A deflation technique for linear systems of equations

K. Burrage [1], J. Erhel [2] and B. Pohl [3]

Research Report No. 94-02
June 1994

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

[1]Department of Mathematics, University of Queensland, Brisbane 4072, Australia.
[2]INRIA, Campus de Beaulieu, 35042 Rennes, France.
[3]Seminar für Angewandte Mathematik, ETH Zürich, 8092 Zürich, Switzerland.

A deflation technique for linear systems of equations

K. Burrage [1], J. Erhel [2] and B. Pohl [3]

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Research Report No. 94-02 June 1994

Abstract

Iterative methods for solving linear systems of equations can be very efficient in a sequential or parallel computing environment if the structure of the coefficient matrix can be exploited to accelerate the convergence of the iterative process. However, for classes of problems for which suitable preconditioners cannot be found or for which the iteration scheme does not converge, iterative techniques are inappropriate. This paper proposes a technique for deflating the eigenvalues, and associated eigenvectors, of the iteration matrix which either slow down convergence or cause divergence. This process is completely general and works by approximating the eigenspace $\mathbb{P}$ corresponding to the unstable or slowly converging modes and then applying a coupled iteration scheme on $\mathbb{P}$ and its orthogonal complement $\mathbb{Q}$.

[1]Department of Mathematics, University of Queensland, Brisbane 4072, Australia.
[2]INRIA, Campus de Beaulieu, 35042 Rennes, France.
[3]Seminar für Angewandte Mathematik, ETH Zürich, 8092 Zürich, Switzerland.

1 Introduction

Computational techniques for solving linear systems of the form

$$Ay = b, \quad y \in \mathbb{R}^m \tag{1}$$

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 Duff et al. (1988)). For problems which have certain structures, pivoting may not be necessary, as in the case for symmetric positive definite matrices.

There have been many attempts to adapt direct schemes to parallel architectures (see, for example, Saad (1986), Ortega (1988) and Bisseling and van der Vorst (1989)), but these approaches are very much architecture dependent. Thus Saad (1986) has considered LU algorithms for bus and ring topologies with distributed memory, while the approach of Ortega (1988) is very much a fine-grained one suitable for SIMD machines. Bisseling and van der Vorst (1989) have investigated how to distribute the rows and columns of A in order to achieve good load balancing and small communication overheads. Their conclusion is that in certain circumstances cyclic distribution of rows and columns is an appropriate strategy. Other parallel direct schemes include blocking techniques which produce matrix-matrix algorithms rather than matrix-vector algorithms, thus introducing parallelism at the level 3 BLAS layer. This offers greater scope than level 1 or level 2 BLAS for exploiting parallelism and is the basis of LAPACK.

Another difficulty with the use of direct schemes in a parallel environment is that efficient algorithms are heavily dependent on the structure of A with algorithms for banded systems (see Dongarra and Johnsson (1987), for example) being entirely different to those for sparse systems (Duff et al. (1988)), which in turn are entirely different to the full dense case. 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. A notable example of slow convergence occurs when solving Laplace's equation by

the use of finite difference techniques on some mesh. If the region is square and the mesh is uniform with a grid size of $h = \frac{1}{N+1}$ then the spectral radii of the Jacobi and Gauss-Seidel iteration schemes are given by

$$\begin{aligned} \rho(H_J) &= \cos h\pi \approx 1 - \frac{1}{2}(h\pi)^2 + O(h^4) \\ \rho(H_G) &= (\cos h\pi)^2 \approx 1 - (h\pi)^2 + O(h^4), \end{aligned}$$

respectively. As the grid size is reduced both the convergence of the Jacobi and Gauss-Seidel schemes slow dramatically.

In order to overcome some of these difficulties associated with iterative schemes, we present here a completely general technique for deflating those eigenvalues of the iteration matrix, which either slow or cause divergence. This process takes place while the iterations are proceeding.

Thus this paper is organized as follows. In section 2 the deflation process is presented. It is based on an idea due to Shroff and Keller (1993) for solving nonlinear parameter-dependent problems, which in turn represents an extension of an adaptive condensation technique proposed by Jarausch and Mackens (1987) for symmetric nonlinear problems. In addition, two new iteration schemes are introduced based on a Gauss-Seidel approach. In section 3, this algorithm and the underlying iteration schemes will be described in full for linear problems, and convergence and complexity results given for various splittings of the matrix A. In section 4 some numerical results will be presented on problems which are either full or block banded together with a discussion on implementation techniques. The paper will conclude with some comments on the parallelization of this approach and its application to other areas of scientific computing such as differential systems.

2 Nonlinear systems

As mentioned in section 1, the process of accelerating the convergence of iterative methods by a deflation process which progressively extracts the largest eigenvalues (in magnitude) associated with the Jacobian of the problem has been studied by Jarausch and Mackens (1987) and Shroff and Keller (1993). It was applied by Shroff and Keller (1993) to the numerical solution of nonlinear parameter-dependent problems of the form

$$y = F(y, \lambda), \quad F : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$$

by a coupled iteration process which forces or accelerates the convergence of a fixed-point iteration scheme and represents an extension of the technique proposed by Jarausch and Mackens (1987) for solving symmetric nonlinear problems. Jarausch (1993) has considered a different approach which uses singular subspaces for splitting the fixed point equation associated with systems of

parabolic partial differential equations. Jarausch (1993) claims that this approach is an efficient one since the systems are effectively decoupled by the construction of right singular subspaces associated with the Jacobian of the problem. In addition the use of invariant subspaces is avoided by transforming the system by a so-called rotator matrix. This approach is called the 'ideal normal equation approach' in that it avoids the squaring of the singular values by the usual approach of normalizing the equations. In this case the singular values of the transformed problem have the same singular values as the original problem. In spite of considerable applications of these projection techniques to nonlinear parameter-dependent problems, little appears to have been done in applying these techniques computationally to linear systems of equations, and this is the focus of this paper. The notation that will be used is the notation used by Shroff and Keller (1993) which is very similar to the notation used by Jarausch and Mackens (1987) and Jarausch (1993). Furthermore, we will only consider applying the techniques used by Shroff and Keller (1993) to linear systems, although the approach of Jarausch (1993) also seems a fruitful one.

The approach of Shroff and Keller (1993), known as the **Recursive Projection Method**, is based on the fact that divergence or slow convergence of the fixed-point iteration scheme

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

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.

We will now give a brief outline of this process assuming that the problem to be solved is parameter independent and can be written as

$$y = F(y), \quad y \in \mathbb{R}^m. \tag{2}$$

Defining

$$\left.\begin{array}{rcl} y & = & p+q, \quad p = Py, \quad q = Qy = (I-P)y \\ f(p,q) & = & PF(p+q), \quad g(p,q) = QF(p+q) \\ f_p(p,q) & = & PF_y(y), \end{array}\right\} \tag{3}$$

where P and Q are the orthogonal projections of $\mathbb{R}^m$ onto $\mathbb{P}$ and $\mathbb{Q}$, respectively, and letting

$$h(p,q) = p + (I - f_p(p,q))^{-1}(f(p,q) - p);$$

then the coupled iteration scheme of Shroff and Keller (1993) is given by

$$\left.\begin{array}{rcll} p^{(k+1)} & = & h(p^{(k)}, q^{(k)}), & k = 0, \ldots, N-1 \\ q^{(k+1)} & = & g(p^{(k)}, q^{(k)}), & k = 0, \ldots, N-1 \\ y & = & p^{(N)} + q^{(N)}. & \end{array}\right\} \tag{4}$$

The overall iteration represents a Jacobi-type process in which the iterations are coupled by a Newton iteration and a fixed-point iteration. Clearly r, the dimension of $f_p(p, q)$, should be kept as small as possible in order to minimize the linear algebra costs. Ultimately, however, the size of r depends on how quickly convergence takes place. Here it should be noted that the fixed-point solution (p^*, q^*) of (4) satisfies

$$f(p^*, q^*) = p^*, \quad g(p^*, q^*) = q^*, \quad h(p^*, q^*) = p^*,$$

while F^* will denote the Jacobian of F evaluated at p^*, q^*.

The projectors P and Q can be computed by observing that if $Z \in \mathbb{R}^{m \times r}$ is an orthonormal basis for $\mathbb{P}$ then

$$P = ZZ^\top, \quad Q = I - ZZ^\top, \quad Z^\top Z = I_r.$$

The matrix Z can be recursively updated by noting from (4) that

$$\begin{array}{rcl} \Delta q^{(k)} & = & q^{(k+1)} - q^{(k)} = g(p^{(k)}, q^{(k)}) - g(p^{(k-1)}, q^{(k-1)}) \\ & = & g(p^{(k-1)} + \Delta p^{(k-1)}, q^{(k-1)} + \Delta q^{(k-1)}) - g(p^{(k-1)}, q^{(k-1)}) \\ & = & g_p^* \Delta p^{(k-1)} + g_q^* \Delta q^{(k-1)} + O(\varepsilon^2). \end{array} \tag{5}$$

Here ε represents terms that are hopefully negligible compared with the linear terms in the expansion and

$$\begin{array}{rcl} g_p^* = g_p(p^*, q^*) & = & QF^*P \\ g_q^* = g_q(p^*, q^*) & = & QF^*Q. \end{array} \tag{6}$$

This implies, that in the case of invariant subspaces $g_p^* = 0$ holds. Thus (5) and (6) imply that (5) is the power method

$$\Delta q^{(k)} = g_q^* \Delta q^{(k-1)} \tag{7}$$

and asymptotically $\{\Delta q^{(k)}\}$ will lie in the dominant eigenspace of g_q^* (assuming $\Delta q^{(0)}$ has a nonzero component in this direction). (7) can now be used to approximate the dominant eigenspace of g_q^* by forming a window of t difference vectors $S = \{\Delta q^{(k)}\}_{k-t+1}^{k}$ as the fixed-point iterations proceed and then computing an orthogonal basis U for span(S) by the modified Gram-Schmidt process. The eigenspace $B = U^\top F^* U$ is then formed and the eigenvectors

along with the Schur vectors of B are computed. For efficiency reasons Shroff and Keller (1993) therefore suggest taking $t = 2$ in which case S is factored as $S = \hat{S}T$, where $\hat{S}$ is orthogonal of dimension $m \times 2$ and T is upper triangular of dimension 2.

If $T_{11} \geq 10^3 T_{22}$ then just the first column of $\hat{S}$ is appended to the basis Z. Alternatively, if convergence is deemed to be slow due to a complex conjugate pair the first two columns of $\hat{S}$ are 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 $QF^*P \neq 0$. Hence Shroff and Keller (1993) suggest performing a subspace iteration on the columns of Z every so often. This takes the form

$$Z \rightarrow orth(F^*Z),$$

where $orth(F^*Z)$ denotes computing an orthonormal basis for the columns of F^*Z by the Gram-Schmidt process. Of course F^* is not computed directly, but can be computed by

$$F^*Z_i \approx \frac{1}{\varepsilon}(F(y + \varepsilon Z_i) - F(y)), \quad i = 1, \ldots, r.$$

The convergence properties of this approach can be analyzed by examining the Jacobian of (4) evaluated at (p^*, q^*). It is given by

$$J = \begin{pmatrix} h_p^* & h_q^* \\ g_p^* & g_q^* \end{pmatrix} \tag{8}$$

where

$$\begin{aligned} g_p^* &= QF^*P, \quad g_q^* = QF^*Q, \quad h_p^* = 0 \\ f_p^* &= PF^*P, \quad f_q^* = PF^*Q, \quad h_q^* = (I - f_p^*)^{-1} f_q^*. \end{aligned} \tag{9}$$

If the orthonormal basis is computed exactly then $g_p^* = 0$ and

$$\sigma(J) = \{0, \sigma(g_q^*)\},$$

and so the convergence of (4) is governed by the spectral norm of g_q^* which is progressively made smaller by the deflation process described above. On the other hand it should be noted that if a modified Newton process is used to compute p then $h_p^* \neq 0$, but is nevertheless small. In fact up to order $O(\epsilon^2)$ in (5), the global error for (4) can be written as

$$e^{(k+1)} = Je^{(k)}, \quad e^{(k)} = (p^{(k)^\top} - p^\top, q^{(k)^\top} - p^\top)^\top, \tag{10}$$

so that the error behaviour is determined by the behaviour of the power method. The ramifications of this when dealing with multiple or closely clustered eigenvalues of F^* will be discussed in more detail in section 4.

It is possible to modify the Jacobi type iteration scheme described in (4) to produce Gauss-Seidel or SOR schemes. In the former case there are two possible iterations, a Gauss-Seidel and Reverse Gauss-Seidel iteration, of the form

$$\begin{aligned} p^{(k+1)} &= h(p^{(k)}, q^{(k)}) \\ q^{(k+1)} &= g(p^{(k+1)}, q^{(k)}) \end{aligned} \tag{11}$$

and

$$\begin{aligned} q^{(k+1)} &= g(p^{(k)}, q^{(k)}) \\ p^{(k+1)} &= h(p^{(k)}, q^{(k+1)}), \end{aligned} \tag{12}$$

respectively.

Clearly these two processes should be very similar since they both compute the same sequence but with different starting and finishing procedures, and this is borne out by the following analysis of the spectra of the Jacobians associated with these schemes.

It is easy to show that the Jacobians associated with (11) and (12), denoted by J_G and J_R, are given by

$$J_G = \begin{pmatrix} h_p^* & h_q^* \\ g_p^* h_p^* & g_p^* h_q^* + g_q^* \end{pmatrix}$$

and

$$J_R = \begin{pmatrix} g_q^* & g_p^* \\ h_q^* g_q^* & h_p^* + h_q^* g_p^* \end{pmatrix} = \begin{pmatrix} I \\ h_q^* \end{pmatrix} \begin{pmatrix} g_q^* & g_p^* \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & h_p^* \end{pmatrix}.$$

Under the assumption that $h_p^* = 0$ (so that a Newton iteration is used for solving for p), the spectra of the associated Jacobian matrices are, respectively, given by

$$\sigma(J_G) = \{0, \sigma(g_q^* + g_p^* h_q^*)\} = \sigma(J_R) \tag{13}$$

while in the case of (4) the eigenvalues of J satisfy

$$\mathrm{Det}(\lambda^2 I - \lambda g_q^* + g_p^* h_q^*) = 0. \tag{14}$$

If the orthonormal basis is computed exactly, the spectral radii of all three schemes are exactly the same but in the presence of inaccuracies in Z, Gauss-Seidel and Reverse Gauss-Seidel and Jacobi behave differently, as will be seen in section 4.

Equations (11) and (12) can be modified to give corresponding SOR schemes of the form

$$\begin{aligned} p^{(k+1)} &= h(p^{(k)}, q^{(k)}) \\ q^{(k+1)} &= g((1-w)p^{(k)} + w p^{(k+1)}, q^{(k)}) \end{aligned}$$

and

$$\begin{aligned} q^{(k+1)} &= g(p^{(k)}, q^{(k)}) \\ p^{(k+1)} &= h(p^{(k)}, (1-w)q^{(k)} + wq^{(k+1)}), \end{aligned}$$

but again if the orthogonal basis is computed exactly, these SOR methods will behave in exactly the same way as their Gauss-Seidel counterparts.

3 Linear systems

The main thrust of this paper is to apply the schemes introduced in (4), (9) and (11) to linear systems of equations of the form given in (1). It should be noted that these schemes can be applied with very general iteration schemes such as Conjugate Gradient or GMRES techniques. GMRES was introduced by Saad and Schultz (1986) and is commonly used to solve large nonsymmetric linear systems, and Erhel, Burrage and Pohl (1994) have adapted these deflation techniques to produce a new variable preconditioning based on an invariant subspace approximation for the restarted GMRES algorithm. However, in this present paper only iteration schemes of the form

$$My^{(k+1)} = Ny^{(k)} + b \tag{15}$$

will be considered, where $A = M - N$ represents a splitting of A.

In this case the fixed-point formulation of (2) is given by

$$y = F(y), \quad F(y) = M^{-1}Ny + M^{-1}b. \tag{16}$$

Using the formulation of section 2, let $\mathbb{P}$ be the invariant subspace of dimension r for

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

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

$$Q = I - ZZ^\top, \quad P = ZZ^\top, \quad I_r = Z^\top Z, \quad QP = 0$$

and writing

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

then (2) and (16) imply

$$\begin{aligned} (I_r - Z^\top HZ)u &= Z^\top M^{-1}b + Z^\top Hq \\ q &= Q(M^{-1}b + Hq + HZu) \\ y &= Zu + q. \end{aligned} \tag{17}$$

The Jacobi, Gauss-Seidel and Reverse Gauss-Seidel schemes can then be written in the general iterative form

$$\begin{aligned} u^{(k+1)} &= (I_r - Z^\top HZ)^{-1}Z^\top(M^{-1}b + Hq^{(i)}) \\ q^{(k+1)} &= (I - ZZ^\top)(M^{-1}b + Hq^{(k)} + HZu^{(j)}) \end{aligned} \tag{18}$$

where the relationships 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 last case it is understood that the q iteration is performed first.

It can be seen from (18) 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. The spectra of these iteration schemes are again given by (12) and (13) with

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

In the case that $\mathbb{P}$ is invariant then the spectral norm of all three iteration schemes is $\rho((I - ZZ^\top)H)$. We 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.

The two different ways in which Z can be computed are described in section 2. The number of eigenvectors that are appended to Z depend on the nature of the desired convergence properties, and this involves the development of a cost function which can be interrogated every so often to see if it is worthwhile to increase the dimension of Z. In the following we show how the implementation of our algorithm proceeds and at the same time an attempt is made to develop a cost function which will allow for adaptive deflation.

This cost analysis is presented only for the dense case and it is also assumed that at most two eigenvalues are extracted at any given time. It will also be assumed that the implementation is a Jacobi implementation and that only multiplicative operations will be counted. These counts will, where possible, be written in terms of level 1 and level 2 BLAS in order to give an indication of likely parallel performance.

The cost function that will be developed will attempt to determine whether a method given by (18) with r extracted eigenvalues should be replaced by a method with $r + s$ extracted eigenvalues. Thus let Z_0 be $m \times r$, Z_1 be $m \times s$ and $Z = (Z_0\, Z_1)$ be $m \times (r + s)$.

Under the assumption of a Jacobi implementation and an increase in the number of eigenvalues by s, the steps to carry out a single iteration of (18) are as follows

I. Form $\bar{H} = HZ$. Since $\bar{H} = (HZ_0\, HZ_1)$, this requires an additional cost of

$$C_1 = (ms) \quad \text{level 1 BLAS of dimension } m.$$

II. Form $D = Z^\top HZ$. Since

$$D = Z^\top HZ = \begin{pmatrix} Z_0^\top HZ_0 & Z_0^\top HZ_1 \\ Z_1^\top HZ_0 & Z_1^\top HZ_1 \end{pmatrix}$$

and using the already formed matrix from I, this requires an additional cost of

$$C_2 = (s^2 + 2rs) \quad \text{level 1 BLAS of dimension } m.$$

III. Form Hq.

$$C_3 = (m) \quad \text{level 1 BLAS of dimension } m.$$

IV. Form $\bar{H}u$.

$$C_4 = (m) \quad \text{level 1 BLAS of dimension } r + s.$$

V. Form Q (update).

$$\begin{aligned} C_5 \quad = \quad & (r+s) \quad \text{level 1 BLAS of dimension } m \\ & + \\ & (m) \quad \text{level 1 BLAS of dimension } r + s. \end{aligned}$$

VI. Form $Z^\top$ (update).

$$C_6 = (r + s) \quad \text{level 1 BLAS of dimension } m.$$

VII. Solve $(I - D)u^{(k+1)} = v$.

The LU factors for $I - D_0 = L_0U_0$ can be used when factorizing $I - D$ in the form

$$\begin{pmatrix} L_0 & 0 \\ E & V \end{pmatrix} \begin{pmatrix} U_0 & X \\ 0 & Y \end{pmatrix}.$$

This can be done with the cost

$$C_7 = (r^2) \quad \text{level 1 BLAS of dimension } s + s^3/3.$$

The final cost factor is that arising from the formation of the new basis. Numerical testing has shown that in most cases it is sufficient to extract at most two eigenvalues at any given iteration. This can be done by the modified Gram Schmidt process. Thus only

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

need be computed. In this case

$$\begin{aligned} Z_1 &= (w_1), \quad \| w_2 \| << \| w_1 \| \\ &= (w_1, \frac{w_2}{\| w_2 \|}), \quad \text{otherwise} \end{aligned}$$

and the cost of this is

$$C_8 = (6) \quad \text{level 1 BLAS of dimension } m.$$

The costs estimated so far represent additional costs if extra eigenvalues are deflated. We now estimate the costs to implement (18) given that the dimension of Z is r and that no more eigenvalues will be extracted. This of course implies that the LU factorization of $I - D$ has already been completed. Suppose now that $\rho(QH) = \lambda_r$. If a desired tolerance to be achieved in some norm is ϵ, then the number of iterations, k_r, needed to achieve this accuracy is given by

$$k_r = \frac{\log(\epsilon)}{\log(\lambda_r)}. \tag{19}$$

Thus the total cost over k_r iterations will be

$$C_r = (m^2 + 4mr + r^2)k_r,$$

with the factor r^2 denoting the number of operations needed for the forward and back substitutions in solving for u given that the LU factors of $I - D$ have already been formed.

On the other hand if at the same stage in the iteration process s additional eigenvalues were to be extracted so that $\rho(QH)$ is now λ_s and the number of iterations to achieve accuracy is k_s then in the case of $s = 2$ the cost function would be

$$C_s = (m^2 + 4rm + (r+2)^2 + 8m)k_s + m(2m + 4r + 10) + \frac{14}{3}r^2.$$

Thus the 2 eigenvalues would be extracted if

$$C_s < \theta C_r, \quad \theta < 1$$

where $\theta(\approx 0.9)$ is a safety figure that prevents too many eigenvalues being extracted.

Ignoring the terms $r^2 k_r$, $(r+2)^2 k_s$ and $\frac{14}{3}r^2$, this gives

$$\left(k_s(1 + \frac{8}{m+4r}) + 2 + \frac{10 - 4r}{m+4r} \right) < \theta k_r.$$

Assuming r is small compared with m, this gives

$$\left(k_s(1 + \frac{8}{m+4r}) + 2 \right) < \theta k_r.$$

It should be noted that this analysis only holds if the iterations are converging. If at any stage in the iteration process the spectral norm is greater than one, then all the eigenvalues outside or on the unit disk should be deflated as quickly as possible.

4 Numerical results and conclusions

In this section a number of results are presented to show the efficacy of the deflation process previously outlined. In particular, three different problems are chosen for which extensive results are presented. We will also give some comments on the behaviour of the deflation process on other test problems that have been run.

The first problem comes from the solution of a two dimensional Poisson equation on the unit square with Dirichlet boundary conditions. This leads to the solution of a system of linear equations of order $m = N^2$ given by (1) in which A is block tridiagonal of the form $A = (I, T, I)$. Here I is the identity matrix of order N and T is the tridiagonal matrix $T = (1, -4, 1)$. It is known that both the Jacobi method and the Gauss-Seidel method will converge for this problem and that the spectral norms of the amplification matrices are, respectively,

$$\rho(H_J) = \cos\frac{\pi}{N}, \quad \rho(H_G) = (\cos\frac{\pi}{N})^2.$$

The second problem is artificially constructed so that the amplification matrix associated with Jacobi iteration has evenly distributed eigenvalues.

The third problem arises from the fitting of surfaces to a set of sparsely scattered meteorological stations in Australia (Burrage et al. (1994) and Williams and Burrage (1994)). The problem size can vary from a few hundred to almost 20,000. Here just three test sets are chosen of dimension 550, 1025 and 1500. These matrices are symmetric positive definite and the condition number is chosen by the addition of a positive scalar to the diagonal elements of the influence matrix.

In each case the problems are solved to very high precision and then using the 'exact' solutions, iteration takes place until a certain tolerance condition is satisfied which is based on a relative error criterion using the 2 norm. For the first two problems calculations are performed in Matlab on a Sparc10, while for the third problem calculations were performed on a Cray YMP-2D sited at the University of Queensland. Sparse Matlab techniques were used where appropriate.

Problem 1: 2D Heat Equation, m=144, tol = 10^{-10}
For this problem we investigate how the convergence depends on the number of eigenvalues extracted ('numeig') and the frequency with which the eigenvalues

are deflated ('freq'). It is assumed that at most two eigenvalues are deflated at any given time.

The results are given in the next four tables, and represent the number of iterations needed to achieve convergence. The results in the first three tables correspond to Jacobi, Gauss-Seidel and Reverse Gauss-Seidel, respectively, given that the underlying iteration (defined by the matrix M) is the Jacobi method. The fourth table was calculated using the same three deflation techniques but with an underlying Gauss-Seidel iteration and with eigenvalues being deflated every 15 iterations.

The number of iterations required to obtain convergence for the unaccelerated Jacobi and Gauss-Seidel, respectively, are 772 Iterations (2.42 seconds) and 389 Iterations.

Table 1: Jacobi method

numeig	1	2	3	4	5	6	7	8	9	10
freq = 5	∞	∞	∞	∞	252	252	169	169	134	134
freq = 10	541	464	464	169	169	99	99	77	77	66
freq = 15	444	302	302	121	121	93	93	73	73	73

Table 2: Gauss-Seidel method

numeig	1	2	3	4	5	6	7	8	9	10
freq = 5	599	599	391	391	193	193	100	100	78	78
freq = 10	529	186	186	169	169	111	111	71	71	62
freq = 15	432	326	326	116	116	89	89	71	71	71

It can be seen from these results that in the case of an underlying Jacobi iteration only about 8 eigenvalues need to be deflated at a frequency of one every 10 iterations to reduce the number of iterations by a factor of 10. In the case of an underlying Gauss-Seidel iteration only 5 eigenvalues need to be deflated at a frequency of every 15 iterations to reduce the number of iterations by a factor of 9.

Another important point to note here is that the eigenvalues when deflated are often fairly inaccurate (this is why the Jacobi technique diverges if only a few eigenvalues are deflated too quickly) but that as the iterations proceed these eigenvalues themselves become more and more accurate.

Table 3: Reverse Gauss-Seidel method

numeig	1	2	3	4	5	6	7	8	9	10
freq = 5	600	600	380	380	176	176	99	99	72	72
freq = 10	532	215	215	151	151	98	98	74	74	64
freq = 15	438	254	254	119	119	92	92	72	72	72

Table 4: Gauss-Seidel iteration, freq = 15

numeig	1	2	3	4	5	6	7	8	9	10
RGS	167	86	86	69	47	47	47	47	47	47
GS	169	82	82	64	46	46	46	46	46	46
Jacobi	169	88	88	70	47	47	47	47	47	47

The first three tables show that there are some differences between the three techniques especially if eigenvalues are deflated too frequently but that as the frequency becomes longer there is very little difference between these techniques, which is to be expected from the theoretical results given in section 3. There is of course a considerable difference between using an underlying iteration of Jacobi compared with Gauss-Seidel as can be seen from comparing table 4 with the first three tables.

Of course a reduction in the number of iterations by a factor of 10 does not imply a similar reduction in time because of the additional overheads imposed by the deflation process. Table 5 gives the number of iterations and times using Reverse Gauss-Seidel and an underlying Jacobi iteration with a frequency of 10.

Table 5: Reverse Gauss-Seidel method, freq=10

numeig	1	2	3	4	5	6	7	8
its	532	403	192	172	102	99	81	78
time	4.36	3.62	1.67	1.70	1.18	1.26	1.22	1.32

The speed-ups in both time and iteration over unaccelerated Jacobi are given in Figure 1. The speed-ups in time are somewhat disappointing compared with the iteration speed-up, but it should be recognized that Matlab is not

an appropriate vehicle for comparing times of different codes as there are high overheads associated with loop structures within Matlab.

In order to see the performance of the deflation process on a larger problem the dimension of the heat equation problem was increased to $m = 900$ and a tolerance of 10^{-8} used as a relative error convergence test. The results are given in table 6 and speed-ups in time and iteration over unaccelerated Jacobi are given in Figure 2.

Table 6: unaccelerated Jacobi: 3519 iterations, 95.78 seconds

freq	5	10	15	20	25	30	35	40	45
numeig	52	33	25	21	19	17	15	15	13
its	132	172	196	240	253	272	294	332	356
time	75.3	48.9	36.7	33.9	31.6	29.1	29.8	32.5	29.8

Here speed-ups in time close to 3.5 are achieved, but again this speed-up is underestimated due to the way Matlab is implemented, and also due to the sparse matrix representations. (See problem 2 for further comments on this.)

Problem 2: Jacobian with evenly distributed eigenvalues, m=200, tol = 10^{-8}

For this problem Reverse Gauss-Seidel was used with an underlying Jacobi iteration. As many eigenvalues as necessary are extracted in order to attain convergence with either 1 or 2 eigenvalues being extracted every 'freq' iterations.

The unaccelerated Jacobi method took 4208 iterations and a time of 102.3 seconds to attain convergence. All calculations were again done in Matlab.

Table 7: Reverse Gauss-Seidel with Jacobi iteration

freq	2	4	6	8	10	12	14	18	25	50
eigs	160	84	62	50	46	39	37	31	27	17
its	161	170	188	204	232	251	272	292	352	452
time	61.6	22.6	17.1	15.0	14.7	14.8	14.6	13.3	16.8	16.3

The speed-ups in timing and iteration are presented in Figure 3. In this case a speed-up in time of about 8 is much better than for Problem 1. One reason for this is that the iteration matrix is dense whereas for Problem 1 it is sparse and in Matlab there are additional overheads for sparse linear algebra techniques

which are not apparent in the dense case.

Problem 3: Surface fitting problem, m = 550, 1025, 1500

For this problem a series of dense linear systems of the form (1) where $A = Q + \lambda I$ are solved which arise from the fitting of surfaces to rainfall data obtained from irregularly scattered meteorological stations sited in Queensland, Australia. Here $\lambda >> 0$ is a surface fitting parameter which is minimized within a cross–validation algorithm (see Williams and Burrage (1994)), but in the results presented here it will be used to control the conditioning of the problem, with a large λ implying a well–conditioned problem.

Three different weather data sets of dimension 550, 1025 and 1500 were chosen and initially solved on a Cray YMP-2D sited at the University of Queensland using a Cray library routine for LU factorization. The only reason for a limit of $m = 1500$ is due to memory limitations on the Cray. Solving these systems using the Cray routines for LU factorization and backward and forward substitution, the following timings in seconds were obtained:

Table 8: $\lambda = 5$

m	550	1025	1500
time	0.377	2.3535	7.443

The deflation process of Reverse Gauss–Seidel with an underlying Jacobi iteration was then run for these three different sets each with two different values for $\lambda, \lambda_1 = 5, \lambda_2 = \frac{5}{64}$. The second value for λ gives a problem of modest ill–conditioning, with a condition number for A in the range (160 – 425) depending on the size of m. For each of these six problems 'eig' eigenvalues are extracted at every 'freq' iterations and iterations and timings are presented for three different cases corresponding to eig = 2,3,4. the results are presented in tables 9 and 10.

Table 9: iteration count

		550		1025		1500	
eig	freq	λ_1	λ_2	λ_1	λ_2	λ_1	λ_2
2	5	16	46	21	66	21	71
3	5	14	34	16	66	16	62
4	10	21	41	21	76	21	81

For a given value of 'eig' a number of runs were performed for different values of 'freq' and only the optimal values in terms of timings are presented here.

Table 10: Timings in seconds

		550		1025		1500	
eig	freq	λ_1	λ_2	λ_1	λ_2	λ_1	λ_2
2	5	0.105	0.247	0.410	1.116	0.863	2.467
3	5	0.097	0.212	0.386	1.247	0.809	2.560
4	10	0.123	0.214	0.406	1.910	0.848	2.666

However there seemed to be little difference between the timings for a given λ as long as the eigenvalues are deflated on average between 0.8 and 0.25 eigenvalues per iteration. But if the eigenvalues are deflated sufficiently frequently the performance can be significantly affected. Nevertheless the timing speed-ups over the Cray library LU factorization routine is substantial, especially when bearing in mind that the problems here are dense. The speed-ups in time for the eig=3 case are plotted in Figure 4.

It should be noted that the conditioning of the matrix can also affect the performance of these accelerated iterative techniques, but it is likely in this case that further fine tuning will improve the results in the modestly ill–conditioned case.

Some additional testing has been performed on problems which have multiple eigenvalues or clustered eigenvalues. The convergence behaviour of the deflation process is described in (10), and because it is essentially the power method it is important to realize that the deflation process works differently in both the case of multiple eigenvalues and clustered eigenvalues.

Thus for example, on a problem of dimension 100 with 99 eigenvalues at 0.95 and one eigenvalue at 0.2 the deflation process will converge in a very few number of iterations by deflating only one eigenvalue. For this example the deflation process behaves as if the problem is of dimension 2 with one eigenvalue at 0.99 and another at 0.2. This is entirely consistent with the way the power method behaves on problems with multiple eigenvalues.

For problems which have clusters of eigenvalues, the behaviour of the deflation process depends on how finely the eigenvalues are clustered within each cluster. If the clustering within each cluster is very fine then the inherent errors in the deflation process will cause all the eigenvalues within the cluster to be treated the same and the process will behave as in the multiple eigenvalue case (without any noticeable loss of accuracy). As the clustering becomes less fine then the deflation process will extract a small number of eigenvalues from each cluster and consider the rest in that cluster as being multiple eigenvalues. Finally, at a moderate clustering the process may well attempt to extract nearly all the eigenvalues from the clusters which are near the unit disk.

As an example of this, a problem of dimension 100 with 10 clustered sets with 10 eigenvalues in each cluster was solved. In the two clusters closest to the unit disk the eigenvalues were equally spaced in the interval (0.990, 0.9987) and (0.892, 0.901). For this problem the Jacobi method could not converge in 5000 iterations, while the Reverse Gauss-Seidel technique with an underlying Jacobi iteration converged in 81 iterations. In this case up to 4 eigenvalues could be extracted every 5 iterations and 68 eigenvalues were extracted in total, with 9 eigenvalues being extracted from each of the first two clusters and 7 eigenvalues from the next cluster.

In conclusion it can be seen that the deflation process is a remarkably robust procedure working on a variety of pathological cases. The number of eigenvalues that have to be extracted is usually modest and the speed-up in time can be impressive. It is hoped to implement these procedures in a parallel environment in later papers, in a truly adaptive manner. It should also be noted that the deflation process can be used to extract an arbitrarily-sized set of eigenvalues in an efficient manner.

Acknowledgements:
The authors would like to thank Alan Williams of the Department of Mathematics at the University of Queensland for providing the timings given in problem 3 for the Cray YMP-2D.
Part of this work was done while Kevin Burrage was a guest professor 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.

References

Bisseling, R.H. and van de Vorst, G.G. (1989). Parallel LU Decomposition on a Transputer Network. *Report Koninklijke/Shell - Laboratorium, Amsterdam.*

Burrage, K., Williams, A., Erhel, J. and Pohl, B. (1994). The implementation of a Generalized Cross Validation algorithm using deflation techniques for linear systems. *In preparation*

Dongarra, J.J. and Johnsson, L. (1987). Solving banded systems on parallel processors. *Par. Comp.*, **5**, 219–246.

Duff, I.S., Erisman, A.M. and Reid, J.K. (1988). Direct methods for sparse matrices. Oxford University Press, London.

Erhel, J., Burrage, K. and Pohl, B. (1994). Preconditioned restarted GMRES. *In preparation.*

Jarausch, H. (1993). Analyzing Stationary and Periodic Solutions of Systems of Parabolic Partial Differential Equations by Using Singular Subspaces as Reduced Basis. *Bericht Nr. 92, Institut für Geometrie und Praktische Mathematik, RWTH Aachen.*

Jarausch, H. and Mackens, W. (1987). Numerical treatment of bifurcation problems by adaptive condensation. In *Numerical Methods for Bifurcation Problems*, (ed. Küpper, T., Mittelmann, H.D. and Weber, H.), Birkhäuser Basel.

Ortega, J.M. (1988) Introduction to parallel and vector solution of linear systems. Plenum Press, New York.

Saad, Y. (1986). Communication complexity of the Gaussian elimination algorithm on multiprocessors. *Lin. Alg. Appl.,* **77**, 315–340.

Saad, Y. and Schultz, M. H. (1986). GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **7**, 856–869.

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

Williams, A. and Burrage, K. (1994). The implementation of a GCV algorithm in a high performance computing environment. *In preparation.*

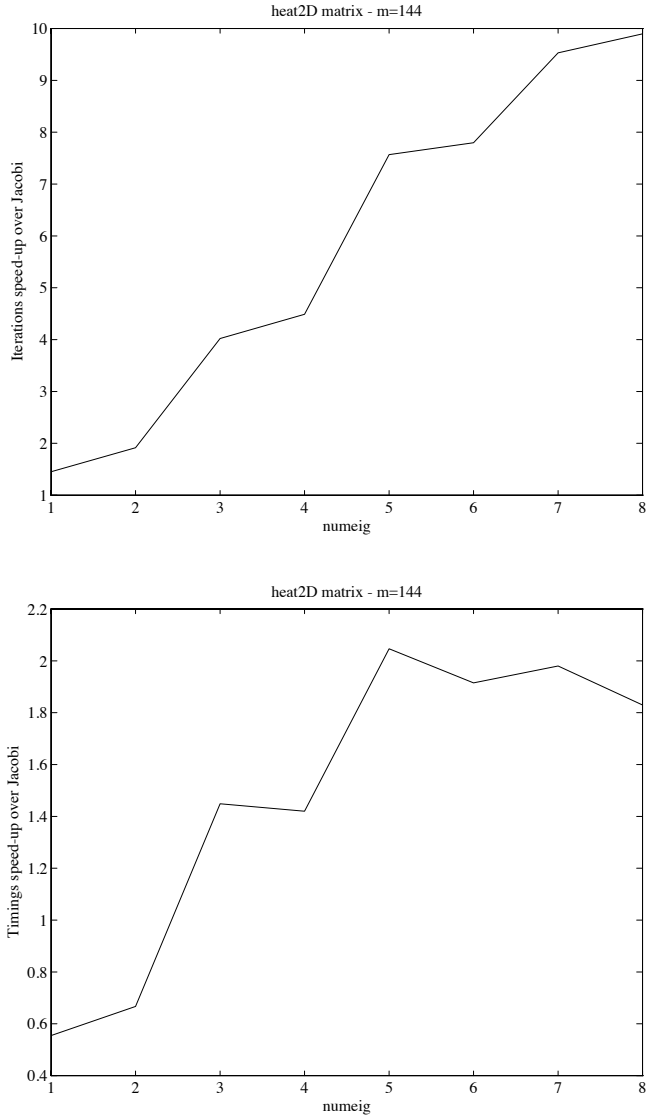

Figure 1: 2D heat equation, $m = 144$

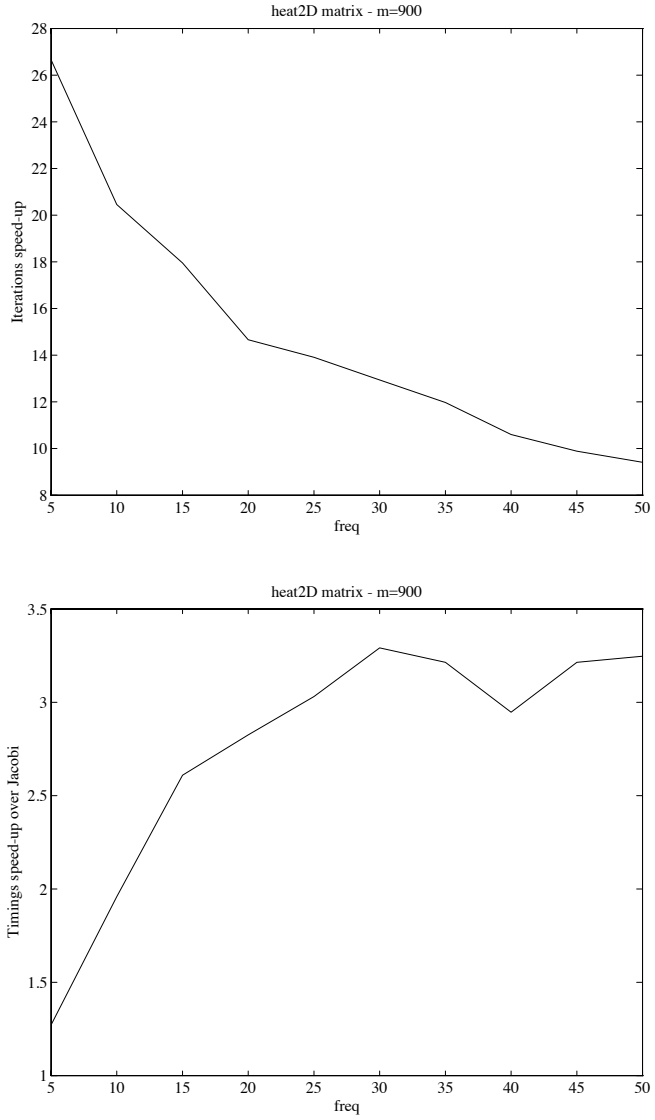

Figure 2: 2D heat equation, $m = 900$

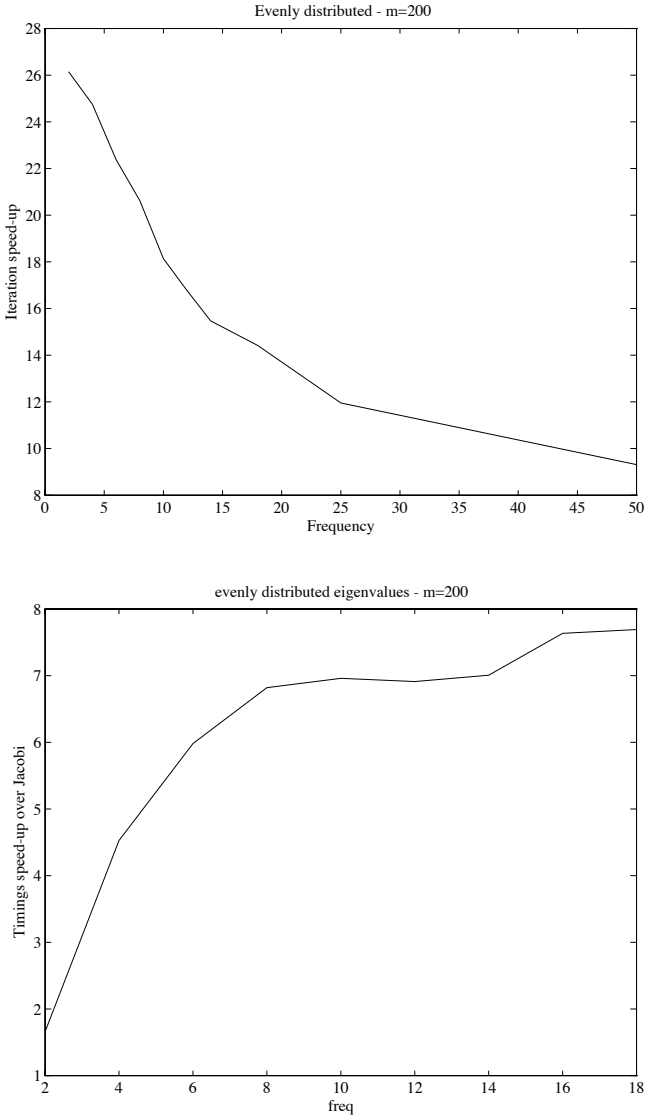

Figure 3: evenly distributed $m = 200$

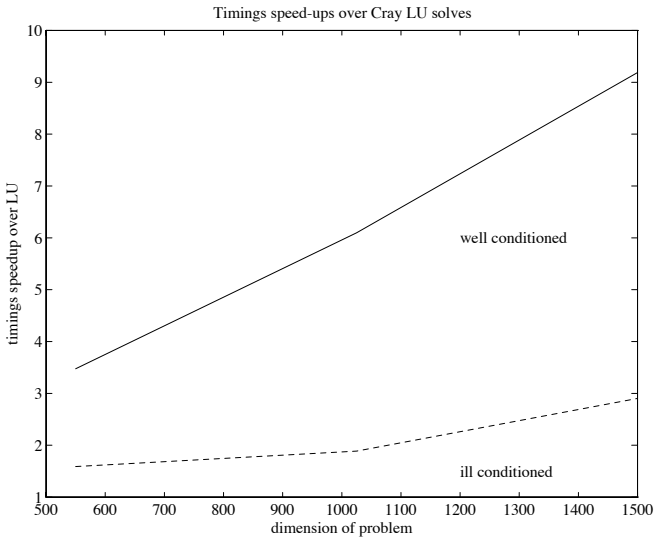

Figure 4: Surface fitting, eig = 3

Research Reports

No.	Authors	Title
94-02	K. Burrage, J. Erhel, B. Pohl	A deflation technique for linear systems of equations
94-01	R. Sperb	An alternative to Ewald sums
93-07	R. Sperb	Isoperimetric Inequalities in a Boundary Value Problem in an Unbounded Domain
93-06	R. Sperb	Extension and simple Proof of Lekner's Summation Formula for Coulomb Forces
93-05	A. Frommer, B. Pohl	A Comparison Result for Multisplittings Based on Overlapping Blocks and its Application to Waveform Relaxation Methods
93-04	M. Pirovino	The Inverse Sturm-Liouville Problem and Finite Differences
93-03	R. Jeltsch, X. Wang	Uniqueness of Piecewise Lipschitz Continuous Solutions of the Cauchy-Problem for 2×2 Conservation Laws
93-02	W.-A. Yong	Difference approximations to the global $W^{1,\infty}$ solutions of the isentropic gas equations
93-01	Ch. Lubich, K. Nipp, D. Stoffer	Runge-Kutta solutions of stiff differential equations near stationary points
92-15	N. Botta	Is the transonic flow around a cylinder always periodic?
92-14	K. Nipp, D. Stoffer	Invariant manifolds of numerical integration schemes applied to stiff systems of singular perturbation type - Part I: RK-methods
92-13	K. Nipp	Smooth attractive invariant manifolds of singularly perturbed ODE's
92-12	D. Mao	A Shock Tracking Technique Based on Conservation in One Space Dimension
92-11	K. Nipp, D. Stoffer	Attractive invariant manifolds for maps: Existence, smoothness and continuous dependence on the map
92-10	M. Fey, R. Jeltsch	A Simple Multidimensional Euler Scheme
92-09	M. Fey, R. Jeltsch	A New Multidimensional Euler Scheme
92-08	M. Fey, R. Jeltsch, P. Karmann	Numerical solution of a nozzle flow
92-07	M. Fey, R. Jeltsch, P. Karmann	Special aspects of reacting inviscid blunt body flow
92-06	M. Fey, R. Jeltsch, S. Müller	The influence of a source term, an example: chemically reacting hypersonic flow