RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Projection Methods for the Numerical Solution of Markov Chain Models

Youcef Saad

October 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.40

NASA Cooperative Agreement Number NCC2-387

Projection Methods for the Numerical Solution of Markov Chain Models

Youcef Saad

October 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.40

NASA Cooperative Agreement Number NCC2-387

Projection Methods for the Numerical Solution of Markov Chain Models

Youcef Saad

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.40
October 1989

This paper gives an overview of projection methods for computing stationary probability distributions for Markov chain models. A general projection method is a method which seeks an approximation from a subspace of small dimension to the original problem. Thus, the original matrix problem of size N is approximated by one of dimension m, typically much smaller than N. A particularly successful class of methods based on this principle is that of Krylov subspace methods which utilize subspaces of the form $span\{v, av, ...A^{m-1}v\}$. These methods are effective in solving linear systems (Conjugate Gradients, GMRES, ...) and eigenvalue problems (Lanczos, Arnoldi, ...) as well as nonlinear equations. They can be combined with more traditional iterative methods such as SOR, SSOR, or with incomplete factorization methods to enhance convergence.

Work reported herein supported by Cooperative Agreement NCC2-387 between the National Aeronautics Space Administration (NASA and the Universities Space Research Association (USRA).

1 Introduction

In Markov chain modeling, one often seeks the stationary probability distributions of a system that can occupy a number of N different states. If we number these states from 1 to N, and call π_i the probability of the system to be in state i at equilibrium, i.e., in the long run, then the basic equation to compute these probabilities is

$$\pi P = \pi \tag{1}$$

where $\pi = [\pi_1, \pi_2, ..., \pi_N]$ is the row vector comprised of the stationary probabilities, and P is the matrix of transition probabilities: p_{ij} is the probability that the system switches from state i to state j for $i \neq j$ and $p_{ii} = 1 - \sum_{j \neq i} p_{ij}$. The system (1) can be viewed as an eigenvalue problem, or more precisely an eigenvector problem, since the eigenvalue is known to be unity. In fact the right eigenvector is known and we seek the left eigenvector. Alternatively, defining

$$Q = I - P \tag{2}$$

we may rewrite (1) as

$$\pi Q = 0 \quad \text{or} \quad Q^T \pi^T = 0 \tag{3}$$

which is a homogeneous linear system to solve. The reason why we distinguish between these two view-points is that there are methods that are well-known for linear systems but have no equivalent for eigenvalue problems and vice versa. We can choose effective algorithms from both camps. For example, subspace iteration which is one of the methods used in Markov chain modeling, is not directly applicable for solving linear systems.

When the number of states is small then there are a number of reliable techniques that can be used to solve (1), for example an inverse iteration approach based on Gaussian elimination. The difficulty is that in practice the number of states for realistic systems can be enormous and the cost and storage of the standard methods becomes prohibitive. The main philosophy of projection methods is to avoid costly manipulations on the original matrix. Rather, the main operations performed with the matrix are matrix by vector multiplications. The matrix P is usually very sparse so that storage is not a big burden and matrix by vector multiplications are inexpensive.

The organization of the paper is as follows. We start by describing general projection processes for both eigenvalue problems and linear systems in Section 2. In Section 3 we take on the eigenvalue point of view and describe two techniques based on this approach. The linear systems point of view will then be described in Section 4 with some specific approaches. Some numerical tests will then be reported.

2 Projection methods

2.1 Projection techniques for solving linear systems

We start with the linear system of linear equations,

$$Ax = b \tag{4}$$

where A is a large sparse nonsymmetric matrix. A projection process to solve (4) is a technique that computes an approximation to (4) from a subspace of dimension m by enforcing some orthogonality condition on the residual vector $r = b - Ax$. More precisely, let K be a subspace of dimension m from which we seek the approximation to x and let L be another subspace of dimension m which will define the orthogonality conditions. We define the approximate solution by writing that

$$\tilde{x} \in K \tag{5}$$
$$b - A\tilde{x} \perp L \tag{6}$$

Since the approximation x lies in a subspace of dimension m and (6) imposes m conditions, there will in general, but not always, exist a unique solution to the above problem, but we will come back to this problem later.

The approach just described is known as the Petrov-Galerkin projection method. A particular case of importance is when $L = K$. Then the method is called an orthogonal projection method or a Galerkin method.

Let P be the orthogonal projector onto K: for any x, Px is uniquely defined by $Px \in K, (I - P)x \perp K$. Similarly, let Q be the (oblique) projector onto K and orthogonal to L: for any x, Qx is uniquely defined by $Qx \in K, (I - Q)x \perp L$. For this projector to be defined it is assumed that no vector of K is orthogonal to L, or equivalently, that no vector of L is orthogonal to K. Then the Petrov-Galerkin method consists of replacing the original problem by the problem of finding $\tilde{x}$ satisfying the equations $P\tilde{x} = \tilde{x}$, and $Q(b - A\tilde{x}) = 0$. In other words, $\tilde{x}$ is a nontrivial solution of

$$Q(b - AP\tilde{x}) = 0 \tag{7}$$

which takes the form

$$\tilde{A}\tilde{x} = Qb \tag{8}$$

where we have set $\tilde{A} = QAP$.

The question that arises next is: given the two subspaces K and L, how accurate should we expect the approximate solution to be? It is in general difficult to answer this question. The element of K that is the closest to the exact solution x^* in the 2-norm sense is clearly Px^*. So the error in the 2-norm sense must be larger than $\|(I - P)x^*\|_2$. This is an upper bound but we wish to find a lower bound. Typically one would like to establish an upper bound on the residual norm $b - A\tilde{x}$ in terms of, for example, $(I - P)x^*$. We know of no simple error bounds of this type for general projection methods in the nonhermitian case. However, we can easily establish an a-priori residual bound for the approximate problem. More precisely,

Theorem 2.1 *Let $\gamma = \|QA(I - P)\|_2$. The exact solution x^* satisfies the following residual condition with respect to the approximate problem (8):*

$$\|Qb - \tilde{A}x^*\|_2 \leq \gamma \|(I - P)x^*\|_2 \tag{9}$$

Proof : We have

$$Qb - \tilde{A}x^* = QAx^* - QAPx^* = QA(I-P)x^* \tag{10}$$

Since $I-P$ is a projector this gives

$$Qb - \tilde{A}x^* = QA(I-P)(I-P)x^* \tag{11}$$

which immediately yields the result. □

In many of the projection methods, b is a vector of K so that we have $Qb = b$. This theorem shows that when the distance between the exact solution x^* and the subspace K is small then a good approximation can be obtained provided that the norm of Q is not too large and that the projected problem is not too poorly conditioned. In the orthogonal projection case, $Q = P$ and we get $\gamma \leq \|A\|_2$.

In practice, one needs to work with some convenient basis of the subspace K. Given a basis $V = [v_1, v_2, ..., v_m]$ of K and a basis $W = [w_1, w_2, ..., w_m]$ of L, and writing $\tilde{x} = V_m y$ where y is in $\mathbb{R}^m$, we see immediately that y must satisfy

$$W_m^T(AV_m y - b) = 0 \tag{12}$$

or

$$Hy = W_m^T b \tag{13}$$

where we have set $H = W_m^T A V_m$.

There are two important particular cases for L. The case when $L = K$, we have already mentioned. The second case of importance is when $L = AK$ because it is equivalent to minimizing the residual norm on the subspace K. More precisely, we have the following theorem, see for example [16].

Theorem 2.2 *When $L = AK$ then the approximate solution $\tilde{x}$ minimizes the residual norm over the subspace K.*

We refer to this class of projection methods as the class of minimal residual methods. An example of such a technique will be described later in Section 3.

It is important to consider the effect of using a nonzero initial guess for approximating the solution. If we had a guess x_0 to the solution of (4) we would attempt to find a vector δ in K so that the update $x_0 + \delta$ satisfies the usual Petrov-Galerkin conditions. In other words, we would write

$$b - A(x_0 + \delta) \perp L \quad \text{or} \quad r_0 - A\delta \perp L \tag{14}$$

with the usual notation $r_0 = b - Ax_0$.

In Markov chain modeling, the matrix A is singular and $b = 0$ and we would like to show how to adapt the methods just described to this situation. A direct application of the previous principles would lead to a projected problem of the form

$$Hy = 0. \tag{15}$$

Unfortunately, there is no reason for H to be a singular matrix and as a result the only solution to (15) would be $y = 0$ in general. The difficulty does not arise if we take the nonzero initial guess formulation discussed above. Then the projected problem becomes

$$Hy = Q^T r_0 \tag{16}$$

in which $r_0 = b - Ax_0 = -Ax_0$. Notice that if the matrix H happens to be singular, then one can take the solution of (16) to be an eigenvector of the matrix H associated with the eigenvalue zero. With this modification one can say that the projected problem always has a solution. The only possible concern might be the practicality of the decision to declare a matrix singular. The real question is whether or not a nearly singular matrix could lead to a substantially different result than would be obtained by considering it to be singular. However, if the matrix is nearly singular then the solution of (16) is equivalent to a form of inverse iteration and the result, after normalization, should be an accurate approximation to the exact eigenvector associated with the eigenvalue zero.

For the minimal residual methods referred to earlier there is no difficulty in defining the approximate solution in the formulation where $x_0 \neq 0$. This is because the approximate solution minimizes the 2-norm of $r_0 - A\delta$ over $\delta \in K$ and this optimization problem always has a unique solution.

2.2 Projection techniques for solving eigenvalue problems

We now consider the standard eigenvalue problem,

$$Ax = \lambda x \tag{17}$$

Similarly to the previous section, we assume that we are given two subspaces K and L, and we seek an approximate eigenvalue $\tilde{\lambda} \in C$ and an approximate eigenvector $\tilde{x}$ from the subspace K, by imposing the Petrov-Galerkin condition

$$A\tilde{x} - \tilde{\lambda}\tilde{x} \perp L \tag{18}$$

Similarly to the previous section, if P is the orthogonal projector onto K and Q is the oblique projector onto K orthogonally to L, then the original problem is replaced by the projected problem

$$\tilde{A}\tilde{x} = \tilde{\lambda}\tilde{x} \tag{19}$$

and we can show a theorem similar to the main theorem of the previous section, namely

Theorem 2.3 *Let $\gamma = \|QA(I-P)\|_2$. The exact eigenvector x associated with the exact eigenvalue λ satisfies the following residual condition with respect to the approximate problem (16),*

$$\|(\tilde{A} - \lambda I)x\|_2 \leq \sqrt{\gamma^2 + |\lambda|^2}\|(I-P)x\|_2 \tag{20}$$

The proof of this result is similar to the one shown in the previous section for linear systems and can be found in [13].

3 Methods for eigenvalue problems

We consider two sample eigenvalue methods used in Markov chain modeling. The first is a technique based on the subspace iteration method of Bauer [3]. It has been used with good success for Markov chain modeling [7, 18]. The second is a method due to Arnoldi [1] in 1951.

3.1 Subspace Iteration

One of the simplest methods for computing invariant subspaces is the so-called subspace iteration [7, 18] methods well-known to the structural engineers. In its simplest form, the method is equivalent to a projection method onto the subspace $K = span\{A^m V_0\}$ where V_0 is an initial set of p columns, i.e., an $N \times p$ matrix. Note that the dimension of the subspace K is constant. One of the forms of the subspace iteration algorithm can be described as follows.

1. Choose an initial orthonormal system $V_0 \equiv [v_1, v_2, \ldots, v_m]$ and an integer k;
2. Compute $X = A^k V_0$ and orthonormalize X to get V.
3. Perform a projection process with V, i.e., compute the eigenvalues and the matrix U of eigenvectors of the matrix $C = V^T A V$.
4. Test for convergence. If satisfied then exit else continue.
5. Take $V_0 = VU$, the set of approximate eigenvectors choose, a new k and go to 2.

A well-known alternative consists of replacing the set of eigenvectors U in steps 3 and 5 by the Schur vectors of the matrix C, i.e., the column vectors of the matrix that transforms C in upper quasi-triangular form [19, 15]

The above algorithm utilizes the matrix A only to compute successive matrix by vector products $w = Av$, so sparsity can be exploited. However, it faces the drawback that it is generally a slow method.

Often, Chebyshev iteration is used to accelerate convergence: step 2 is replaced by $X = t_k(A)V_0$, where t_k is obtained from the Chebyshev polynomial of the first kind, of degree k, by a linear change of variables. The three-term recurrence of Chebyshev polynomials allows to compute a vector $w = t_k(A)v$ at almost the same cost as $A^k v$. Moreover, it is then possible to compute the rightmost (or leftmost) eigenvalues of A. Also, performance is improved as this is the usual primary reason for using Chebyshev iteration. Details on implementation can be found in [14].

Subspace iteration as described above is seldom competitive with methods that use a combination of preconditioning and Krylov subspace methods described later. If we use a block-size of p, then the convergence rate for the eigenvalue $\lambda_1 = 1$, is of the order of $|\lambda_{p+1}|$, if we order the eigenvalues by decreasing order of magnitude.

3.2 Arnoldi's method

A second method used in the literature is the Arnoldi process [1, 11] which is a projection process onto the so-called Krylov subspace

$$K_m = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}. \tag{21}$$

The algorithm starts with some nonzero vector v_1 and generates the sequence of vectors v_i from the following algorithm,

Algorithm: Arnoldi

1. *Initialize:*

   Choose an initial vector v_1 of norm unity.

2. *Iterate:* Do $j = 1, 2, ..., m$

   1. Compute $w := Av_j$
   2. Compute a set of j coefficients h_{ij} so that

$$w := w - \sum_{i=1}^{j} h_{ij} v_i \tag{22}$$

      is orthogonal to all previous v_i's.

   3. Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$.

By construction, the above algorithm produces an orthonormal basis of the Krylov subspace $K_m = span\{v_1, Av_1, \ldots, A^{m-1}v_1\}$. The $m \times m$ upper Hessenberg matrix H_m, consisting of the coefficients h_{ij} computed by the algorithm, represents the restriction of the linear transformation A to the subspace K_m, with respect to this basis, i.e., we have

$$H_m = V_m^T A V_m, \tag{23}$$

where $V_m = [v_1, v_2, \ldots, v_m]$. Approximations to some of the eigenvalues of A can be obtained from the eigenvalues of H_m. This is Arnoldi's method in its simplest form.

Note the useful relation for later use,

$$AV_m = V_{m+1}\bar{H}_m \tag{24}$$

$$= V_m H_m + h_{m+1,m} v_{m+1} e_m^T \tag{25}$$

where $\bar{H}_m$ is the $(m+1) \times m$ upper Hessenberg matrix whose nonzero elements are the h_{ij} defined in the above algorithm. In other words $\bar{H}_m$ is obtained from H_m by appending the row $[0, 0, \ldots, 0, h_{m+1,m}]$ to it.

As m increases, the eigenvalues of H_m that are located in the outermost part of the spectrum start converging towards corresponding eigenvalues of A. In practice, however, one difficulty with the above algorithm is that as m increases cost and storage increase rapidly. One solution is to use the method iteratively: m is fixed and the initial vector v_1 is taken at each new iteration as a linear combination of some of the approximate eigenvectors. Moreover, there are several ways of accelerating convergence by preprocessing v_1 by a Chebyshev iteration before restarting, i.e., by taking $v_1 = t_k(A)z$ where z is again a linear combination of eigenvectors.

A technique related to Arnoldi's method is the nonsymmetric Lanczos algorithm [9, 4] which delivers a nonsymmetric tridiagonal matrix instead of a Hessenberg matrix. Unlike Arnoldi's process, this method requires multiplications by both A and A^T at every step. On the other hand it has the big advantage of requiring little storage (5 vectors). Although no comparisons of the performances of the Lanczos and the Arnoldi type algorithms have been made, the Lanczos methods are usually recommended whenever the number of eigenvalues to be computed is large.

4 Methods for linear systems

4.1 FOM and GMRES

In this section we take the point of view that we want to solve the linear system $Ax = b$. We start by assuming that the system is nonsingular. The methods to be described here are based on Krylov subspaces. Assume that we have an initial guess x_0 with residual vector $r_0 = b - Ax_0$. We will take

$$v_1 = r_0/\beta, \quad \text{where} \quad \beta \equiv \|r_0\|_2 \tag{26}$$

and run m steps of Arnoldi's methods starting with the vector v_1 thus defined. To apply a Galerkin projection process onto the subspace K_m, we need to seek a vector $\delta_m \in K_m$ that satisfies the conditions (14) with $L = K$. Writing $\delta_m = V_m y_m$ and the orthogonality condition

$$V_m^T(r_0 - Av_m y_m) = 0$$

yields immediately

$$y_m = H_m^{-1}\beta e_1 \tag{27}$$

where we have set $\beta \equiv \|r_0\|_2$ and used the orthogonality of V_m and the relation (23). Thus the approximate solution takes the form

$$x_m = x_0 + V_m H_m^{-1}\beta e_1 \tag{28}$$

The method described above which consists of generating the Arnoldi basis V_m with v_1 defined by (26) and the approximate solution via (28) is referred to as the Arnoldi process for linear systems or "Full Orthogonalization Method" [12]. A detail that is important for

the implementation is that the residual norm of the approximate solution can be determined, without explicitly computing the solution, via the formula

$$\|b - Ax_m\|_2 = |h_{m+1,m} e_m^T y_m|. \tag{29}$$

which is a direct consequence of (25).

Although not proved rigorously, it is usually observed that as m increases, the approximate solution x_m rapidly approaches the exact solution. Ideally, one would like to use a large enough m that x_m is as close as desired to the solution. However, this is not feasible in practice because of the rapid increase in the storage requirement as the dimension m of the subspace increases. Therefore, the basic idea described above is usually implemented with restarting. The dimension m is fixed, and the method is after each outer loop consisting of the process just described the initial guess x_0 is reset to be equal to x_m and the process is restarted until convergence is achieved.

One drawback of the above algorithm is that from the theoretical point of view there is little known concerning convergence. For this reason, several authors have instead turned to methods with optimal properties: for example one may seek a method that minimizes the residual norm over the whole subspace K. As was seen before, this is realized by taking $L = AK$. One such version is the GMRES algorithm [17]. In GMRES, one seeks to minimize the residual norm of the approximate solution in the affine subspace $x_0 + K_m$. This means that the approximate solution $x_0 + V_m y$ must minimize $J(y) = \|b - AV_m y\|_2$ over $y \in \mathbb{R}^m$. Utilizing the relation (24) we get

$$\begin{aligned} J(y) &= \|b - A(x_0 + V_m y)\|_2 = \|r_0 - AV_m y\|_2 \\ &= \|\beta v_1 - V_{m+1}\bar{H}_m y\|_2 = \|V_{m+1}[\beta e_1 - \bar{H}_m y]\|_2 \\ &= \|\beta e_1 - \bar{H}_m y\|_2 \end{aligned}$$

The last equality is a consequence of the orthogonality of the vectors v_i's. Therefore the only difference between this approach and the FOM approach is the way in which the vector y is obtained. In one case it is obtained by solving the $m \times m$ linear system (27) and in the second by solving the least squares problem

$$\text{Find } y_m \text{ solution of: } \min_{y \in \mathbb{R}^m} \|\beta e_1 - \bar{H}_m y|_2 \tag{30}$$

Similarly to the FOM method, there exists a formula that allows to compute the residual norm of x_m without computing x_m. This is based on the trivial equality

$$\|b - Ax_m\|_2 = \min_{y \in \mathbb{R}^m} \|\beta e_1 - \bar{H}_m y\|_2. \tag{31}$$

In a practical implementation of the GMRES algorithm, the Hessenberg matrix is progressively reduced to upper triangular form by using Givens rotations. The same rotations are applied to the right-hand-side βe_1. Because the last row of the resulting matrix is a zero row, the above minimum can be seen to be equal to the bottom element of the right-hand-side

after these rotations. This provides the residual norm of the current iterate for free without having to compute the iterate itself.

In summary we can put the two methods within the same framework as follows.

Algorithm : FOM / GMRES

1. *Start:* Choose x_0.

2. *Arnoldi process:*

   - Compute $r_0 := b - Ax_0$;
   - Compute $\beta = \|r_0\|_2$; and $v_1 = r_0/\beta$.
   - Arnoldi process. For $j = 1, 2, \cdots$, do:

     (a) Form Av_j and orthogonalize it against the previous $v_1, \cdots, v_j$ via

$$\begin{aligned} h_{i,j} &= (Av_j, v_i), \quad i = 1, 2, \cdots, j, \\ \hat{v}_{j+1} &= Av_j - \sum_{i=1}^{j} h_{i,j} v_i \\ h_{j+1,j} &= \|\hat{v}_{j+1}\|_2, \quad \text{and} \\ v_{j+1} &= \hat{v}_{j+1}/h_{j+1,j}. \end{aligned} \tag{32}$$

     (b) Compute the residual norm $\rho_j = \|b - Ax_j\|_2$, of the solution x_j via the formula (29) and (31).

     (c) If $\rho_j \leq \epsilon$ set $m = j$ and go to (3).

3. *Form the approximate solution:*

   Define $\bar{H}_m$ to be the $(m+1) \times m$ (Hessenberg) matrix whose nonzero entries are the coefficients h_{ij}, $1 \leq i \leq j+1$, $1 \leq j \leq m$, and H_m the $m \times m$ submatrix obtained from $\bar{H}_m$ by removing its last row. Let $V_m \equiv [v_1, v_2, \cdots, v_m]$, and where $e_1 = [1, 0, \ldots, 0]^T$. Then:

   FOM:

   - Compute $x_m = x_0 + \beta V_m H_m^{-1} e_1$.

   GMRES:

   - Find the vector y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$, over all vectors y in $\mathbb{R}^m$.
   - Compute $x_m = x_0 + V_m y_m$,.

4. *Stopping test:* If x_m is determined to be a good enough approximate solution to (4), then stop, else set $x_0 := x_m$ and go to 2.

4.2 Preconditionings

Often the projection methods themselves are not sufficient to achieve good performance and preconditioning is necessary. Typical preconditioning techniques consist of approximating the original matrix by a 'close by' matrix M and then solving a preconditioned system such as

$$M^{-1}Ax = M^{-1}b \tag{33}$$

by some suitable iterative method. For this approach to be practical it is necessary that the linear system solution with the matrix M be inexpensive and easy to implement. In the core of the Krylov subspace algorithm that is used, the matrix by vector product is replaced by a matrix vector product followed by a linear system solve with the matrix M. Thus the modifications to the algorithm FOM/ GMRES displayed above is simply to replace all occurrences of A by $M^{-1}A$, and the right hand side b by $M^{-1}b$. One can also precondition by solving the system

$$AM^{-1}z = b \tag{34}$$

whose solution z is related to the solution of the original system by $x = M^{-1}z$. There are no a-priori reasons for using this right-preconditioning approach rather than the left preconditioning approach, except that an approximate solution $\tilde{z}$ for the system (34) has the same residual vector as for the original system.

The simplest preconditioning technique is the incomplete LU factorization which consists of factoring A as

$$A = LU + E \tag{35}$$

where the matrix LU matches A everywhere where there are nonzero elements and E is a remainder. The matrix $M = LU$ is then used as a preconditioning matrix. Note that the exact LU factorization of A would require far more computation and storage than the incomplete LU factorization whose cost is typically of the order of NZ, the number of nonzero elements of A. The simplest form of incomplete factorization is one in which the L matrix has the same structure as the lower part of A and the U has the same structure as the upper triangular part of A. This is referred to as the ILU(0) preconditioning to account for the fact that no fill-in is allowed.

The incomplete LU factorization outlined above exists for M matrices and can be computed by a very simple procedure which consists of performing the LU factorization and replacing by zero any nonzero elements that is introduced outside of the nonzero structure of A, during the process.

We are interested in the incomplete factorization of the matrix $A = Q^T$ but since this matrix is singular the classical results on the existence of the ILU factorization [8] do not readily apply to this case. However, these classical results can be trivially extended to such matrices by, for example, adapting the results in [2], page 42.

The quality of the ILU(0) preconditioning can be improved in several ways by allowing more fill-in. A notion that is used in this regard is that of level of fill-in: initially all elements have level of fill-in equal to zero. Thereafter, at each step the level of fill-in of an element is updated by adding one to the sum of the levels of fill-ins of its parents in L and U. Here the

parent-child relation corresponds to the creation of a fill-in element by the basic operation in Gaussian elimination. ILU(k) will then correspond to dropping all elements whose level of fill-in exceeds k. Thus ILU(0) is the usual incomplete LU factorization with no fill-in.

As can be seen this incomplete factorization technique relies entirely on the structure of the matrix and not at all on the actual values of its elements. For a large class of problems that come from PDE's this is usually sufficient because of the fact that these matrices are M-matrices. On the other hand, the situation may be different for other classes of problems.

An alternative used is to drop the elements during the ILU factorization according to their magnitude rather than their position. Several such techniques exist [6, 20]. Some are implemented in the context of direct solvers such as in MA28 [5] and in Y12M [20]. Here a rather accurate incomplete factorization is usually performed and a simple technique such as iterative refinement is used as an iterative procedure. In the context of iterative solvers, one can simplify the factorization process enormously [6, 10]. One drawback of this approach is that it is rather difficult to predict the storage that will be necessary during the factorization. A second alternative would be to only keep a given number of elements per row during the incomplete factorization. Two techniques based on the two approaches outlined above have been implemented and tested on realistic Markov chain problems in [10].

In addition to incomplete factorization preconditionings one can also use the more traditional relaxation methods such as the SOR, or SSOR iteration, as preconditioners. Our experience in [10] with these techniques on real Markov chain problems is that they are not as efficient as the ILU type preconditioners. For further details see [10].

5 Numerical tests

The following illustration is taken from [10]. It compares a few of the methods described in this paper and includes the power method referred to here as the fixed point iteration and a direct solver (GE).

The example deals with the system architecture of a time shared, multiprogrammed, paged, virtual memory computer. The system is composed of a set of N terminals, a central processing unit, a secondary memory, and a filing device. This real-life example models requests to each of the devices of the system that are queued and scheduled on a first come first served basis. For further details see [10]. The matrix Q obtained here is of dimension 1771 with 11,011 nonzero elements.

The following table shows various statistics associated with the performance of several methods for solving the matrix equation $Q^T x = 0$. The direct solver called GE is without pivoting and without any reordering. All runs have been made on a Ardent Titan computer in double precision. The compiler option used was -O3. but no additional optimization of the code was performed.

Method	Method Parameters	Total Time	Set-up Time	Iter. Time	Flops	Additional Memory	Iters	Residual Norm
ARNOLDI	m=10	51.4			32.9	17,971	*1010	0.121E-04
	m=20	46.5			51.5	36,341	*1020	0.131E-03
	m=25	83.1			96.8	45,676	*1625	0.184E-03
PCARN/								
+ILU0	m=5	19.8	0.3	19.3	4.8	23,408	160	0.324E-10
	m=10	15.7	0.3	15.2	5.6	32,263	150	0.811E-10
+ILUK	m=10, K=5	9.1	3.3	5.6	2.5	30,050	70	0.291E-10
	m=10, K=10,	5.9	4.3	1.4	0.5	38,735	10	0.409E-11
+ILUTH	m=10, $\tau = .01$	15.1	1.6	13.4	7.1	26,104	230	0.543E-10
	m=10, $\tau = .001$	11.6	1.8	9.7	3.0	32,142	80	0.205E-10
GMRES /								
+ILU0	m=5	149.1	0.3	148.6		23,408	*1600	0.210E-06
	m=10	16.4	0.3	15.9	5.2	32,263	140	0.632E-10
	m=20	16.5	0.3	16.0	8.7	49,973	160	0.715E-10
+ILUK	m=10, K=5	7.5	3.3	4.1	1.8	30,050	50	0.922E-10
	m=10, K=10	5.8	4.2	1.4	0.5	38,735	10	0.438E-11
+ILUTH	m=10, $\tau = .01$	13.0	1.6	11.2	5.5	26,104	180	0.579E-10
	m=5, $\tau = .001$	92.4	1.7	90.5		23,287	*1000	0.298E-06
	m=10, $\tau = .001$	97.2	1.8	95.2		32,142	*1000	0.204E-06
	m=20, $\tau = .001$	9.9	1.8	8.0	4.3	49,852	80	0.746E-10
GMRES /								
+SOR	m=10, $\omega = 1.0$	94.5			37.3	17,710	*1000	0.529E-06
	m=10, $\omega = 1.95$	49.8			18.7	17,710	*500	0.416E-03

Table 1: Performance results for test example. N=1,771; NZ=11,011. NCD Case.

In the figure ILUTH refers to an ILU factorization with threshold τ, which consists of dropping all elements during the factorization that are smaller than τ in magnitude. ILUK refers to a strategy whereby only the K largest elements in L and in U are kept.

As can be seen from this example Gaussian elimination is faster than the iterative solvers, although it requires far more storage. However, for the larger problems treated in [10] Gaussian elimination was too slow or required too much memory. In some cases it also failed to produce an answer. Among the iterative solvers tested and not shown in the table, were the SOR method and the preconditioned power method. The SOR method was generally not reliable because of the demand for an optimal parameter. As for the power method it converged more slowly than the methods shown here. One surprising observation made in the experiments in [10] is the fact that the FOM method was often more reliable than its GMRES counterpart. Another point that is not too well understood is the effect of improving the incomplete factorization by taking, for example, a smaller threshold in

ILUTH, or a larger K in ILUK. Whereas the quality of the factorization improves, since the error matrix E is smaller, this does not always mean that the number of iterations in the preconditioned Krylov Subspace method will be smaller. This is clearly illustrated in the table. The phenomenon may be due to the fact that the L and U factors produced by a more accurate factorization have in fact a worse condition number than with the less accurate factorization. The relation between the accuracy in the preconditioning technique and the overall performance is not clear. Note however, that the best results are obtained with the more accurate preconditioners.

References

[1] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[2] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problems.* Academic Press, Orlando, Florida, 1984.

[3] F. L. Bauer. Das verfahren der treppeniteration und verwandte verfahren zur losung algebraischer eigenwertprobleme. *ZAMP*, 8:214–235, 1957.

[4] J. Cullum and R. Willoughby. A Lanczos procedure for the modal analysis of very large nonsymmetric matrices. In *Proceedings of the 23rd Conference on Decision and Control, Las Vegas*, 1984.

[5] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Clarendon Press, Oxford, 1986.

[6] A. Jennings and G. M. Malik. Partial elimination. *J. Inst. Math. Appl.*, 20:307–316, 1977.

[7] A. Jennings and W.J. Stewart. A simultaneous iteration algorithm for real matrices. *ACM, Trans. of Math. Software*, 7:184–198, 1981.

[8] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31(137):148–162, 1977.

[9] B. N. Parlett and D. R. Taylor andf Z. S. Liu. A look-ahead Lanczos algorithm for nonsymmetric matrices. *Mathematics of Computation*, 44:105–124, 1985.

[10] B. Philippe, Y. Saad, and W. J. Stewart. Numerical methods in markov chain modeling. Technical Report 89.39, RIACS, NASA Ames, Mofett Field, CA, 1989.

[11] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.*, 34:269–295, 1980.

[12] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37:105–126, 1981.

[13] Y. Saad. Projection methods for solving large sparse eigenvalue problems. In B. Kagstrom and A. Ruhe, editors, *Matrix Pencils, proceedings, Pitea Havsbad*, pages 121–144, Berlin, 1982. University of Umea, Sweden, Springer Verlag. Lecture notes in Math. Series, Number= 973.

[14] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Mathematics of Computation*, 42:567–588, 1984.

[15] Y. Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Comput. Phys. Comm*, 53, 1989.

[16] Y. Saad and M. H. Schultz. Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Mathematics of Computation*, 44(170):417–424, 1985.

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

[18] G.W. Stewart. Simultaneous iteration for computing invariant subspaces of non-hermitian matrices. *Numer. Math.*, 25:123–136, 1976.

[19] G.W. Stewart. SRRIT - a fortran subroutine to calculate the dominant invariant subspaces of a real matrix. Technical Report TR-514, University of Maryland, College Park, MD, 1978.

[20] Z. Zlatev. Use of iterative refinement in the solution of sparse linear systems. *SIAM J. Numer. Anal.*, 19:381–399, 1982.