ELSEVIER

Applied Numerical Mathematics 19 (1995) 17-31

APPLIED NUMERICAL MATHEMATICS

The implementation of a Generalized Cross Validation algorithm using deflation techniques for linear systems

K. Burrage [a,*], A. Williams [a], J. Erhel [b], B. Pohl [c]

[a] *Department of Mathematics, University of Queensland, Brisbane, 4072, Australia*

[b] *INRIA, Campus de Beaulieu, 35042 Rennes, France*

[c] *Seminar für Angewandte Mathematik, ETH Zürich, 8092 Zürich, Switzerland*

Abstract

The fitting of a thin plate smoothing spline to noisy data using the method of minimizing the Generalized Cross Validation (GCV) function is computationally intensive involving the repeated solution of sets of linear systems of equations as part of a minimization routine. In the case of a data set of more than a few hundred points, implementation on a workstation can become unrealistic and it is then desirable to exploit high performance computing. The usual implementation of the GCV algorithm performs Householder reductions to tridiagonalize the influence matrix and then solves a sequence of tridiagonal linear systems which are updated only by a scalar value (the minimization parameter) on the diagonal. However, this approach is not readily parallelizable. In this paper the deflation techniques described by Burrage et al. (1994), which are used to accelerate the convergence of iterative schemes applied to linear systems, will be adapted to the problem of minimizing the GCV function. This approach will allow vector and parallel architectures to be exploited in an efficient manner.

Keywords: Linear systems; Deflation; Iterative techniques; Generalized Cross Validation algorithms

1. Introduction

As a consequence of the recent El Nino which has resulted in severe drought over much of Eastern Australia for 1991-1993, with a concomitant loss of at least one billion dollars in agricultural production, a collaborative arrangement between the Queensland Department of Primary Industries and CIAMP (Centre for Industrial and Applied Mathematics and Parallel Computing) at the University of Queensland has been formed. As a result of this collaboration, an interactive visualization environment, ADVISE [11], has been developed to enable the monitoring and modelling of drought conditions over Australia. This is an enormously complicated and computationally intensive problem, and so

* Corresponding author. E-mail: kb@maths.uq.oz.au.

0168-9274/95/$09.50
SSDI 0168-9274(95)00016-X

ADVISE is designed to run in a heterogeneous computing environment. Within ADVISE a number of applications are run including a simulation application and a surface fitting program, and these are now briefly described.

In the simulation application, a number of environmental variables (such as edible green matter in pastures) are calculated at every point on a grid over the state of Queensland. These variables are then used to produce a colour-coded map of the state, so that the overall effects of drought, and the condition of farming and grazing land, can be visualized. These pasture model calculations need to make use of weather data, such as rainfall and temperature readings, at each point on the grid. The sizes used for the grid spacings range from 2.5 km to 10 km, depending on the resolution of the image being produced. Since weather data is not available at a resolution this fine, an estimate must be obtained by fitting a spline surface to the data from irregularly spaced weather stations, of which there are approximately 3000 in the state of Queensland and 16000 throughout Australia.

Since the data is known to contain noise, a smoothing spline is constructed within a Generalized Cross Validation approach, of which there are a number of implementations including those by Gu et al. [6] and Hutchinson [8]. The method seeks to fit a smooth surface to n data points modelled by

$$y_j = L_j f + \varepsilon_j,$$

where $f \in H$ (a Hilbert space), L_j are bounded linear functionals and ε_j are errors or noise in the data. Thus it is desired to find f_λ that minimizes the variational problem

$$\frac{1}{n}\sum_{j=1}^{n}(y_j - L_j f_\lambda)^2 + \lambda J_m(f_\lambda), \tag{1}$$

where $J_m(f)$ is a measure of the "roughness" of the surface in terms of the mth derivatives of f and where λ is the smoothing parameter.

The problem of minimizing (1) is equivalent to minimizing the Generalized Cross Validation function $GCV(\lambda)$ (see [16]), where

$$GCV(\lambda) = \frac{(1/n)\|(I - A(\lambda))y\|^2}{[(1/n)\mathrm{tr}(I - A(\lambda))]^2}. \tag{2}$$

Here $\mathrm{tr}(A)$ represents the trace of a matrix A and $A(\lambda)$ is the $n \times n$ influence matrix satisfying

$$(L_1 f_\lambda, \ldots, L_n f_\lambda)^\top = A(\lambda)y.$$

Let the observed data values (in this case rainfall) have the independent variables latitude, longitude and elevation which are stored in the matrix X $(n \times 3)$, and let the observed rainfall readings (or surface values) be stored in the vector y (an n-vector). Then a matrix S $(n \times 4)$ is formed which contains unity in the first column, and X in the last three columns. A symmetric matrix $\tilde{Q}$ $(n \times n)$ is also formed, with elements given by

$$\tilde{Q} = \begin{cases} \theta \ln(d_{ij}) d_{ij}^{(m/2)}, & m \text{ an even integer}, \\ \theta \sqrt{d_{ij}} d_{ij}^{((m-1)/2)}, & m \text{ an odd integer}, \end{cases} \tag{3}$$

where

$$d_{ij} = \sum_{k=1}^{3} (X_{ik} - X_{jk})^2, \quad \forall i, j = 1, \ldots, n,$$

and where θ is calculated from the derivative order and the number of independent variables. Since

$m = 2$(derivative order) − number of independent variables,

a derivative order of 2 and three independent variables imply $m = 1$. Thus the elements of $\tilde{Q}$ are, from (3), proportional to the Euclidean distance between data points i and j.

Let the QR factorization of S be

$$S = (F_1 \quad F_2) \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Here F_1 and F_2 are, respectively, $n \times 4$ and $n \times (n-4)$, while R is an upper triangular matrix of dimension 4. Then writing $Q = F_2^{\top} \tilde{Q} F_2$ and $z = F_2^{\top} y$ (Q is now an $(n-4) \times (n-4)$ matrix), the GCV function can be written as

$$GCV(\lambda) = \frac{(1/n) z^{\top} (Q + \lambda I)^{-2} z}{[(1/n) \mathrm{tr}(Q + \lambda I)^{-1}]^2}. \tag{4}$$

Here Q is a full symmetric matrix, but at each evaluation only the diagonal elements are updated by the addition of a scalar value λ. For this reason it is possible to first transform Q to tridiagonal form using Householder reductions and then minimize the GCV function much more economically by calculating $Q = UTU^{\top}$ (where U is orthogonal and T is tridiagonal) and $w = U^{\top} z$, so that

$$GCV(\lambda) = \frac{(1/n) w^{\top} (T + \lambda I)^{-2} w}{[(1/n) tr(T + \lambda I)^{-1}]^2}. \tag{5}$$

This function can now be evaluated quite economically using the LU factorization of a tridiagonal matrix, so that the time required to evaluate (5) increases only linearly with problem size. The Householder reduction needed to produce T, however, is an $O(n^3)$ algorithm and for large data sets this is where approximately 90% of execution time is spent. This is also the section of the program that does not readily vectorize or parallelize. Thus in order to exploit vectorization or parallelization protocols, the algorithm needs to be restructured or recast to avoid the Householder reduction. This will be done using iterative techniques which will be rapidly accelerated using the deflation techniques discussed by Burrage et al. [2].

The outline of this paper is as follows. In Section 2 a brief description of the deflation process as applied to iterative methods for linear systems will be given. In Section 3 we will discuss how the GCV application can be restructured in order to exploit these techniques. Numerical results on a Cray YMP-2D sited at the University of Queensland will be presented in Section 4 to illustrate the efficacy of this approach. This paper will conclude in Section 5 with some comments on the suitability of this deflation process for many applications where linear and nonlinear systems need to be solved in either sequential or parallel computing environments. In particular, it will be briefly discussed how this technique can be used to solve systems of ordinary differential equations in an efficient manner

2. Deflation techniques for linear systems

Computational techniques for solving linear systems of the form

$$Ay = b, \quad y \in \mathbb{R}^n \tag{6}$$

can be divided into two broad categories: direct and iterative methods. In the direct case, elementary row operations are performed on the augmented matrix (A, b) in order to reduce the system to a simpler form which can be more easily solved by exploiting the architecture (sequential or parallel) of the target machine. If pivoting techniques are used then this process is usually a stable and reliable one, although in the case of sparse systems the underlying algorithms and data structures can be complicated (see [4]). For problems which have certain structures, pivoting may not be necessary, as is the case for symmetric positive-definite matrices.

There have been many attempts to adapt direct schemes to parallel architectures (see, for example, [1,12,13]), but these approaches are very much architecture-dependent. Thus Saad [13] has considered LU algorithms for bus and ring topologies with distributed memory, while the approach of Ortega [12] is very much a fine-grained one suitable for SIMD machines. Furthermore, efficient parallel direct algorithms are heavily dependent on the structure of A with algorithms for banded systems (see [3], for example), being entirely different to those for sparse systems [4], which in turn are entirely different to the full dense case. Most iterative schemes, on the other hand, have a very simple and conceptually appealing algorithmic structure in that they can often be written very simply in terms of level 1 and level 2 BLAS, as is the case for the Jacobi and Conjugate Gradient methods, for example. Such iterative schemes are readily parallelizable and the structure of the algorithm does not change if A is full, banded or sparse.

On the other hand a different type of structure often has to be imposed on A (such as diagonal dominance or symmetric positive-definiteness or an M-matrix) in order to guarantee the convergence of some iterative algorithms. Furthermore, even if convergence is guaranteed it may be slow and may have to be accelerated by a preconditioning process which itself may not be readily parallelizable.

In order to overcome some of these difficulties associated with iterative schemes, Burrage et al. [2] presented a completely general technique for deflating the eigenvalues of the iteration matrix which either slow or cause divergence. This process has also been extended by Erhel et al. [5] to a preconditioning approach based on an invariant subspace approximation for the restarted GMRES algorithm. This approach will be applied to the problem of minimizing the GCV in Section 3, after first giving a brief description.

The technique is based on an idea due to Shroff and Keller [14] for solving nonlinear parameter-dependent problems, which in turn represents an extension of an adaptive condensation technique proposed by Jarausch and Mackens [9] for symmetric nonlinear problems. Shroff and Keller [14] call this technique the Recursive Projection method. It is based on the fact that divergence or slow convergence of the fixed-point iteration scheme

$$y^{(k+1)} = F(y^{(k)}, \lambda) \tag{7}$$

is due to the eigenvalues of F_{y^*} (the Jacobian of F evaluated at the fixed-point y^*) approaching or leaving the unit disk. The Recursive Projection method recursively approximates the eigenspace $\mathbb{P}$ corresponding to the unstable or slowly converging modes using the iterates of the fixed-point

iteration. A coupled iteration process takes place by performing Newton iteration on $\mathbb{P}$ and fixed-point iteration on $\mathbb{Q}$ (the orthogonal complement of $\mathbb{P}$) where fast convergence is assured. The scheme will be particularly effective if the dimension of $\mathbb{P}$ is small.

In addition to these techniques, Jarausch [10] has considered a different approach which effectively decouples the problem by the construction of right singular subsystems associated with the Jacobian of the problem. The use of invariant subspaces is avoided by transforming the nonlinear system by a so-called orthogonal rotator matrix. This approach is called by Jarausch [10] the *ideal normal equation approach* and it avoids the squaring of the singular values by the usual technique of normalizing the equations. In this case the singular values of the transformed problem and the original problem are the same.

In spite of considerable recent work on the application of these subspace approaches to nonlinear problems little appears to have been done in applying these techniques computationally to linear system of equations until the work of Burrage et al. [2] and Erhel et al. [5]. This is described briefly below.

For the linear system given by (6), a splitting of $A = M - N$ gives an iteration scheme of the form

$$My^{(k+1)} = Ny^{(k)} + b, \tag{8}$$

so that the fixed-point formulation of (7) is given by

$$y = F(y), \quad F(y) = M^{-1}Ny + M^{-1}b. \tag{9}$$

Let $\mathbb{P}$ be the invariant subspace of dimension r for the iteration matrix

$$H = M^{-1}N,$$

I_r be the identity matrix of order r and let Z be the orthogonal basis of $\mathbb{P}$. Thus with

$$Q = I - ZZ^{\top}, \qquad P = ZZ^{\top}, \qquad I_r = Z^{\top}Z, \qquad QP = 0 \tag{10}$$

and writing

$$y = (P + Q)y = Py + q = Zu + q, \quad u = Z^{\top}y, \tag{11}$$

then (7), (9) and (11) imply

$$\begin{aligned} &(I_r - Z^{\top}HZ)u = Z^{\top}M^{-1}b + Z^{\top}Hq, \\ &q = Q(M^{-1}b + Hq + HZu), \\ &y = Zu + q. \end{aligned} \tag{12}$$

Burrage et al. [2] have proposed a number of iteration schemes based on (12) including Jacobi, Gauss–Seidel and Reverse Gauss–Seidel schemes. These can be written in the general iterative form

$$\begin{aligned} u^{(k+1)} &= (I_r - Z^{\top}HZ)^{-1}Z^{\top}(M^{-1}b + Hq^{(i)}), \\ q^{(k+1)} &= (I - ZZ^{\top})(M^{-1}b + Hq^{(k)} + HZu^{(j)}), \end{aligned} \tag{13}$$

where the relationship between i, j and the method is given by

i	j	method
k	k	Jacobi
k	$k+1$	Gauss–Seidel
$k+1$	k	Reverse Gauss–Seidel

In the case of Reverse Gauss–Seidel it is understood that the q-iteration is performed first. Note that here q represents the fixed-point iteration and u the Newton iteration.

It can also be seen from (13) that Gauss–Seidel and Reverse Gauss–Seidel have very similar properties in that they both compute the same sequence but with different starting and finishing values. This similarity is confirmed by a study of the spectra of the iteration matrices. These are (from [2]) given by

$$\begin{aligned} \sigma(J_G) &= \{0, \sigma(g_q^* + g_p^* h_q^*)\}, \\ \sigma(J_R) &= \{0, \sigma(g_q^* + g_p^* h_q^*)\}, \end{aligned} \tag{14}$$

while for the Jacobi scheme the eigenvalues of J satisfy

$$\mathrm{Det}(\lambda^2 I - \lambda g_q^* + g_p^* h_q^*) = 0, \tag{15}$$

where

$$\begin{aligned} g_q^* &= (I - ZZ^\top) H (I - ZZ^\top), \\ g_p^* &= (I - ZZ^\top) HZ, \\ h_q^* &= Z(I_r - Z^\top HZ)^{-1} Z^\top H (I - ZZ^\top). \end{aligned}$$

In the case that $\mathbb{P}$ is invariant, $HZu \in \mathbb{P}$ and so from (10) it can be seen that $(I - ZZ^\top)HZu = 0$. Hence the spectral norm of all three iteration schemes is given by

$$\rho((I - ZZ^\top)H).$$

Recall here that H is the iteration matrix of the underlying iteration scheme, and this underlying scheme can be chosen depending on both the problem and the architecture. In the case of a parallel environment a Jacobi or block Jacobi iteration may be appropriate in which case M will be diagonal or block diagonal; while in a sequential environment Gauss–Seidel or block Gauss–Seidel or SOR schemes may be more appropriate as this will lead to faster convergence but less parallelism.

Eigenvectors can be appended to Z in essentially two different ways which depend on what sort of convergence properties are desired. This involves the development of a cost function which relates implementation costs to convergence rates and which can be automatically interrogated every so often to see if it is worthwhile to increase the dimension of Z (see [2]). In the case that at most two eigenvectors are going to be deflated at any given time, then Burrage et al. [2] propose using the modified Gram–Schmidt process in which only

$$\begin{aligned} w_1 &= \Delta q^{(k)} / \|\Delta q^{(k)}\|, \\ w_2 &= \Delta q^{(k-1)} - w_1 w_1^\top \Delta q^{(k-1)} \end{aligned}$$

need be computed. In this case

$$Z_1 = \begin{cases} (w_1), & \|w_2\| \ll \|w_1\|, \\ (w_1, w_2/\|w_2\|), & \text{otherwise.} \end{cases}$$

and so $Z = (Z\ Z_1)$, with the vectors of Z_1 being appended to Z.

As more eigenvalues are removed the basis Z will become increasingly inaccurate due to the loss of orthogonality in the Gram–Schmidt process and so added eigenvectors in Z_1 can be orthogonalized against the previous basis.

3. Minimizing the GCV function

In the evaluation of the GCV function the matrix inverses are not explicitly calculated and stored. Instead the evaluations are performed by solving the general matrix problem given by (6) where A has the form $A = Q + \lambda I$ and $b = z$ in (4). In order to do this, a way is needed to calculate the denominator of (4) which involves calculating the trace of A^{-1} without reducing to a tridiagonal form. As a result Hutchinson [7] has suggested the use of a stochastic trace estimator.

Thus if u is a vector with elements being samples from a random variable which takes the values 1 and -1, each with probability of $1/2$, then an unbiased estimate for the trace of a matrix B is given by $\mathrm{tr}(B) = u^{\mathrm{T}} Bu$. By first solving the system $Av = u$ for v, then the trace of the inverse of A is given by $\mathrm{tr}(A^{-1}) = u^{\mathrm{T}} v$. In this form with $Ay = b$ and $Av = u$, the GCV function can be written as

$$GCV(\lambda) = \frac{(1/n)(y^{\mathrm{T}} \cdot y)}{[(1/n)(u^{\mathrm{T}} \cdot v)]^2}.$$

Hence the main task in the evaluation of the GCV function now involves the repeated solution of the two linear systems of equations of the form

$$Ay = b, \qquad Av = u, \qquad A = (Q + \lambda I), \tag{16}$$

at each iteration step. It should be added that since the stochastic trace estimator is essentially a Monte Carlo simulation, n should be at least several hundreds in order for this process to be sufficiently accurate.

Since the matrix A is dense, any benefit arising from using an iterative method requires convergence in significantly less than n iterations to be competitive with the Householder approach. Numerical testing has shown that as convergence takes place within the GCV process, $Q + \lambda I$ becomes more and more ill-conditioned as λ approaches its minimum value (which is usually small and positive). For the problems of interest here, n can vary from a value of several hundreds up to 16,000 and the condition numbers of the matrices involved can become huge due to the presence of some very small positive eigenvalues.

In order to enhance the performance of this iterative approach, a number of modifications have been made to the original direct code due to Hutchinson [8]. These have been documented fully by Williams and Burrage [15], but a brief overview is given below.

The search algorithm used by Hutchinson [8] to minimize the GCV function is a Golden Section search, and depending on the desired accuracy and n up to 40 GCV evaluations may be needed. If

Householder transformations are used to tridiagonalize the influence matrix first, then the evaluation of the GCV function is done very quickly since only a sequence of tridiagonal solves are needed. Consequently, little attention need be given to how the search is carried out for minimizing λ. However, this becomes an important issue if full matrix systems are solved at each evaluation as is the case for the iterative approach. Thus the Golden Section search has been replaced by Brent's algorithm. This method starts with a bracketing triplet of points as before, and fits a parabola to these three points. The abscissa of the minimum of this parabola is then the point at which the function is evaluated next. This is then done repeatedly until the minimum is contained in a small subinterval (length about $2 \times tol$ where tol is the fractional accuracy specified). Using this algorithm, λ can be found in around 15 evaluations (for n of moderate size between 1000 and 3000). It should be noted that the success of this search depends on the "smoothness" of the function. The GCV function can have more than one local minimum if the accuracy tolerance for the iterative matrix solver is not set small enough.

The time taken for the iterative solution of a matrix system can also be significantly reduced if a reasonably good initial guess for the solution vector can be supplied. Since the system being solved changes by a relatively small amount from iteration to iteration during the minimization of the GCV, each solution vector can be kept and used as the initial guess for the next one. But this can be improved on by taking, as an initial guess, a linear combination of more than one of the previous solution vectors. Two-term, three-term and four-term approximations have been obtained by Williams and Burrage [15]. Good results have been obtained by calculating a two-term approximation at the third iteration, a three-term at the fourth, and then four-term approximations for the remaining steps in the minimization process.

There are two ways to use the deflation process described in Section 2. At each minimization step, two linear systems of equations of the form given in (16) need to be solved with different right-hand sides.

In the first technique the deflation process can be used to first solve, for a given λ, $Ay = b$. At the end of the iteration process an orthogonal basis Z will have been computed. Since the coefficient matrix remains the same for solving $Av = u$, Z will also be an orthogonal basis of $\mathbb{P}$ for the iteration matrix (as long as the same iteration technique is used in both cases). Thus deflation can continue using (12) as the starting point but with b replaced by u. For the next value of λ, the deflation process has to start anew with, if appropriate, a different iterative method.

In the second approach, the same deflation process can continue across all the sets of equations. But in order to do this the complete set of eigenvectors of

$$H_0 = M_0^{-1}N_0 = I - M_0^{-1}A \tag{17}$$

corresponding to (6) must be exactly the same as the complete set of eigenvectors of

$$H_l = M_l^{-1}N_l = I - M_l^{-1}(A + \lambda I) \tag{18}$$

corresponding to

$$(A + \lambda I)y = b. \tag{19}$$

Now from (17) and (18)

$$H_l = I - M_l^{-1}(M_0(I - H_0) + \lambda I),$$

so that if λ_0 and x are an eigenvalue and associated eigenvector of H_0 then

$$H_l x = x - (1 - \lambda_0) M_l^{-1} M_0 x - \lambda M_l^{-1} x.$$

Thus x will be an eigenvector of H_l if x is an eigenvector of both M_l and M_0, which in turn implies that x is an eigenvector of A as well. This can be achieved, for example, if

$$H_0 = p(A),$$
$$H_l = q(A + \lambda I),$$

where p and q are arbitrary polynomials of degree 1 or more satisfying $p(0) = 1$ and $q(0) = 1$. The simplest such polynomials occur if

$$M_0 = \omega_0 I, \qquad M_l = \omega_l I, \tag{20}$$

where $\omega_0, \omega_l \in \mathbb{R}$ in which case

$$H_0 = I - \frac{1}{\omega_0} A, \qquad H_l = I - \frac{1}{\omega_l}(A + \lambda I). \tag{21}$$

These iteration schemes are of course just examples of Richardson iteration, and the remaining question concerns the choice of ω_0 and ω_l. Since the systems that are being solved are symmetric positive-definite (but not diagonally dominant) all the eigenvalues of the systems are real and positive.

Thus let λ_m and λ_s denote, respectively, the maximum and minimum eigenvalues of A and let

$$c = \frac{\lambda_m}{\lambda_s} > 0$$

represent the conditioning of A. In this case it is easily seen from (21) that $\rho(H_0) < 1$ if and only if

$$\omega_0 > \theta \lambda_m, \quad \theta > \tfrac{1}{2}, \tag{22}$$

in which case

$$\rho(H_0) = \max\left\{1 - \frac{1}{\theta c}, 1 - \frac{1}{\theta}\right\}.$$

The value of θ which minimizes $\rho(H_0)$ is

$$\theta = \frac{1}{2}\left(1 + \frac{1}{c}\right)$$

and in this case

$$\rho(H_0) = \frac{c - 1}{c + 1}.$$

As the iterations proceed, information is obtained about the magnitude of the larger eigenvalues of H, but unfortunately these correspond to the smaller eigenvalues of A and so there is no information readily available about λ_m from this iteration process. However, there are a number of simple and cheap ways in which a ω_0 satisfying (22) can be chosen and this will be discussed in more detail in Section 4.

Table 1
Comparison timings

Data points	550	1025	1500
LU solve	0.377	2.354	7.443
Deflation	0.097	0.386	0.809
Speed-up	3.88	6.1	9.2

Thus given that a suitable ω_0 has been found, a value for ω_l for the updated system given by (19) can be chosen as

$$\omega_l = \omega_0 + \tfrac{1}{2}\lambda > \tfrac{1}{2}\lambda_m + \lambda. \tag{23}$$

This satsifies (22) for the system (19) since $\lambda_m + \lambda$ is the maximum eigenvalue of $A + \lambda I$. Thus (20), (21) and (23) imply

$$H_l = \frac{\omega_0}{\omega_l} H_0 - \frac{\lambda}{2\omega_l} I. \tag{24}$$

On the other hand it is not necessary for ω_l to satisfy the equality condition in (23), all that is necessary is that $\omega_l > (\lambda_m + \lambda)/2$. In fact even this last constraint need not hold as the GCV minimization progresses. If it does not hold, then $\rho(H)$ may be greater than one, but on the other hand the clustering may be less severe and the deflation process will ultimately turn any divergence into convergence. Thus as the deflation process continues from one set of equations to another, (12) is used from system to system. This requires only an updating of M and H as given by (20) and (24) and of the basis Z. Clearly, as more and more eigenvalues are extracted and as λ converges to its minimum value, there comes a point at which it is not efficacious to perform any more deflation as convergence takes place in a very small number of iterations.

4. Timings

In this section the results of some timing runs performed with different size data sets in Fortran77, on a Cray Y-MP 2D sited at the University of Queensland, are presented. The size of these problems is respectively 550, 1025 and 1500. The only reason for a restriction on the dimension of 1500 is due to limitation on the memory of the Cray Y-MP 2D. In order to see first how efficient the deflation process can be, some comparisons are made between the solution of (6) by a Cray library routine based on LU factorization and backward and forward substitution and the Reverse Gauss–Seidel deflation technique with an underlying Jacobi iteration. The timings are presented in Table 1 in seconds and were obtained by extracting three eigenvalues every five iterations. The systems here are only mildly conditioned.

It can be seen that the performance of the deflation approach compares very favourably with a highly optimized Cray library routine for doing a linear solve based on LU factorization especially in view of the fact that the problem is dense.

In order to explore the effectiveness of the deflation approach in more detail, a number of systems are solved in which the conditioning is controlled by the addition of a scalar value to the diagonal.

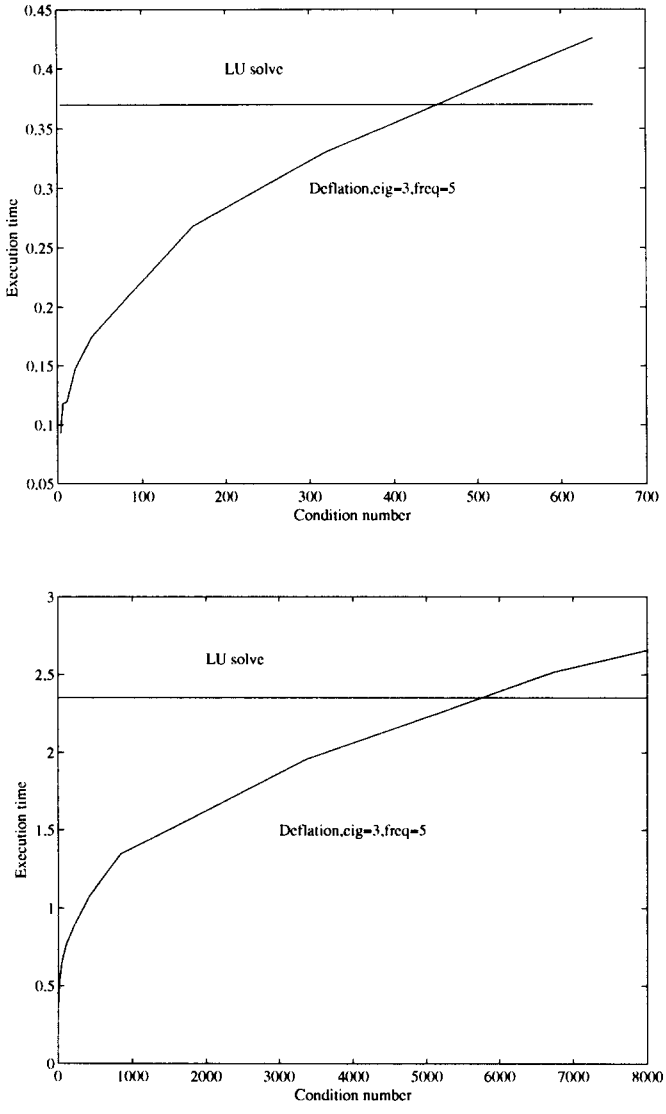

Fig. 1. $m = 550, 1025$.

(This represents the type of systems of equations that have to be solved in the Generalized Cross Validation process.) The timings for both the deflation process (in which three eigenvalues are extracted every five iterations) and the LU solve are plotted in Fig. 1 as a function of the condition number of the matrix for problems of dimension 550 and 1025.

As can be seen from these two figures, the deflation process slows down as the condition number increases. However, these graphs were produced using an extraction of three eigenvalues every five iterations and the timings can be improved by using different extraction protocols. However, clearly

Table 2
Original GCV timings

Data points	550	1025	1500
Householder	2.69	12.95	36.37

Table 3
RGS and Richardson iteration

eig	freq	w_0	n	iterations	time
2	5	80.0	550	110	2.44
2	4	80.0	550	94	2.40
2	4	380.0	550	85	2.21
2	10	22.0	1025	297	10.18
3	15	22.0	1025	293	14.54
*3	8	17.0	1500	262	32.74
*3	10	20.0	1500	315	33.07

the algorithm should attempt to work efficiently for differently conditioned matrices and in future research an adaptive algorithm will be developed which adapts the frequency with which differing numbers of eigenvalues are extracted to the convergence rates of the deflation process.

As discussed in Section 3 in order to be able to solve a sequence of equations of the form (4) by deflating over the complete sequence of linear systems, the Jacobi iteration has to be replaced by, for example, the Richardson iteration. The effect of this on the performance of the deflation code will now be analysed.

The Householder version of the spline routine was run on the Cray Y-MP 2D (using Cray library routines wherever appropriate) and the timings (in seconds) given in Table 2 were obtained.

The deflation routine using Reverse Gauss–Seidel and Richardson iteration was then run for a variety of parameters including the frequency ("freq") with which the eigenvalues are extracted and the number of eigenvalues ("eig") extracted at each deflation step as well as the initial relaxation parameter ω_0. It was seen in (22) that the convergence of Richardson iteration was guaranteed if

$$\omega_0 > \lambda_m/2,$$

where λ_m is the maximum eigenvalue of A. In addition to this convergence relationship there is the effect that ω_0 has on the clustering of the eigenvalues of H_0. It is easily seen from (21) that as $\omega_0 \to \infty$ the eigenvalues of H_0 become more and more clustered around unity. This can seriously degrade the performance of the deflation process, depending on the original clustering of the eigenvalues of A. This was described in [2]. In Table 3 a selection of timings and iteration numbers are given for various values of "eig", "freq" and ω_0.

In the case of the results labelled by (*) in Table 3, ω_l was chosen to satisfy (23). However, we also found that it was advantageous to choose ω_l adaptively in the program and the other results in Table 3 were produced using this approach. It should be noted that as λ converges to its minimizing value the linear systems become more and more ill-conditioned. This suggests that the performance of the deflation code can be substantially improved by allowing the various free parameters to vary as the conditioning worsens. Even now there is a speed-up of up to 20% over the original code which

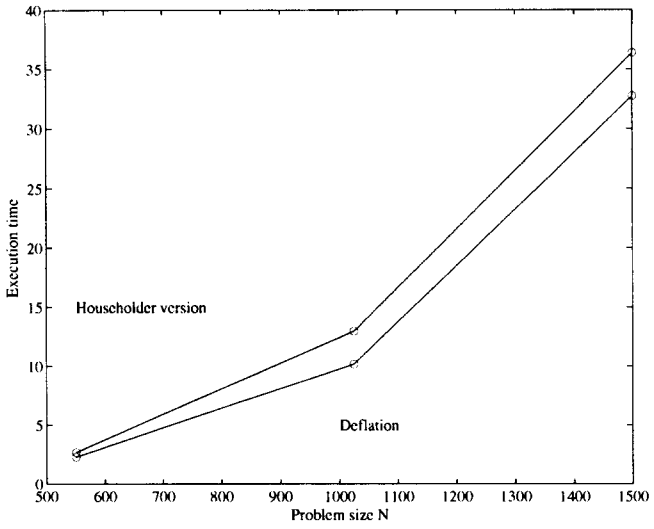

Fig. 2. Timings speed-ups.

uses the Cray library Householder routines and some execution times are graphed in Fig. 2.

With further fine-tuning it is expected that these times will be reduced substantially. It is also planned to automate this approach so that the algorithm will deflate an arbitrary number of eigenvalues with variable frequencies depending on an estimated convergence rate and an interrogation of a cost function. The initial relaxation parameter ω_0 can be chosen automatically by either estimating λ_m by a few iterations of the power method on the matrix A or by setting $\omega_0 = \text{Trace}(A)/2$ and both of these approaches produce a cheap and effective estimation of ω_0. It should also be noted that the deflation code currently runs at 75% of the peak performance of the Cray while the Householder code runs only at 50% of the peak performance. Finally, since the deflation process is rich in level 2 and level 3 BLAS, this technique should parallelize well.

5. Conclusions and extensions

The aim of this paper has been to show that the general deflation techniques described by Shroff and Keller [14] and Jarausch and Mackens [9] for nonlinear problems and applied to general systems of linear equations by Burrage et al. [2] and Erhel et al. [5] can be adapted to important applications such as the GCV approach. These approaches have also a more general applicability in the area of differential equations (see, for example, [10]).

Large differential equations are often stiff and can be characterized by rapidly changing and slowly changing modes. The techniques of Shroff and Keller [14], Jarausch [10] and Burrage et al. [2] can thus be applied directly to the differential equation system to produce an automatic partitioning technique into stiff and nonstiff methods which can then be solved as appropriate for implicit and explicit numerical methods. Some attempts at partitioning in this way have been done previously but never in a truly adaptive manner which these new techniques may now allow.

For example, consider the applicaton of the implicit Euler method to the system of ordinary differential equations given by

$$y' = f(y), \qquad y(x_0) = y_0, \qquad f : \mathbb{R}^m \to \mathbb{R}^m.$$

If this method is applied at a sequence of points $x_0, x_1, x_2, \ldots$ where $x_{n+1} = x_n + h_n$, then this method can be written as a nonlinear difference equation

$$F(y_{n+1}) = 0, \qquad F(y) = y - y_n - h_n f(y). \tag{25}$$

The technique of Shroff and Keller [14] can be applied directly to (25), but alternatively if some linearization, based on the modified Newton method is used, then at each step point x_{n+1} a sequence of linear systems of the form

$$\begin{aligned} &\Delta_n = y_{n+1}^{(k)} - y_n - h_n f(y_{n+1}^{(k)}), \\ &y_{n+1}^{(k+1)} = y_{n+1}^{(k)} - \Delta, \qquad k = 0, 1, \ldots, l-1, \\ &A_n = I - h_n J_n, \end{aligned} \tag{26}$$

has to be solved, where $J_n = f'(y_n)$.

At a given step it is customary to perform an LU factorization of A_n at the first iteration, so that for the remaining $l-1$ iterations only backwards and forward substitutions are needed in (26). However, if at the next step h_n is changed but the Jacobian is not changed (as is often the case in many stiff codes) these LU factors cannot be reused.

On the other hand, if the deflation process based on Richardson iteration described in Section 3 is used then not only can the deflation process continue across the iterations in one time step but also across many time steps (as long as the Jacobian is kept constant throughout this region of integration). This is because $J_{n+1} = J_n$ implies

$$H_{n+1} = r_n H_n + (1 - r_n) I, \quad r_n = \frac{h_{n+1}}{h_n},$$

and this is the same relationship as described in (24) in Section 3. Hence continuous deflation has an advantage over LU factorization in this respect.

The deflation techniques described in this paper and the more general techniques described by both Burrage et al. [2] and Erhel et al. [5] are currently being parallelized for the Intel Paragon and the results of this work will be presented in later papers.

Acknowledgements

Part of this work was done while Kevin Burrage and Alan Williams were, respectively, guest professor and guest academic of the Seminar für Angewandte Mathematik at ETH Zürich and while Jocelyne Erhel and Bert Pohl were Ethel Raybould Fellows in the Department of Mathematics at the University of Queensland in Australia.

Kevin Burrage and Alan Williams gratefully acknowledge the financial support provided by the Intel Corporation.

References

[1] R.H. Bisseling and G.G. van de Vorst, Parallel LU decomposition on a transputer network, Report Koninklijke/Shell Laboratorium, Amsterdam (1989).

[2] K. Burrage, J. Erhel and B. Pohl, A deflation technique for linear systems of equations, Research Report 94-02, Seminar für Angewandte Mathematik, ETH Zürich, Switzerland (1994).

[3] J.J. Dongarra and L. Johnsson, Solving banded systems on parallel processors, *Parallel Comput.* 5 (1987) 219–246.

[4] I.S. Duff, A.M. Erisman and J.K. Reid, Direct Methods for Sparse Matrices (Oxford University Press, London, 1988).

[5] J. Erhel, K. Burrage and B. Pohl, Restarted GMRES preconditioned by deflation, Research Report 94-04, Seminar für Angewandte Mathematik, ETH Zürich, Switzerland (1994).

[6] C. Gu, D. Bates, Z. Chen and G. Wahba, The computation of the GCV functions through Householder tridiagonalisation with application to the fitting of interaction spline models, *SIAM J. Matrix Anal.* 10 (1989) 459–480.

[7] M.F. Hutchison, A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines, *Comm. Statist. Simulation Comput.* 18 (1989) 1058–1076.

[8] M.F. Hutchison, The application of thin-plate smoothing splines to continent-wide data simulation, BMRC Research Report Ser. 28, Bureau of Meteorology (1991).

[9] H. Jarausch and W. Mackens, Numerical treatment of bifurcation problems by adaptive condensation, in: T. Küpper, H.D. Mittelmann and H. Weber, eds., *Numerical Methods for Bifurcation Problems* (Birkhäuser, Basel, 1987).

[10] H. Jarausch, Analyzing stationary and periodic solutions of systems of parabolic partial differential equations by using singular subspaces as reduced basis, Report 92, Inst. für Geometrie und Praktische Mathematik, Aachen, Germany (1993).

[11] L. Lau, M. Rezny, J. Belward, K. Burrage and B. Pohl, ADVISE–Agricultural Development Visualization Interactive Software Environment, in: Proceedings CONPAR 1994 (1994).

[12] J.M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems* (Plenum, New York, 1988).

[13] Y. Saad, Communication complexity of the Gaussian elimination algorithm on multiprocessors, *Linear Algebra Appl.* 77 (1986) 315–340.

[14] G.M. Shroff and H.B. Keller, Stabilization of unstable procedures: the recursive projection method. *SIAM J. Numer. Anal.* 30 (4) (1993) 1099–1120.

[15] A. Williams and K. Burrage, The implementation of a GCV algorithm in a high performance computing environment (in preparation).

[16] G. Wahba, How to smooth curves and surfaces with splines and cross validation, in: *Proceedings 24th Conference on the Design of Experiments*, US Army Research Office (1979).