performance of 4D-Var strategies using the FSU global spectral model with its full physics adjoint, submitted to Monthly Weather Review.

Liu D.C. and J. Nocedal, 1989, On the limited memory BFGS method for large scale optimization, Mathematical Programming, 45, pp 503-528.

Mahfouf J-F. and F. Rabier, 1999, The ECMWF operational implementation of four dimensional variational data assimilation Part II: Experimental results with improved physics. ECMWF Research Dept. Tech. Memo 272 (submitted to Quart. J. Royal Meteor. Soc.).

Nash S.G. 1984, User's guide for TN/TNBC: Fortran routines for nonlinear optimization, Report 397, Mathematical Sciences Dept., The John Hopkins University.

Nash S.G. and J. Nocedal, 1989, A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization, Technical Report NAM 02, Dept. Electrical Engineering and Computer Science, Northwestern University.

Nash S.G. and A. Sofer, 1989, Block truncated-Newton methods for parallel optimization, Mathematical Programming, 45, pp 529-546.

Navon I.M. and D.M. Legeler, 1987, Conjugate-gradient methods for large-scale minimization in meteorology, Monthly Weather Review, 115, pp 1479-1502.

Nocedal J., 1980, Updating quasi-Newton matrices with limited storage, Mathematics of Computation, 35, No. 151, pp 773-782.

Notay Y., 1993, On the convergence rate of the conjugate gradients in presence of rounding errors, Numerische Mathematik, 65, pp 301-317.

O'Leary D.P., 1980, The block conjugate-gradient algorithm and related methods, Linear Algebra Applics., 29, pp 20-33.

O'Leary D.P., 1987, Parallel implementation of the block conjugate gradient algorithm, Parallel Computing, 5, pp 127-139.

Paige, C.C., 1971, The computation of eigenvalues and eigenvectors of very large sparse matrices, Ph.D. Thesis, University of London.

Paige C.C. and M.A. Saunders, 1975, Solution of sparse indefinite systems of linear equations. SIAM J. Numerical Analysis, 12, pp 617-629.

Parlett B.N. and D.S. Scott, 1979,The Lanczos algorithm with selective orthogonalization, Mathematics of Computation, 33, No. 145, pp 217-238.

Polak E. and G. Ribiere, 1969, Note sur la convergence de methodes de directions conjugueés, Rev. Franç. Informat. Rech. Operationnelle, 16, 35-43.

Rabier F. 1999, The ECMWF operational implementation of four dimensional variational data assimilation Part I: Experimental results with simplified physics. ECMWF Research Dept. Tech. Memo 271 (submitted to Quart. J. Royal Meteor. Soc.).

Simon H.D., 1984, The Lanczos algorithm with partial reorthogonalization, Mathematics of Computation, 42, No. 165, pp 115-142

Sherman J. and W.J. Morrison, 1949, Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix, Ann. Math. Statist. 20, pp 621.

Shewchuk J.R. 1994, An introduction to the conjugate gradient method without the agonizing pain. Edition $1\frac{1}{4}$. http://www.cs.cmu.edu/~jrs/

Steihaug T., 1983, The conjugate gradient method and trust regions in large scale optimization, SIAM J. Num. Anal., 20, pp 626-637.

Van der Sluis A. and H.A. van der Vorst, 1986, The rate of convergence of conjugate gradients, Numerische Mathematik, 48, pp 543-560

Steihaug T. 1983, The conjugate gradient method and trust regions in large scale optimization. SIAM J. Numer. Anal., 20, pp 626-637.

Stoffelen A. and E. Andersson, 1997, Ambiguity removal and assimilation of scatterometer data. Quart. J. Royal Meteor. Soc. 123, pp 491-518.

Strakos Z., 1992, On the real convergence rate of the conjugate gradient method, Linear Algebra Appl., 154, p 535-549.

Thépaut J-F. and P. Moll, 1990, Variational inversion of simulated TOVS radiances using the adjoint technique, Quart. J. Royal Meteor. Soc. 116, pp 1425-1448.

Veersé F. and J-N. Thépaut, 1998, Multiple-truncation incremental approach for four-dimensional variational data assimilation, Quart. J. Royal Meteor. Soc., 124, pp 1889-1908.

Wang Z., I. Navon, F.X. Le Dimet and X. Zou, 1992, The second order adjoint analysis: theory and applications. Meteorology and Atmospheric Physics, 1992.

Wang Z., I. Navon and X. Zou and F.X. LeDimet, 1995, A truncated Newton optimization algorithm in meteorology applications with analytic Hessian/vector products, Computational Optimization and Applications, 4, pp 241-262.