SIAM J. Sci. Comput. © 1998 Society for Industrial and Applied Mathematics
Vol. 19, No. 4, pp. 1245–1260, July 1998 011

A DEFLATION TECHNIQUE FOR LINEAR SYSTEMS OF EQUATIONS*

K. BURRAGE†, J. ERHEL‡, B. POHL§, and A. WILLIAMS†

Abstract. Iterative methods for solving linear systems of equations can be very efficient 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 may be 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 $\mathbb{Q}$.

Key words. linear systems, deflation, preconditioning, iterative techniques

AMS subject classifications. 65F10, 65F15

PII. S1064827595294721

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 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 [2] for example). For problems which have certain structures, pivoting may not be necessary, as in the case for symmetric positive definite matrices. The question of when a direct method or an iterative method should be used is hard to resolve, since an informed answer will depend on both the structure of the problem and the target computer architecture. It is certainly true that many iterative schemes have a simple and conceptually appealing algorithmic structure in that they can often be written very concisely 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 the property of

*Received by the editors October 30, 1995; accepted for publication (in revised form) September 23, 1996; published electronically April 16, 1998. Part of this work was done while Kevin Burrage and Alan Williams were, respectively, a guest professor and visiting student at 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.
http://www.siam.org/journals/sisc/19-4/29472.html

†Department of Mathematics, University of Queensland, Brisbane, 4072, Australia (kb@maths.uq.oz.au, william@dancer.ca.sandia.gov).
‡INRIA, Campus de Beaulieu, 35042 Rennes, France (Jocelyne.Erhel@irisa.fr).
§Seminar für Angewandte Mathematik, ETH Zürich, 8092 Zürich, Switzerland (Bartholomaeus.pohl@ubs.com).

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 may not be suitable to the underlying computer architecture. 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

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

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

Of course approaches such as multigrid techniques can rapidly accelerate the convergence of iterative schemes by using them as smoothers in various sequences of coarsening and refining a discretization mesh. But this is at the cost of introducing considerable computational complexity, and the multigrid approach is not always appropriate when trying to exploit certain forms of parallelism.

In order to overcome some of these difficulties associated with iterative schemes, we present here a completely general iterative technique for solving linear systems of equations by adaptively deflating those eigenvalues of the iteration matrix, which either slow convergence or cause divergence. A Newton-like method (or direct method in the case of linear systems) is used on an invariant subspace corresponding to the eigenvalues of the iteration matrix which are near the unit disk, and an iterative scheme is used in the orthogonal subspace. As the iterations proceed, convergence is accelerated since the eigenvalues in the orthogonal subspace can be made small enough by deflation.

Since the iterative scheme is based on a splitting of the matrix, the matrix can be nonsymmetric, and any storage can be used. The matrix can be dense, sparse, or even a matrix-free technique is feasible.

The advantage of this approach is not only that it is conceptually very simple, but, as will be seen later, it can prove to be remarkably efficient. In some sense the deflation approach allows stationary methods such as Gauss–Seidel and Jacobi iteration to compete on an equal footing with powerful Krylov subspace methods.

The GMRES algorithm [6] is commonly used to solve large sparse nonsymmetric systems. The convergence behavior is related to the convergence of the Ritz values, and superlinear convergence has been established in [8]. The deflation process described in this present paper has the same effect.

We note that a similar approach to the deflation approach described here has been developed in [3] in which a preconditioner is built by deflation for restarted GMRESs based on approximating and updating an invariant subspace at each restart. This approach gives a much more robust scheme than the usual restarted GMRES algorithm and retains the superconvergence properties of full GMRES by the building of an appropriate preconditioner.

This 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 in [5] and [7]. It was applied in [7] 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 in [5] for solving symmetric nonlinear problems. Recently, [4] has considered a different approach which uses singular subspaces for splitting the fixed-point equation associated with systems of parabolic partial differential equations. 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 in [7] which is very similar to the notation used in [5] and [4]. Furthermore, we will only consider applying the techniques used by [7] to linear systems, although the approach in [4] also seems a fruitful one.

The approach developed in [7], known as the *recursive projection method*, is based on the fact that divergence or the 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.

It should be noted that this approach has some similarities with partitioning techniques for solving stiff systems of ordinary differential equations in which attempts are made to split the system into stiff and nonstiff subspaces. For this reason we will sometimes refer to $\mathbb{P}$ as the stiff subspace.

Thus the outline of the paper is as follows. In section 2 the deflation algorithm will be described in full for linear systems of equations, and a new modification will also be described. Convergence results will also be given. In section 3 various implementation techniques will be addressed. In section 4 some numerical results are given in both Matlab (for investigating convergence issues) and Fortran (for comparing computational performance). Both dense and sparse systems will be considered, and comparisons will be made with other effective schemes such as LU factorization and standard iterative schemes such as conjugate gradient techniques on a Cray YMP-2D located at the University of Queensland. Section 5 will be devoted to the application of deflation to areas of scientific computing such as the numerical solution of ordinary differential systems and the generalized cross validation (GCV) techniques for the fitting of nonsmooth data. These areas involve the repeated solution of successive sets of linear systems in which the system matrix is repeatedly updated by a constant value on the diagonal, and deflation can be shown to be very effective in this situation.

2. Deflation applied to linear systems.

Consider the linear system given by (1) and consider the splitting

$$My = Ny + b$$

with $A = M - N$, where M is nonsingular. The underlying iteration scheme that will be considered in this paper will be of the form

$$My^{(k+1)} = Ny^{(k)} + b. \tag{2}$$

This of course can be written in fixed-point form

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

where

$$F(y) = Hy + M^{-1}b, \quad H = M^{-1}N. \tag{3}$$

Now let $\mathbb{P}$ be a subspace of dimension r, I_r be the identity matrix of order r, and Z be an orthonormal basis of $\mathbb{P}$. Let Q and P be the projections on the orthogonal of $\mathbb{P}$ and on $\mathbb{P}$. We have

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

Now partition y as

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

then (2) and (3) imply

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

Here u represents the solution in the stiff subspace. A coupled iteration then can be performed between u and q. In [7] only a Jacobi-type coupling was considered but in fact other couplings are possible. These will be referred to as the Jacobi, Gauss–Seidel (GS), and reverse Gauss–Seidel (RGS), couplings, and these can then be written in the general iterative form

$$\begin{cases} Wu^{(k+1)} = Z^\top (M^{-1}b + Hq^{(i)}), \\ q^{(k+1)} \quad = Q(M^{-1}b + Hq^{(k)} + HZu^{(j)}), \end{cases} \tag{4}$$

where

$$W = I_r - Z^\top HZ.$$

The relationships between i, j, and the coupling is given by

i	j	coupling,
k	k	Jacobi,
k	$k+1$	GS,
$k+1$	k	RGS.

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

It can be seen from (4) that GS and RGS couplings have very similar properties in that they both compute the same sequence but with different starting and finishing values.

There are a number of factors that should be borne in mind when considering this deflation approach. The most significant is that r should be kept as small as possible, since a linear system of dimension r has to be solved at each step of the iteration process and since r vectors (the basis Z) must be stored. This of course has to be judged in terms of the number of iterations to attain convergence. 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 computer 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 GS or block GS or SOR schemes may be more appropriate as this will lead to faster convergence but less parallelism depending on the ordering of the components.

Before studying the convergence properties, we must ensure that this method is well defined or, in other words, that W is nonsingular. We have the following result.

PROPOSITION 2.1. *W is nonsingular if and only if unity is not an eigenvalue of PHP.*

Proof. W is singular and is equivalent to unity being an eigenvalue of $Z^T HZ$ which is in turn equivalent to unity being an eigenvalue of PHP. $\square$

Now let us assume that W is nonsingular. The convergence properties can be analyzed by examining the iteration matrix associated with the fixed-point scheme. Let

$$e^{(k)} = (Zu^{(k)^\top} - Zu^\top, q^{(k)^\top} - q^\top)^\top;$$

then

$$e^{(k+1)} = Je^{(k)}.$$

In the case of the three couplings, Jacobi, GS, and RGS, the Jacobians associated with these schemes can be written as

$$J_J = \begin{pmatrix} 0 & C \\ E & B \end{pmatrix},$$

$$J_G = \begin{pmatrix} 0 & C \\ 0 & EC + B \end{pmatrix},$$

$$J_R = \begin{pmatrix} B & E \\ CB & CE \end{pmatrix},$$

where

$$E = QHP, \quad B = QHQ, \quad C = P(ZW^{-1}Z^\top)PHQ.$$

Thus the spectra of the associated Jacobian matrices for a GS and RGS coupling are, respectively, given by

$$\sigma(J_G) = \{0, \sigma(B + EC)\} = \sigma(J_R),$$

while in the case of Jacobi coupling the eigenvalues of J_J satisfy

$$\mathrm{Det}(\lambda^2 I - \lambda B - EC) = 0.$$

Now the idea of the deflation method is to choose an invariant subspace $\mathbb{P}$ corresponding to the eigenvalues close to 1. We have the following theoretical result.

PROPOSITION 2.2. *Let $\mathbb{P}$ be an invariant subspace for the matrix H, with an orthonormal basis Z, Q, and let P be the projections on the orthogonal of $\mathbb{P}$ and on $\mathbb{P}$. Let us assume that* 1 *is not an eigenvalue of PHP and that $\rho(QHQ) < 1$, where $\rho(QHQ)$ is the spectral radius of QHQ. Then the method* (4) *is well defined and convergent for all three schemes.*

Proof. If $\mathbb{P}$ is an invariant subspace for the matrix H then

$$E = QHP = 0$$

and the spectral radii of all three schemes are exactly the same and are given by

$$\rho(B) = \rho(QHQ).$$

The conclusion follows readily. $\square$

Because the invariant subspace $\mathbb{P}$ is only approximated in practice, the matrix E is not null but hopefully small. In the presence of inaccuracies in Z, GS, RGS, and Jacobi couplings behave differently, as will be seen in section 4.

3. Implementation issues.

An important question that must be addressed is the computation of Z. The matrix Z in fact can be recursively updated. If the subspace $\mathbb{P}$ is invariant, then $QHZ = 0$ and (4) can be written as

$$q^{(k+1)} = Q(M^{-1}b + Hq^{(k)})$$

so that

$$\Delta q^{(k+1)} = q^{(k+1)} - q^{(k)} = (QHQ)\Delta q^{(k)}. \tag{5}$$

This implies, using the power method, that asymptotically $\{\Delta q^{(k)}\}$ will lie in the dominant eigenspace of $B = QHQ$ (assuming $\Delta q^{(1)}$ has a nonzero component in this direction). This eigenspace is the eigenspace of H for remaining dominant eigenvalues, so that the basis Z can be increased with new vectors. Although the subspace used is not exactly invariant, other terms in the expansion of $\Delta q^{(k+1)}$ are hopefully negligible.

Equation (5) can now be used to approximate the dominant eigenspace of B by computing the Krylov subspace spanned by a window of *wind* difference vectors $\{\Delta q^{(k)}\}_{j-wind+1}^{j}$ as the fixed-point iterations proceed and then computing an orthogonal basis S of this subspace. The Schur vectors T of the dominant eigenspace of $S^T HS$ are computed and ST approximates Schur vectors of H.

There are a number of ways that this deflation algorithm can be implemented. Currently the user is allowed to specify the size of the Krylov subspace (denoted *wind*), the (constant) number of eigenvalues that are deflated at each outer iteration (denoted def), the (constant) frequency at which these eigenvalues are deflated (called $freq$), and the maximal number of deflated eigenvalues (denoted *numeig*). Two complex conjugate eigenvalues are always extracted together so that *wind* is at least 2 and $def = 1$, for example, means deflation using one real eigenvalue or two complex conjugate eigenvalues.

A more sophisticated implementation is currently under development in which a cost function is developed which allows for automatic and adaptive deflation. This cost function can be interrogated to see at which points in the iteration process the dimension of Z can be increased. However, this approach is not reported on in this paper.

Of course, this deflation method requires slightly more memory than the underlying splitting scheme. The memory overhead is to store the basis Z and the window S. Typically, as will be seen in section 4, Z contains 30 to 50 vectors and S contains less than 4 vectors.

The following represents an outline of the deflation algorithm. It is written here for the Jacobi splitting and the RGS coupling.

Algorithm Jacobi–RGS–deflation

```
* Jacobi splitting *
    $M = \mathrm{diag}(A), N = M - A, H = M^{-1}N, c = M^{-1}b$
    * First : usual Jacobi iterations *
    choose some $y^{(0)}$
    **do** $k$=0,$freq$−1
        $y^{(k+1)} = c + Hy^{(k)}$
        $\Delta^{(k+1)} = y^{(k+1)} - y^{(k)}$
    **enddo**
    $Z = \emptyset$
    test of convergence
    * deflated iterations until convergence *
    * initialization *
    $y^{(0)} = y^{(freq)}$
    $u^{(0)} = Z^T y^{(0)}$
    $q^{(0)} = y^{(0)} - Zu^{(0)}$
    $t^{(0)} = c + Hq^{(0)}$
    $k = 0$
    **while** not converged
        * extract Schur vectors if $Z$ is not maximal *
        * and if $\mathrm{mod}(k, freq) = 0$ *
        if $size(Z) < numeig$
            $S = \{\Delta^{(freq)}, \ldots, \Delta^{(freq-wind+1)}\}$
            orthogonalize $S$
            compute $S^T HS$
            compute $def$ Schur vectors $T$
            $Z_1 = ST$
            $Z = (Z, Z_1)$
            orthogonalize $Z$
            $W = I - Z^T HZ$
        **endif**
        * fixed-point deflated iterations (RGS coupling) *
        $q^{(k+1)} = (I - ZZ^T)(t^{(k)} + (HZ)u^{(k)})$
        $t^{(k+1)} = c + Hq^{(k+1)}$
        $u^{(k+1)} = W^{-1}Z^T t^{(k+1)}$
        $\Delta^{(k+1)} = q^{(k+1)} - q^{(k)}$
        $y^{(k+1)} = Zu^{(k+1)} + q^{(k+1)}$
        $k = k + 1$
        test of convergence
    **endwhile**
```

4. Numerical results and discussions. In this section a number of results are presented to show the efficiency of the deflation process previously outlined. In particular, four different problems are chosen for which extensive results are presented.

We will use a Matlab implementation on a Sparc10 workstation purely to compare the number of iterations needed to obtain suitable convergence for various numerical methods. In order to compare computational performance we will use one CPU of a two processor YMP-2D located at the University of Queensland, which has a peak performance of approximately 330 Mflops on each processor, using Fortran77.

Problems 1 and 2 are solved to very high precision and then we use the "exact" solutions to build a stopping criterion based on the relative error in the solution. Of course, this is not a practical test. The stopping criterion could be based as often on the residual

$$\|r^{(k)}\| \leq tol,$$

but it would require us to compute r, involving another matrix–vector product. We chose the classical test with splitting methods which is based on the difference of two successive iterates

$$\|y^{(k+1)} - y^{(k)}\| \leq tol,$$

where *tol* is a user-defined tolerance parameter. This is implemented for Problems 3 and 4 except when comparing the method with other iterative schemes. In order to have a fair basis of comparison, we then used in all the methods the same stopping criterion based on the residual.

Problem 1 arises 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 symmetric, sparse, and 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 GS method will converge for this problem and that the spectral norms of the iteration matrices are, respectively,

$$\rho(H_J) = \cos\frac{\pi}{N+1}, \quad \rho(H_G) = \left(\cos\frac{\pi}{N+1}\right)^2.$$

Problem 2 is artificially constructed so that the iteration matrix H associated with Jacobi splitting has evenly distributed eigenvalues. We choose $H = V^{-1}DV$, where V is a random matrix (using the function rand in Matlab) and D is the diagonal matrix defined by $D_i = i/(n+1)$. Then we choose $M = D^{-1}$ so that $N = M * J$ and $A = M - N$. We did other experiments with different eigenvalue distributions (other matrices D) with similar results. Here the matrix A is stored as a dense matrix.

Problem 3 is again an artificial problem in which A is a dense and symmetric positive definite matrix with randomly generated elements in [0,1] (using a random number generator from public domain). On the diagonal the value $(i+1)/2 + d$ is added to the random A_{ii} element. The conditioning can be controlled by varying the parameter d. Two values of d will be considered, namely,

$$d = 2.0, \quad d = -0.149,$$

which correspond to the condition number of approximately 400 and 450,000, respectively.

Problem 4 arises from the fitting of surfaces to rainfall data obtained from a set of sparsely scattered meteorological stations in Queensland, Australia [1], [9]. The matrix A is a dense symmetric positive definite matrix of the form $A = A_0 + \lambda I$. Here $\lambda > 0$ is a surface fitting parameter which is minimized within a cross-validation algorithm (see [9]), 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.

The problem size can vary from a few hundred to almost 10,000. Here just three test sets are chosen of dimension 550, 1076, and 1500 because of memory limitations on the Cray YMP-2D.

Problem 1. For this problem we investigate how the convergence depends on the number of eigenvalues *numeig* and the frequency $freq$ with which the eigenvalues are deflated. It is assumed that at most two eigenvalues are deflated at any given time ($wind = 2, def = 1$).

The results given in the next four tables represent the number of iterations needed to achieve convergence for a problem of dimension $m = 144$ and a tolerance of 10^{-10} (for the error in the solution).

The results in Tables 1–3 correspond to Jacobi, GS, and RGS couplings, respectively, given that the underlying splitting (defined by the matrix M) is the Jacobi method. Table 4 was calculated using the same three deflation techniques but with an underlying GS splitting and with one or two eigenvalues being deflated every 15 iterations ($freq = 15$).

The number of iterations required to obtain convergence for the unaccelerated Jacobi and GS, respectively, are 772 iterations and 389 iterations.

From Tables 1–3 it can be seen that only about eight eigenvalues need to be deflated at a frequency of one every 10 iterations to reduce the number of iterations by a factor of 10 (from 772 to 71). It can also be seen that there are some performance differences between the three couplings especially if eigenvalues are deflated too frequently but that as the frequency becomes longer there is very little difference between couplings, which is to be expected from the theoretical results given in section 3. However, of these three couplings the GS and RGS couplings appear to be the most robust when Jacobi splitting is used.

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.

There is a considerable difference in terms of iteration count between using an underlying Jacobi splitting compared with GS as can be seen from comparing Table 4 with Tables 1–3. In particular, if only a small number of eigenvalues are deflated, GS splitting appears to be much more efficient in terms of the number of iterations than Jacobi (by at least a factor of two). As the number of eigenvalues that are deflated increases this ratio between the two schemes appears to approach about two which is the situation when no deflation is used for this problem.

Of course a reduction in the number of iterations by a large factor does not in itself necessarily imply a similar reduction in time because of the additional computational overheads imposed by the deflation process.

In order to see the performance of the deflation process on a larger problem the dimension of the heat equation problem was increased to 900 and a tolerance of 10^{-8} used as a relative error convergence test. A RGS coupling was used with an underlying Jacobi splitting. Unaccelerated Jacobi took 3519 iterations to converge to the same

Table 1
Problem 1 ($m = 144$)*—Number of iterations.*

Jacobi splitting—Jacobi coupling—*wind*=2—*def*=1										
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
Problem 1 ($m = 144$)*—Number of iterations.*

Jacobi splitting—GS coupling—*wind*=2—*def*=1										
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

Table 3
Problem 1 ($m = 144$)*—Number of iterations.*

Jacobi splitting—RGS coupling—*wind*=2—*def*=1										
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
Problem 1 ($m = 144$)*—Number of iterations.*

GS splitting—*freq* = 15—*wind*=2—*def*=1										
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

Table 5
Problem 1 ($m = 900$)*—Number of iterations and Matlab time.*

Jacobi splitting—RGS coupling—*wind*=2—*def*=1—*numeig*=*m*
#eig = final no. of eigenvalues—#iter = no. of iterations
time = CPU time given by Matlab (seconds)									

freq	5	10	15	20	25	30	35	40	45
#eig	52	33	25	21	19	17	15	15	13
#iter	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

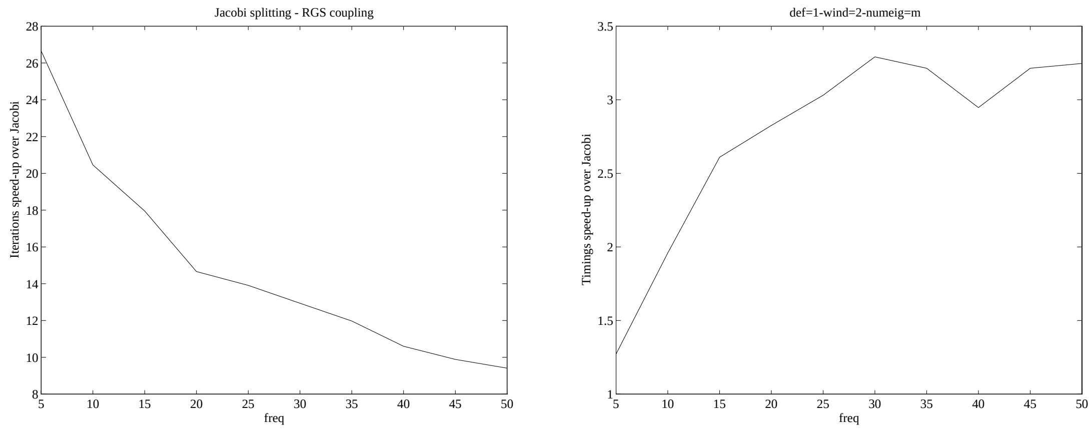

FIG. 1. *Problem* 1 ($m = 900$)*—Matlab speedups.*

tolerance in 95.79 seconds on a Sparc10 workstation. The deflation results are given in Table 5 and speedups in time and iteration over unaccelerated Jacobi are given in Figure 1.

Here speedups in time close to 3.5 are achieved, but again this speedup is underestimated due to the way Matlab is implemented and also due to the sparse matrix representations. In fact Matlab is not an appropriate vehicle for comparing times of different codes as there are high overheads associated with loop structures within Matlab. Thus for Problems 3 and 4 computational performance will be compared using Fortran77 on a Cray YMP-2D.

Problem 2. For this problem a dimension of 200 was chosen and an error in the solution less than 10^{-8} was required. RGS coupling was used with an underlying Jacobi iteration. As many eigenvalues as necessary ($numeig = m$) are extracted in order to attain convergence with either 1 or 2 eigenvalues ($def = 1, wind = 2$) being extracted every $freq$ iterations. The actual number of eigenvectors stored is given by #eig in Table 6.

The unaccelerated Jacobi method took 4208 iterations and a time of 102.3 seconds to attain convergence. All calculations were again done in Matlab. Results are given in Table 6.

The speedups in timing and iteration are presented in Figure 2. In this case a speedup in time of about 8 is much better than for Problem 1. One reason for this is that the iteration matrix H is a dense matrix whereas for Problem 1 it is a sparse matrix. The overhead due to deflation relative to the cost of Jacobi is higher in the sparse case, so that the same speedup in iterations leads to a smaller speedup in time. But again timings in Matlab are only an indication of what can be expected.

Problem 3. Here a problem of dimension $m = 1000$ is solved to a tolerance of 10^{-8} for the residual with different conditionings corresponding to different values of d as described above. Results on one processor of a Cray YMP-2D are given in Table 7 for Jacobi with and without deflation, GS with deflation, and conjugate gradient without preconditioning.

The original iterative scheme (in this case Jacobi) was either diverging or had not converged after 500 iterations. Not only did deflation in conjunction with a Jacobi

TABLE 6
Problem 2 ($m = 200$)*—Number of iterations and Matlab time.*

Jacobi splitting—RGS coupling—*def*=1,*wind*=2—*numeig*=*m* #eig = final no. of eigenvalues—#iter = no. of iterations time = CPU time given by Matlab										
freq	2	4	6	8	10	12	14	18	25	50
#eig	160	84	62	50	46	39	37	31	27	17
#iter	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

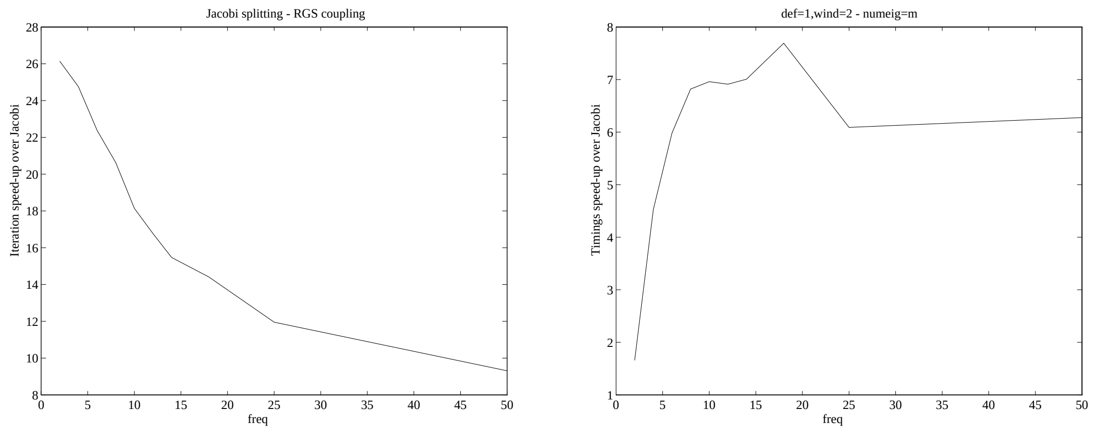

FIG. 2. *Problem* 2 ($m = 200$)*—Matlab speedups.*

iteration cause rapid convergence, but for both values of the parameter d deflation with Jacobi was more effective than conjugate gradient. It should also be noted that deflation with GS needed fewer iterations than deflation with Jacobi to attain the given accuracy. However, the times on a Cray were substantially higher because of the poor vectorizing capabilities of the GS iteration compared with Jacobi.

We had to store $wind = 3$ vectors for the window (the matrix S). The total number of extracted eigenvalues was less than 20, so that we had to store less than 20 vectors for the approximated invariant basis Z.

Problem 4. For this problem, we will compare our deflation scheme with the conjugate gradient iterative method and with the LU direct method. We first experiment with a problem of dimension $m = 1500$ and for $\lambda = 0.173$. In this instance, the underlying iterative scheme is Richardson iteration; the coupling is RGS.

Figure 3 shows a plot of the error in the solution versus the number of iterations. The effects of the periodic eigenvalue extractions can be clearly seen here, as the error in the current iterate reduces noticeably. In this case, three eigenvalues are extracted every eight iterations ($def = 3, wind = 3, freq = 8$).

Table 8 compares the performance on one processor of the Cray YPM-2D of Richardson and Jacobi iterations with and without deflation with a conjugate gradient for the same problem, using RGS coupling and ($def = 3, wind = 3, freq = 5$). The stopping criterion here relies on the residual with a tolerance 10^{-8}. Although

TABLE 7
Problem 3 ($m = 1000$)*—Results on a Cray YMP-2D.*

RGS coupling—*freq* 5—*def* 3—*wind* 3—*numeig* 20				
	$d = 2.0$		$d = -0.149$	
Method	Iterations	Seconds	Iterations	Seconds
Jacobi	∞	-	∞	-
Jacobi (deflation)	21	0.45	36	0.73
GS (deflation)	20	1.92	30	2.60
Conjugate gradient	80	0.69	187	1.57

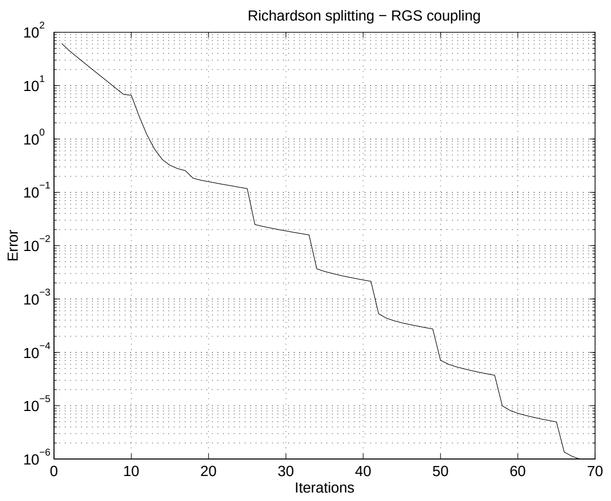

FIG. 3. *Problem* 4 ($m = 1500,\ \lambda = 0.173$)*—Residual.*

the convergence of deflated Richardson in terms of the number of iterations is impressive, it cannot compete with conjugate gradient. However, when the Jacobi iteration is used conjugate gradient and deflation give more or less the same computational performance.

In order to compare the computational performance of deflation with a direct method, three different weather data sets of dimension 550, 1025, and 1500 were solved on one processor Cray YMP-2D. The values for λ were given by

$$\lambda = \frac{1}{5}, \quad \lambda = \frac{5}{64}.$$

In the first case the problem is well conditioned, and in the second case there is a modest ill conditioning, with a condition number for A in the range (160–425) depending on the size of the dimension m.

The deflation implementation was based on RGS coupling with an underlying Jacobi splitting with tolerance 10^{-8} (the stopping criterion here used the difference between two iterates) with one eigenvalue deflated every three iterations ($def = 1, wind = 2, freq = 3$). In all tests, the final size of the basis Z was less than 50. We used Cray library routines for LU factorizations with backward and forward substitution. The speedups over the direct solve are given in Figure 4.

TABLE 8
Problem 4 ($m = 1500,\ \lambda = 0.173$)—*Results on a Cray YMP-2D.*

RGS coupling—*def* 3, *wind* 3, *freq* 5				
Method	Flop count (Millions)	Flop rate (Mflops)	Iterations	Time (CPU seconds)
Richardson	1334	309	294	4.32
Richardson (deflation)	637	286	66	2.23
Jacobi			diverges	
Jacobi (deflation)	131	233	15	0.556
Conjugate gradient	171	303	36	0.57

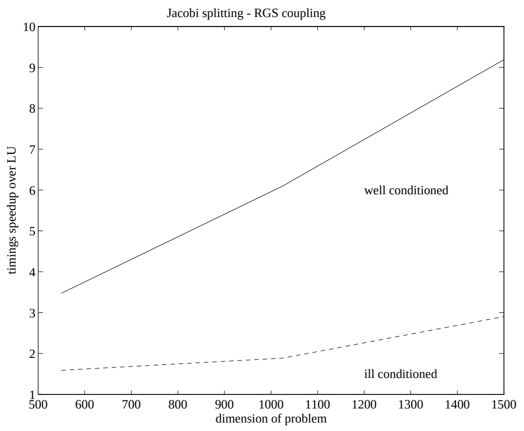

FIG. 4. *Problem* 4—*Speedup over LU.*

5. Deflation applied to a sequence of linear systems. So far we have only considered the solution of single linear systems, but one very important area where deflation can prove extremely effective is when a sequence of equations of the form

$$(A + sI)y = b \tag{6}$$

has to be solved. This can arise in many areas; some of them are quoted below.

1. If an implicit method is used to solve a stiff system of differential equations, the nonlinear implicit equations are solved by some variant of the Newton method. If the Jacobian of the problem is kept constant over a number of steps but the stepsize varies, then a sequence of linear systems of the form given by (6) have to be solved in which only the value of s changes.
2. If the inverse power method is being used to find an approximation to an eigenvalue near a given value, the systems of the form (6) have to be solved at each step.
3. In a surface fitting application which requires a GCV process in order to find a minimization parameter which gives a tradeoff between a least squares fit and a smooth surface, a sequence of systems of the form (6) has to be solved (see [9]).

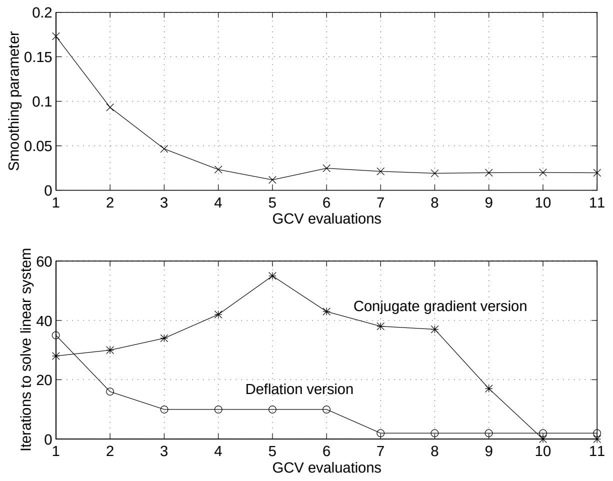

FIG. 5. *Linear solver iterations versus GCV evaluations.*

Thus if systems of the form $Ax = b$ and $(A + sI)x = b$ have to be solved in succession, then the iteration matrices for a Richardson scheme are given by

$$H_1 = I - \frac{1}{\omega_1}A,$$
$$H_2 = \left(1 - \frac{s + \omega_1}{\omega_2}\right) I + \frac{\omega_1}{\omega_2}H_1,$$

and the eigenspaces of these two matrices are clearly related by a very simple equation. This enables deflation with a Richardson iteration to be carried across these systems of equations. This is not the case for more sophisticated iterative schemes such as Jacobi or GS as then there is no particular relation between the eigenspaces of H_1 and H_2. Because direct methods such as LU factorization cannot exploit any structure in this case, deflation with Richardson extrapolation could prove very useful when such sequences of linear systems need to be solved.

Figure 5 compares the performance of a Richardson deflation code with that of the conjugate gradient code in solving a sequence of linear systems of dimension 1500 of the form given by (6) at each GCV function evaluation (this figure only shows the iterations required to solve the first of the two systems that must be solved at each evaluation). In this case RGS coupling is used with $def = 3, wind = 3$.

As can be seen for the first five GCV evaluations, reducing the estimate of the smoothing parameter (which is the parameter added to the diagonal of the coefficient matrix) directly increases the number of conjugate gradient iterations required to solve the system, as a reduction in s causes an increase in the condition number. For the later iterations, a significantly more accurate initial guess can be calculated and this helps the conjugate gradient method's performance. As Figure 5 illustrates, the deflation technique is able to accumulate information about the eigenspace of the systems being solved and becomes distinctly more efficient with each successive system. Indeed as the minimization parameter converges, the number of iterations needed to solve the system is only two or three.

6. Concluding remarks. Usually iterative schemes based on a splitting do not converge very fast. We have designed a new method which boosts these methods by deflating the eigenvalues slowing down the convergence. This method is very general and useful for any large nonsymmetric, dense, or sparse matrix. The deflation provides a distinct advantage for ill-conditioned systems where the underlying iterative scheme would either diverge or converge very slowly. For systems where the underlying scheme is already converging reasonably well, then the accelerated convergence provided by deflation is not worth the extra work required. We did several numerical experiments to demonstrate the efficiency of our method. A suitable conclusion to be drawn from these results is that deflation can be a very robust and efficient procedure when solving large linear systems of equations. It can turn a divergent iterative scheme into a rapidly converging one and can outperform conjugate gradient methods for some problems. In many cases the number of eigenvalues that have to be extracted in order to get good performance is often modest so that the memory requirements are kept small. Whether Krylov subspace methods or deflation techniques are more economical will in general be problem dependent.

Acknowledgements. The authors would like to thank the referees for their very helpful comments in modifying this paper.

REFERENCES

[1] K. Burrage, A. Williams, J. Erhel, and B. Pohl, *The implementation of a generalized cross validation algorithm using deflation techniques for the implementation of a generalized cross validation algorithm using deflation techniques for linear systems*, J. Appl. Numer. Math., 9 (1995), pp. 17–31.

[2] I. Duff, A. Erisman, and J. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, London, UK, 1988.

[3] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.

[4] H. Jarausch, *Analyzing Stationary and Periodic Solutions of Systems of Parabolic Partial Differential Equations by Using Singular Subspaces as Reduced Basis*, Tech. Report 92, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 1993.

[5] H. Jarausch and W. Mackens, *Numerical treatment of bifurcation problems by adaptive condensation*, in Numerical Methods for Bifurcation Problems, W. Kupper and H. Mittelmann, ed., Birkhauser, Boston, 1987.

[6] Y. Saad and M. H. Shultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM, J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

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

[8] H. van der Vorst and C. Vuik, *The superlinear convergence behaviour of GMRES*, J. Comput. Appl. Math., 48 (1993), pp. 327–341.

[9] A. Williams and K. Burrage, *Surface fitting using GCV smoothing splines on supercomputers*, in Proc. of Supercomputing, San Diego, 1995.