COMPARISON OF PARTITIONING TECHNIQUES FOR TWO-LEVEL ITERATIVE SOLVERS ON LARGE, SPARSE MARKOV CHAINS

TUĞRUL DAYAR* AND WILLIAM J. STEWART†

Abstract. Experimental results for large, sparse Markov chains, especially the ill-conditioned nearly completely decomposable (NCD) ones, are few. We believe there is need for further research in this area, specifically to help in understanding the effects of the degree of coupling of NCD Markov chains and their nonzero structure on the convergence characteristics and space requirements of iterative solvers. The work of several researchers has raised the following questions that led to research in a related direction. How one must go about partitioning the global coefficient matrix into blocks when the system is NCD and a two-level iterative solver (such as block SOR) is to be employed? Are block partitionings dictated by the NCD normal form of the stochastic one-step transition probability matrix necessarily superior to others? Is it worth investing alternative partitionings? Better yet, for a fixed labeling and partitioning of the states, how does the performance of block SOR (or even that of point SOR) compare to the performance of the iterative aggregation-disaggregation (IAD) algorithm? Finally, is there any merit in using two-level iterative solvers when preconditioned Krylov subspace methods are available? We seek answers to these questions on a test suite of thirteen Markov chains arising in seven applications.

Key Words. Markov chains, near complete decomposability, partitioning, block SOR, iterative aggregation-disaggregation, Krylov subspace methods, preconditioning.

1. Introduction. Solving for the stationary distribution of an irreducible Markov chain amounts to computing a positive solution vector to a homogeneous system of linear equations with a singular coefficient matrix subject to a normalization constraint. That is, the $(n \times 1)$ unknown stationary vector x in

$$Ax = 0, \quad \|x\|_1 = 1 \tag{1}$$

is to be found. Here $A = I - P^T$ is an $n \times n$ singular M-matrix [6] and P is a one-step stochastic transition probability matrix.

Of special interest are nearly completely decomposable (NCD) Markov chains [21]. An NCD Markov chain may be symmetrically permuted to the normal form

$$P_{n\times n} = \begin{array}{c} \begin{array}{cccc} n_1 & n_2 & \cdots & n_N \end{array} \\ \left(\begin{array}{cccc} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{array} \right) \end{array} \begin{array}{c} n_1 \\ n_2 \\ \vdots \\ n_N \end{array}, \tag{2}$$

in which the nonzero elements of the off-diagonal blocks are small compared with those of the diagonal blocks. The subblocks P_{ii} are square and of order n_i, with $n = \sum_{i=1}^{N} n_i$. Let $P = \text{diag}(P_{11}, P_{22}, \ldots, P_{NN}) + E$. The quantity $\|E\|_\infty$ is referred to as the degree of coupling and is taken to be a measure of the decomposability of the matrix.

* Department of Computer Engineering and Information Science, Bilkent University, 06533 Bilkent, Ankara, Turkey (tugrul@cs.bilkent.edu.tr).

† Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA (billy@csc.ncsu.edu).

Despite recent advances, practicing performance analysts generally prefer iterative methods based on splittings when they want to compare the performance of newly devised algorithms against existing ones, or when they need candidate solvers to evaluate the performance of a systems model at hand. Experimental results for large, sparse Markov chains, especially the ill-conditioned NCD ones, are few. We believe there is a need for further research in this area, specifically to help in understanding the effects of the degree of coupling of NCD Markov chains and their nonzero structure on the convergence characteristics and space requirements of iterative solvers.

The work of several researchers [23, 16, 17, 15, 8, 22, 18] has raised important and interesting questions that led to research in a related direction. These questions are the following: "How must one go about partitioning the global coefficient matrix A in (1) into blocks when the system is NCD and a two-level iterative solver (such as block SOR) is to be employed? Are block partitionings dictated by the NCD normal form of P necessarily superior to others? Is it worth investing alternative partitionings? Better yet, for a fixed labeling and partitioning of the states, how does the performance of block SOR (or even that of point SOR) compare to the performance of the iterative aggregation-disaggregation (IAD) algorithm [32]? Finally, is there any merit in using two-level iterative solvers when preconditioned Krylov subspace methods [3, 26, 13, 24, 11, 27] are available?"

Four block partitioning techniques are considered. The first one results from the near-complete decomposability test (*ncdtest*) of the MARkov Chain Analyzer (MARCA) [33]. It determines the strongly connected components of the transition probability matrix by ignoring the nonzeros less than a prespecified decomposability parameter. Then symmetric permutations are performed to put the matrix into the form in which the diagonal blocks form the strongly connected components. In a recent paper [9], it is shown that the *ncdtest* algorithm may fail to produce a correct NCD partitioning of the state space. The same paper highlights an improved NCD partitioning algorithm, which has the same run-time complexity as that of *ncdtest*. We name this new NCD partitioning algorithm *newncd* and experiment with it. Also two straightforward partitionings are investigated. The *equal* partitioning forms (approximately) equal order blocks. The second straightforward partitioning, *other*, uses blocks of order respectively 1,2,3,... Finally, the Threshold PABLO (TPABLO) partitioning algorithm [8] is considered on some of the test problems.

When seeking answers to these questions, we have not considered two-level solvers of the inner-outer iteration type [22], but have attempted at solving diagonal blocks (and the coupling matrix [21] in IAD) directly by Gaussian elimination. The memory needed to solve the coupling matrix is set aside at the beginning and what is left is used for diagonal blocks. Blocks of order 1 and 2 are treated separately. We obtain the LU factorizations of as many diagonal blocks as possible given available memory and do this in such a way that smaller blocks are treated first, leaving the big blocks to be solved using point SOR when there is insufficient memory. Currently, we use a considerably large tolerance (i.e., 10^{-3}), a relaxation parameter of 1.0 (hence, Gauss-Seidel), and a maximum number of iterations of 100 with the point SOR algorithm when solving diagonal blocks. Furthermore, the block Gauss-Seidel correction [36] in the disaggregation step of IAD is replaced by block

SOR.

Preconditioned Krylov subspace methods [28] are state-of-the-art iterative solvers developed mostly in the last fifteen years that may be used, among other things, to solve for the stationary distribution of Markov chains [34]. A concise discussion on popular Krylov subspace methods and the motivation behind preconditioning may be found in [4]. In this study, we consider the methods Generalized Minimum RESidual (GMRES), Direct Quasi-GMRES (DQGMRES), BiConjugate Gradient (BCG), Conjugate Gradient Squared (CGS), BiConjugate Gradient Stabilized (BCGStab), and Quasi-Minimal Residual (QMR) with Incomplete LU (ILU) factorization preconditioning. Chapter 4 of [34] presents some of these methods for Markov chains.

Results of experiments on a test suite of thirteen Markov chains show that two-level iterative solvers are in most cases superior to ILU preconditioned Krylov subspace solvers. For two-level iterative solvers, there are cases in which a straightforward partitioning of the coefficient matrix gives a faster solution than can be obtained using the *ncdtest* or *newncd* partitioning algorithms. However, in between *newncd* and *ncdtest*, the former gives faster converging iterations than the latter in a larger number of the test cases. In general, it is possible to solve each of the problems (except one which takes a minimum of about 82 seconds to solve) in less than 1 minute. This includes time spent for partitioning or preconditioning.

Section 2 discusses the methods used in the experiments with their space and time complexities per iteration and introduces relevant issues. The results of the numerical experiments are analyzed in Section 3 after a detailed description of the implementation framework. Appendices A through G provide a detailed explanation of each test problem, the nonzero plots of the underlying matrices, information about the matrices and the partitionings, and the complete results.

2. Numerical Solution Methods. The term iterative methods refers to a wide range of techniques that use successive approximations to obtain a more accurate solution to a linear system at each step. Iterative methods of one type or another are the most commonly used methods for obtaining the stationary probability from either the stochastic transition probability matrix or from the infinitesimal generator of a Markov chain. This choice is due to several reasons. First, in iterative methods, the only operations in which the matrices are involved are multiplications with one or more vectors. These operations preserve the nonzero structure of the matrix. This may lead to considerable savings in memory required to solve the system especially when dealing with large, sparse matrices. Besides, an iterative process may be terminated once a prespecified tolerance criterion has been satisfied, and this may be relatively lax. For instance, it may be wasteful to compute the solution of a mathematical model correct to full machine precision when the model itself contains errors. However, a direct method (such as Gaussian elimination) is obligated to continue until the final operation has been performed.

In this study, we experiment with the (point) successive overrelaxation (SOR) method [34, 4], a stationary iterative method. Moreover, two types of two-level iterative methods are considered: block SOR (BSOR) [34, 28, 22] and IAD [36, 32, 34, 10]. As for projection techniques, we choose to implement and experiment with the Krylov subspace methods GMRES [29], DQGMRES [30], BCG [4], CGS [31], BCGStab [37], and QMR [12].

2.1. Successive Over Relaxation (SOR). Stationary iterative methods are methods that can be expressed in the simple form [4]

$$x^{(k+1)} = T\,x^{(k)} + c, \quad k = 0, 1, \ldots, \tag{3}$$

where neither T nor c depend on the iteration count k. Equation (1) can be written in the form of (3) by splitting the coefficient matrix A. Given a splitting

$$A = M - N$$

with nonsingular M, we have the iterative procedure

$$x^{(k+1)} = M^{-1}N\,x^{(k)} = T\,x^{(k)}, \quad k = 0, 1, \ldots, \tag{4}$$

where $x^{(0)}$ is the initial guess. $T = M^{-1}N$ is the iteration matrix, and in our case, the vector c appearing in (3) is just the zero vector.

For convergence of (4), it is required that $\lim_{k\to\infty} T^k$ exists (since $x^{(k)} = T^k\,x^{(0)}$). A necessary condition for convergence is for all eigenvalues of T to be less than or equal to 1 in modulus, i.e., $\rho(T) \leq 1$, where $\rho(T)$ is the spectral radius of T. When $\rho(T) = 1$, the unit eigenvalue of T must be the only eigenvalue with modulus 1 for convergence to be realized.

In general [4], stationary iterative methods differ in the way the coefficient matrix is split. The splitting uniquely defines the iteration matrix, and hence, determines the convergence rate of the method. For the SOR method with relaxation parameter ω, the splitting is

$$A = (\frac{1}{\omega}D - L) - (\frac{1-\omega}{\omega}D + U),$$

where $D, -L, -U$ represent respectively diagonal, strictly lower triangular, strictly upper triangular parts of A. The iteration matrix for (forward) SOR is then given by

$$T_{SOR} = (\frac{1}{\omega}D - L)^{-1}(\frac{1-\omega}{\omega}D + U).$$

The SOR iteration may be expressed as

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega\left\{\frac{1}{d_{ii}}\left(\sum_{j=1}^{i-1} l_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} u_{ij}x_j^{(k)}\right)\right\}, \quad i = 1, 2, \ldots, n,$$

or in matrix form as

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega\left\{D^{-1}(Lx^{(k+1)} + Ux^{(k)})\right\}. \tag{5}$$

It can be verified for (5) that the solution vector x (which is the transpose of the stationary probability vector) is the eigenvector corresponding to the unit eigenvalue of the SOR iteration matrix. The SOR method converges only if $0 < \omega < 2$. The optimal value of ω is that which maximizes the difference between the unit eigenvalue and the subdominant eigenvalue of T_{SOR}. Therefore, the convergence rate of SOR is

highly dependent on ω. In general, it is not possible to compute in advance the optimal value of ω. Even when this is possible, the cost of such computation is usually prohibitive.

It is worth stressing that for $0 < \omega < 1$, $\rho(T_{SOR}) = 1$. Given an irreducible coefficient matrix, for the case of underrelaxation, the unit eigenvalue is the dominant eigenvalue of the iteration matrix (see [3, p.362]) and one has a converging iteration.

Table 2.1 at the end of this section presents a summary of the operations per iteration and the storage requirement for all the methods used in the experiments. Only the space required to store the matrices and vectors that appear in the iteration loop of the algorithms is considered. The order of the coefficient matrix is given by n. The SAXPY column gives the number of scalar times vector plus vector operations (excluding inner products) per iteration. The meaning of "Precond Solve" will be explained in the next subsection, where we discuss two-level iterative methods and the partitionings used.

2.2. Two-Level Iterative Methods. These methods follow a decompositional approach in solving systems of linear equations. If the model is too large or too complex to analyze as an entity, it is divided into subsystems, each of which is analyzed separately, and a global solution is then constructed from the partial solutions. Ideally, the problem is broken into subproblems that can be solved independently, and the global solution is obtained by concatenating the subproblem solutions. Although two-level iterative methods generally require more computation per iteration than stationary iterative methods, this is usually offset by a faster rate of convergence.

The convergence study of classical block methods, such as Block Jacobi and Block SOR, for Markov chains appear, for instance, in [16]. To study the convergence of non-stationary two-level iterative methods of the inner-outer iteration type, consider the defining homogeneous system of linear equations in (1). Let A be partitioned as

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NN} \end{pmatrix}. \tag{6}$$

Now consider the splitting $A = M - N$, where A has the form in (6) and M is a nonsingular block diagonal matrix given by

$$M = \operatorname{diag}(A_{11}, A_{22}, \ldots, A_{NN}). \tag{7}$$

At each iteration of these two-level methods, N linear systems of the form

$$A_{ii} x_i^{(k+1)} = z_i, \quad i = 1, \ldots, N$$

are solved at the second level. Nonstationarity of the method implies the possibility to perform different number of iterations at the second level for each of the N linear systems. The following theorem regarding the convergence of non-stationary block two-level methods for Markov chains appears in [22].

THEOREM 2.1. *Let P be a transition matrix of a finite homogeneous Markov chain. Consider $A = I - P^T$ partitioned as in (6) and the splitting $A = M - N$ defined in (7). If*

each matrix A_{ii}, $1 \leq i \leq N$, is either strictly or irreducibly column diagonally dominant, then Block Jacobi and Block GS both give convergent iterations.

When applied to NCD Markov chains, one possibility is to order and partition the state space so that the stochastic matrix of transition probabilities has the form in (2). Obviously a zero degree of coupling (i.e., $\|E\|_\infty = 0$) implies a completely decomposable matrix. In NCD systems, there are eigenvalues close to 1. The poor separation of the unit eigenvalue results in slow rate of convergence for standard matrix iterative methods. Two-level iterative methods in general do not suffer from this limitation which makes them suitable for such systems.

2.2.1. Block SOR (BSOR). Let A be partitioned as in (6). We introduce the block splitting

$$A = D_N - (L_N + U_N),$$

where D_N is a block diagonal matrix and L_N and U_N are respectively strictly lower and upper block triangular matrices. We then have

$$D_N = \begin{pmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{NN} \end{pmatrix},$$

$$L_N = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ L_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & 0 \end{pmatrix}, \quad U_N = \begin{pmatrix} 0 & U_{12} & \cdots & U_{1N} \\ 0 & 0 & \cdots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

In analogy with (5), the BSOR method is given by

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega \left\{ D_N^{-1}(L_N x^{(k+1)} + U_N x^{(k)}) \right\}.$$

If we write this for each subvector, we get

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega \left\{ D_{ii}^{-1} \left(\sum_{j=1}^{i-1} L_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{N} U_{ij} x_j^{(k)} \right) \right\},$$

where the subvectors x_i are partitioned conformally with D_{ii} for $i = 1, 2, \ldots, N$. This implies that at each iteration we must solve N systems of linear equations

$$D_{ii} x_i^{(k+1)} = z_i, \quad i = 1, 2, \ldots, N, \tag{8}$$

where

$$z_i = (1-\omega)\, D_{ii}\, x_i^{(k)} + \omega \left(\sum_{j=1}^{i-1} L_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{N} U_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, N.$$

The right-hand side z_i has to be computed before the ith system is solved. A step that solves a (preconditioned) system as in (8) is called one "Precond Solve" throughout the paper.

If the matrix A is irreducible (which is the case in our experiments), then it is clear from (8) that at each iteration we are going to solve N nonhomogeneous systems of equations with nonsingular coefficient matrices. This can be achieved by employing either direct or iterative methods. Different criteria may affect the choice of the method for solving a diagonal block as there is no requirement to use the same method to solve all diagonal blocks. In general, for a given coefficient matrix A, the larger the order of blocks (and hence the smaller the number of blocks), the fewer the (outer) iterations required to achieve convergence (see [34, p.141]). The reduction in the number of iterations is usually offset to a certain degree by an increase in the number of operations that are to be performed at each iteration. However, this is not always true as it is highly dependent on matrix structure. We return to implementation issues in Section 3.

2.2.2. Iterative Aggregation-Disaggregation (IAD). Suppose we have a (NCD) Markov chain characterized by a probability matrix P having the block structure in (2), and let π be the stationary distribution of P (i.e., $\pi P = \pi$, $\|\pi\|_1 = 1$) partitioned conformally with P such that $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$.

For each diagonal block P_{ii}, in the transition probability matrix P, there exists a stochastic complement S_{ii} [21]. The stochastic complement reflects the behavior of the system within the corresponding block of states. Each stochastic complement is itself a stochastic transition probability matrix of an irreducible Markov chain whose state space is composed of the states of that block. The probability that the system is in a certain state of block i given that the process is in one of the states of that block, can be determined from the conditional stationary probability vector of the ith block, $\pi_i/\|\pi_i\|_1$. This can be computed by solving $(\pi_i/\|\pi_i\|_1)S_{ii} = \pi_i/\|\pi_i\|_1$. As can be inferred, a stochastic complement may be too expensive to compute as it has an embedded matrix inversion. One way to overcome this problem is to approximate S_{ii} by accumulating the mass in the off-diagonal blocks of the ith row of blocks into the diagonal block P_{ii} on a row-by-row basis. This can be achieved in various ways. An approximation to the conditional stationary vector of the corresponding block can then be found by solving the linear system as described before.

It is possible to compute the probability of being in a given block of states if we have an $N \times N$ stochastic matrix whose ijth element denotes the probability of transitioning from block i to block j. This matrix is called the coupling matrix and it characterizes the interactions among blocks. To construct this matrix, we need to shrink each block P_{ij} of P down to a single element. This is accomplished by first replacing each row of each block by the sum of its elements. Mathematically, the operation performed for each block is $P_{ij}\, e$, where e is a column vector of 1's whose length is determined by the context in which it is used. The sum of elements of row k of block P_{ij} gives the probability of leaving state k of block i and entering one of the states of block j. To determine the total probability of leaving (any state of) block i to enter (any state of) block j, we need to sum the elements of $P_{ij}\, e$ after each of these elements has been weighed by the probability that the system is in (one of the states of) block i. These weighing factors may be obtained from the

elements of the stationary probability vector; they are the components of $\pi_i/\|\pi_i\|_1$. Hence the ijth element of the coupling matrix is given by $(\pi_i/\|\pi_i\|_1)P_{ij}e$. The stationary vector of the coupling matrix gives the stationary probability of being in each block of states. More precisely, the multiplicative constants mentioned before, form the elements of the stationary vector of the coupling matrix. However, forming the coupling matrix requires computing the stationary vector. This can be achieved by approximating the coupling matrix by starting with an approximate stationary vector and improving the approximate solution iteratively. The method motivated as such has come to be known as iterative aggregation-disaggregation (IAD) [36, 32, 34, 10].

It is possible to perceive each iteration of the IAD algorithm as being formed of a preprocessing step followed by one iteration of a two-level method, such as BSOR. The preprocessing step corresponds to the solution of the coupling matrix and is called aggregation. The BSOR iteration is the disaggregation step. In the IAD algorithm, the residual error (i.e., $\|\pi(I-P)\|$) decreases by a factor of $\|E\|$ at each iteration if each diagonal block and the coupling matrix are solved exactly. This global convergence result of IAD with block Gauss-Seidel at the second level is given in [32]. The study of local and global convergence of IAD (with the possibility of using iterative solution methods for the aggregation step) appear in [17] and [18].

One of the crucial steps in the IAD algorithm is solving the coupling matrix accurately. The coupling matrix is a singular irreducible stochastic matrix of order N whose states form a single communicating class. Consequently it has a unique unit eigenvalue and $(N-1)$ other eigenvalues. The smaller the degree of coupling the closer these other eigenvalues to 1. One has several alternative methods to consider for solving the coupling matrix. We choose to use Gaussian elimination (GE) for several reasons. Since the coupling matrix is a singular M-matrix with 0 column sums, GE preserves column diagonal dominance in exact arithmetic throughout its computation. Hence, the multiplier element at each step is bounded by 1 thereby avoiding the need of pivoting. Besides, iterative methods tend to converge slowly if all the nonunit eigenvalues of the coupling matrix are close to 1. On the other hand, GE may suffer from unstability in the presence of rounding errors on coupling matrices [10] obtained from the NCD form in (2).

2.2.3. Partitioning Techniques. Four block partitioning techniques are considered. The first one is the *ncdtest* partitioning algorithm in MARCA. This algorithm searches for the strongly connected components (SCCs) of the directed graph (digraph) associated with the matrix obtained by zeroing the elements of P that are less than a user specified decomposability parameter γ, a real number between 0 and 1. The subset(s) of states output by the SCC search algorithm are identified as forming the NCD blocks P_{ii}. If the matrix is not already in the form (2), then symmetric permutations are performed to put it into the form in which the diagonal blocks form the SCCs. The *ncdtest* algorithm may fail to produce a correct NCD partitioning of the state space due to the possibility of having nonzeros greater than or equal to γ in the off-diagonal blocks. This is simply because the algorithm zeroes out the elements that are smaller than γ, but *not* those that are larger. The example in [9] shows how this can happen and presents an improved NCD partitioning algorithm, which has the same run-time complexity as that of *ncdtest*. We name this new NCD partitioning algorithm *newncd* and experiment with it.

For clarity, we use γ' to denote the decomposability parameter of the *newncd* algorithm. Also two straightforward partitionings are investigated. The *equal* partitioning has $\sqrt{n}$ blocks of order $\sqrt{n}$ if n is a perfect square. If $n \neq \lfloor\sqrt{n}\rfloor^2$, there is an extra block of order $n - \lfloor\sqrt{n}\rfloor^2$. The second straightforward partitioning, *other*, has nb blocks of order respectively $1, 2, \ldots, nb$ if $n = \sum_{i=1}^{nb} i$ (and possibly an extra block of order $n - \sum_{i=1}^{nb} i$ if the difference is positive). This last partitioning ensures that there are about $\sqrt{2n}$ blocks and the largest block solved is of order roughly $\sqrt{2n}$.

We have also experimented with the TPABLO partitioning algorithm [8] on some of the test problems. The original PABLO (PAramaterized BLock Ordering) algorithm presented in [23] aims at obtaining dense diagonal blocks by performing symmetric permutations of a given sparse coefficient matrix using two input parameters. The first parameter $\alpha > 0$ is used to ensure that the addition of a new state to a diagonal block will keep the ratio, of the percentage of nonzero elements in that block to the percentage of nonzero elements in that block if the state were not added, above α. The second parameter $0 \leq \beta \leq 1$ is used to ensure that each state in a diagonal block is adjacent to at least a certain proportion, i.e., β, of the states inside the diagonal block. In addition to these two parameters, TPABLO has three other parameters requiring a total of five parameters. The third parameter $\gamma \geq 0$ either makes sure the permuted matrix does not have any elements in the off-diagonal blocks that are larger than γ in absolute value, or it makes sure all elements in the diagonal blocks are above γ in absolute value with the possibility that some elements in the off-diagonal blocks are also larger than γ in absolute value. The fourth and fifth parameters *minbs* and *maxbs* are used to control the minimum and maximum permissible order of diagonal blocks, respectively.

The next subsection discusses a different class of solvers, namely those that are based on the Krylov subspace.

2.3. Projection Methods. Projection methods differ from stationary and two-level iterative methods in that successive approximations are computed from small dimension subspaces. Projection methods, themselves, differ from each other in the way subspaces are selected and solution approximations are extracted from them. A projection step is defined formally with two objects: a subspace $\mathcal{K}$ of dimension m from which the approximation is to be selected and another subspace $\mathcal{L}$ (of the same size m) that is used to set the constraints necessary to extract the new approximated solution vector from $\mathcal{K}$ [24, 27].

Consider the linear system

$$Ax = b. \tag{9}$$

Let $V = [v_1, v_2, \ldots, v_m]$ and $W = [\omega_1, \omega_2, \ldots, \omega_m]$ be respectively the bases of $\mathcal{K}$ and $\mathcal{L}$ [34]. Then we can write the approximate solution as $\tilde{x} = Vy$, where y is now a vector of $\mathbb{R}^m$. This gives us m degrees of freedom, and in order to extract a unique y we require that the residual vector $b - A\tilde{x}$ be orthogonal to $\mathcal{L}$; i.e.,

$$b - AVy \perp \omega_i, \quad i = 1, 2, \ldots, m.$$

In matrix form this can be written as

$$W^T(b - AVy) = 0,$$

which yields,

$$y = [W^T AV]^{-1} W^T b.$$

Thus the minimum assumption that must be made in order for the projection processes to be feasible is for $W^T AV$ to be nonsingular. If we start with $x^{(0)}$ as an initial approximate solution to the system, then $x^{(0)}$ may be adjusted by a vector δ such that $x^{(0)} + \delta$ is a solution, i.e., $A(x^{(0)} + \delta) = b$. If we set $r_0 \equiv b - Ax^{(0)}$, then

$$A(x^{(0)} + \delta) = b \quad \Rightarrow \quad Ax^{(0)} + A\delta = b \quad \Rightarrow \quad A\delta = b - Ax^{(0)} = r_0,$$

and hence the projection step is applied to the system $A\delta = r_0$ to compute the unknown vector δ. It follows that a general projection algorithm may be given by [34]:

Until Convergence Do:

1. Select a pair of subspaces $\mathcal{K}$ and $\mathcal{L}$, and an initial guess $\tilde{x}$.
2. Choose bases $V = [v_1, v_2, \ldots, v_m]$ and $W = [\omega_1, \omega_2, \ldots, \omega_m]$ for $\mathcal{K}$ and $\mathcal{L}$.
3. Compute

$$\begin{aligned} r &\leftarrow b - A\tilde{x}, \\ y &\leftarrow [W^T AV]^{-1} W^T r, \\ \tilde{x} &\leftarrow \tilde{x} + Ay. \end{aligned}$$

FIGURE 2.1 *Algorithm: Prototype Projection Method.*

Let (x, y) denote the inner product of vectors x and y. For a matrix A, denote by $\|x\|_A$ the A-norm of vector x, which is defined as $\|x\|_A \equiv (Ax, x)^{1/2}$.

Projection methods are classified in two main groups [24, 27]. The first is when the Krylov subspace $\mathcal{K}$ is taken as $\mathcal{K} = \mathcal{L} = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$ and $V = W$ is an orthogonal basis of $\mathcal{K}$. This represents the class of Galerkin projection methods (also known as orthogonal projection methods). In this group of methods, each iteration minimizes $\|x - \tilde{x}\|_A$ in the direction of the residual vector r $(= b - A\tilde{x})$. The second group of projection methods is when $\mathcal{L} = A\mathcal{K} = \text{span}\{Ar_0, A^2r_0, \ldots, A^m r_0\}$ (and hence $W = AV$). Each iteration of this kind of methods minimizes the 2-norm of the residual vector, i.e., $\|b - A\tilde{x}\|_2 = \min_{z \in \mathcal{K}} \|b - Az\|_2$. This explains why these methods are referred to as minimal residual methods.

In this study, we consider six Krylov subspace methods. To provide effective solvers, all methods are used with preconditioners. The main idea behind preconditioning is to transform the linear system so that the difference between the dominant and the subdominant eigenvalue of the preconditioned coefficient matrix is larger than what it used to be in the original system. We return to preconditioning techniques later in this subsection. The pseudocodes of all methods but DQGMRES may be found in [4]. For DQGMRES, we refer to [30].

2.3.1. Generalized Minimum Residual (GMRES). The Generalized Minimum Residual method [29] lies in the class of minimal residual methods and is designed to solve nonsymmetric linear systems. The GMRES version [4, pp.18-21] discussed in this subsection is based on the Arnoldi method which is a modified Gram-Schmidt orthogonalization

procedure applied to the Krylov subspace span$\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$ to form the basis of the subspace and store it in a Hessenberg matrix. The GMRES iterates are constructed as

$$x^{(k)} = x^{(0)} + y_1 v^{(1)} + \cdots + y_k v^{(k)},$$

where $v^{(i)}$ are the Arnoldi vectors that span $\mathcal{K}_m$ and y_i are the coefficients that minimize the residual 2-norm $\|b - Ax^{(k)}\|_2$. The GMRES algorithm has the property that this residual norm can be determined before computing the iterate. This enables postponing the expensive operation of forming the iterate until the residual norm is deemed small enough. To control the storage requirements, restarts are used, i.e., the iterate is formed after each m iterations. At each restart a new basis of the Krylov subspace is formed; hence, we have GMRES(m).

The crucial element for successful application of GMRES(m) resolves around the decision of when to restart, that is the choice of m. Obviously if no restarts are used (i.e., $m = n$), GMRES, and all orthogonalizing Krylov subspace methods, converge in n steps. However, because of storage limitation this may not be feasible for large n. The amount of computation and storage required by GMRES in one iteration increases linearly with the (inner) iteration count i. This is regarded as the major drawback of the method.

2.3.2. Direct Quasi-GMRES (DQGMRES). The Direct Quasi-Generalized Minimum Residual method is a direct version of the Quasi-GMRES (QGMRES) method [30] in which the Arnoldi process of GMRES is replaced with an incomplete orthogonalization procedure. There are no restarts involved and only the last k vectors need to be kept, hence incomplete orthogonalization and DQGMRES(k). DQGMRES may save computations over GMRES but not storage.

2.3.3. BiConjugate Gradient (BCG). The BiConjugate Gradient method is an orthogonal projection method and it takes an advantage over GMRES by reducing the storage demand [4, pp.21-23]. This is achieved by replacing the orthogonal sequence of residuals (formed by GMRES to build the basis of the Krylov subspace) by two mutually orthogonal sequences of residual vectors

$$r^{(k)} = r^{(k-1)} - \alpha_k A p^{(k)}, \quad \tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \alpha_k A^T \tilde{p}^{(k)}.$$

The two sequences of search directions are

$$p^{(k)} = r^{(k-1)} + \beta_{k-1} p^{(k-1)}, \quad \tilde{p}^{(k)} = \tilde{r}^{(k-1)} + \beta_{k-1} \tilde{p}^{(k-1)}.$$

To ensure the bi-orthogonality relation

$$\tilde{r}^{(k)^T} r^{(j)} = \tilde{p}^{(k)^T} A p^{(j)} = 0 \quad \text{for } k \neq j,$$

one sets

$$\alpha_k = \frac{\tilde{r}^{(k-1)^T} r^{(k-1)}}{\tilde{p}^{(k)^T} A p^{(k)}}, \quad \beta_k = \frac{\tilde{r}^{(k)^T} r^{(k)}}{\tilde{r}^{(k-1)^T} r^{(k-1)}}.$$

It is observed that the convergence behavior of BCG is quite irregular. The method breaks down when $z^{(k-1)^T}\tilde{r}^{(k-1)} \approx 0$. Another possible breakdown situation is when $\tilde{p}^{(k)^T}q^{(k)} \approx 0$. To increase the effectiveness of BCG, variants such as CGS [31] and BCGStab [37] have been proposed.

2.3.4. Conjugate Gradient Squared (CGS). Consider the residual vector $r^{(k)}$ computed at the kth iteration of BCG. This vector may be written as a product of $r^{(0)}$ and an kth degree polynomial in A [4] such that

$$r^{(k)} = P_k(A)\, r^{(0)}.$$

The same polynomial is applicable to $\tilde{r}^{(k)}$ (i.e., $\tilde{r}^{(k)} = P_k(A)\,\tilde{r}^{(0)}$). As can be inferred, the role of the polynomial $P_k(A)$ is to reduce the initial residual $r^{(0)}$ to $r^{(k)}$ in k iterations. Therefore, applying the same polynomial twice (i.e., $r^{(k)} = P_k^2(A)\, r^{(0)}$) will logically reduce $r^{(0)}$ much faster. This approach leads to the Conjugate Gradient Squared method [31].

The rate of convergence of CGS is generally twice that of BCG. However, this is not always the case since a reduced residual vector $r^{(k)}$ may not be reduced any further. This explains the highly irregular behavior of CGS. Moreover, rounding errors are very likely to occur in CGS as local corrections to the current solution may be very large, and hence the final computed solution may not be very accurate [4, pp.25-27]. Another property which seems to be paradoxic at first glance is that the method tends to diverge if one chooses to start with an initial guess close to the solution. For what concerns the time complexity, CGS is almost as expensive as BCG. However, it is worth mentioning that CGS does not involve computations with A^T.

2.3.5. BiConjugate Gradient Stabilized (BCGStab). The BiConjugate Gradient Stabilized method [37] was developed so that it is as fast as CGS while avoiding the often irregular convergence patterns of the latter [4, pp.27-28]. It can then be said that BCGStab is suitable for nonsymmetric linear systems. The idea behind this method is to use a kth degree polynomial other than P_k, say Q_k, to further reduce the residual vector [37]. In other words, instead of writing the residual as $r^{(k)} = P_k^2(A)r^{(0)}$, one writes $r^{(k)} = Q_k(A)P_k(A)r^{(0)}$. BCGStab requires slightly more computations per iteration than CGS and BCG as it requires two matrix-vector products and four inner products.

2.3.6. Quasi-Minimal Residual (QMR). The Quasi-Minimal Residual method [12] attempts to overcome the problems of irregular convergence behavior and breakdowns observed in some of the projection methods such as BCG. The QMR method uses a least squares approach similar to that followed in GMRES. However, GMRES uses an orthogonal basis for the constructed Krylov subspace whereas QMR uses a bi-orthogonal one. Thereby, the obtained solution is viewed as quasi-minimal residual solution, which explains the name.

To avoid breakdowns, QMR uses look-ahead techniques which makes it more robust than BCG. These techniques enable QMR to prevent all breakdowns except the so-called "incurable breakdown". The version of QMR [4, pp.23-25] we used in our experiments is simpler than the full QMR method with look-ahead, but it is still more robust than BCG. The algorithm we used includes a relatively inexpensive recurrence relation for

computing the residual vector at the expense of a few extra vectors of storage and vector update operations per iteration. It also avoids performing a matrix-vector product to compute the residual vector. A full-fledged implementation of QMR with look-ahead is available through `netlib` at `http://netlib.cs.utk.edu/liblist.html`.

TABLE 2.1
Summary of Operations and Storage Requirements.

Method	Inner Product	SAXPY	Matrix-Vector Product	Precond Solve	Storage Requirement
SOR	0	1	1 [a]	0	matrix+n
BSOR	0	1	1	N [b]	matrix+$2n$
IAD	1	2	2	$(N+1)$ [b]	2 matrices[c]+N+$2n$
GMRES(m)[d]	i+1	$2i$	1	1	2 matrices[e]+ $(i+5)n+im$
DQGMRES(k)[f]	$k+2$	$2k+3$	1	1	2 matrices[e]+$n(2k+1)$
BCG	2	5	1/1[g]	1/1[g]	2 matrices[e]+$9n$
CGS	2	7	2	2	2 matrices[e]+$10n$
BCGStab	4	6	2	2	2 matrices[e]+$9n$
QMR	2	8	1/1[g]	1/1[g]	2 matrices[e]+$15n$

[a] The method performs no real matrix-vector product or preconditioner solve, but the number of operations is equivalent to a matrix-vector multiply.

[b] Since the order of diagonal blocks (and for IAD also the number of blocks) in the partitioning are not necessarily the same, the size of the operands in the given counts are most likely different.

[c] Two square matrices of orders n and N.

[d] Note that the (inner) iteration count i ranges between 1 and m.

[e] The coefficient matrix and the preconditioner.

[f] Note that the operation counts are given for number of iterations larger than k.

[g] One with the coefficient matrix, one with its tranpose.

2.3.7. Preconditioners. A very important issue for iterative methods is the concept of preconditioning. Although preconditioning can be used in all iterative methods, we employ it in Krylov subspace methods only. The idea behind preconditioning is to accelerate the convergence process by redistributing the eigenvalues of the coefficient matrix so that the difference between the dominant and the subdominant eigenvalue becomes larger without changing the solution vector. Therefore, the need for a preconditioner becomes vital when dealing with NCD systems.

Again consider the system of linear equations in (9), which can be transformed into the right-preconditioned equivalent system

$$AM^{-1}(Mx) = b,$$

or into the left-preconditioned equivalent system

$$M^{-1}Ax = M^{-1}b,$$

where the preconditioner matrix M (also called preconditioner) has the property that it is a cheap approximation of A. The more M^{-1} resembles A^{-1}, the faster the method converges [20].

In the case of right-preconditioning, the system $AM^{-1}y = b$ is solved for the unknown $y \equiv Mx$, and the final solution x is obtained through the post-transformation $x = M^{-1}y$. To use right-preconditioning, M should also be chosen so that $M^{-1}v$ is cheap to compute for any arbitrary vector v.

In the left-preconditioning case, the system is solved based on imposing the necessary stopping constraints on the preconditioned residual vector $r = M^{-1}(b - Ax)$. The matrix M^{-1} need not be formed explicitly and the preconditioned residual may be computed by solving the system $Mr = b - Ax$. Therefore, the preconditioner M should be chosen so that solving any linear system of the form $Mv = u$ for any vector v is cheap.

Various types of preconditioners have been (and are still being) developed (see [28, 5]). Their efficiency is highly dependent on the system to be solved, and it is quite difficult to forecast which preconditioner is the best for a given system. In this study, we only consider preconditioners obtained from Incomplete LU factorizations (ILU). First, an LU factorization of the coefficient matrix A is initiated. Throughout the factorization, nonzero elements are omitted according to different rules. These rules characterize the ILU type. Thus, instead of ending up with an exact LU factorization, what we obtain is of the form

$$A = \tilde{L}\tilde{U} + E, \tag{10}$$

where E, called the remainder, is expected to be small in some sense. The incomplete LU factors $\tilde{L}$ and $\tilde{U}$ are respectively lower and upper triangular matrices.

Recall that the coefficient matrices appearing in the systems of interest are irreducible singular M-matrices. It has been shown that ILU factorizations exist for such matrices [7] (in exact arithmetic) and that they are at least as stable as the complete LU factorization without partial pivoting (see [20, p.152]).

Three types of incomplete LU factorizations are considered. The first imposes on the computed preconditioner the same nonzero structure as the original matrix and is called ILU0. The idea of ILU0 is to drop all fill-in elements which occur during the LU factorization (recall that a fill-in element refers to a nonzero element introduced in the matrix which holds the LU factors in a location where there was initially a zero element in the original matrix).

The second is called ILUTH and is a threshold-based approach. In ILUTH, the factorization takes place in a row-by-row manner. The dropping rule of this preconditioning technique is to zero out all elements having an absolute value less than a prespecified threshold. The only exception is that the dropping rule does not apply to the diagonal elements which are kept no matter how small they become. The dropping rule is applied just after the multipliers are formed, once, and applied one more time right after the reduction of a row is over.

The third type of ILU preconditioner forces the computed factors to have at most a prespecified fixed number of nonzero elements per row and is called ILUK. This approach enables the user to control the amount of fill-in. Therefore, it is suitable in case there is

only a fixed amount of memory available to store the incomplete factors $\tilde{L}$ and $\tilde{U}$. Each time a row has been reduced, a search is conducted to find the K largest elements in absolute value, a timewise costly process. All other elements in the row are annihilated. As for ILUTH, the diagonal elements are preserved regardless of their magnitude.

Finally, we should stress that not much work has been done in studying what constitutes a good incomplete factorization for Markov chain models [26, 24, 27]. Further studies are still needed.

2.4. Stopping Criteria. One of the most critical steps in iterative methods is to decide when to stop the iteration. A good stopping criterion should (see Section 4.2 in [4])

- identify when the error $e^{(k)} \equiv x^{(k)} - x$ is small enough to stop,
- stop if the error is no longer decreasing, or decreasing too slowly, and
- limit the maximum amount of time spent iterating.

Ideally the iteration should stop when the magnitudes of entries of the error $e^{(k)}$ fall below a user supplied threshold, *stop_tol*. Nevertheless, since the exact solution x is generally not known, it is practically not feasible to compute $e^{(k)}$. Instead, the residual vector $r^{(k)} = b - Ax^{(k)}$ for $Ax = b$ which is more readily computed, is used. The user may choose the value of *stop_tol* as the approximate uncertainty in the entries of A and b relative to respectively[1] $\|A\|$ and $\|b\|$.

The quantity $\|e^{(k)}\|$, known as the forward error, is hard to estimate directly. Hence, it is usually the backward error that is used to bound the forward error. The normwise backward error is defined in [4, p.53] as the smallest possible value of $\max\{\|\delta A\|/\|A\|, \|\delta b\|/\|b\|\}$, where $x^{(k)}$ is the exact solution of $(A + \delta A)x^{(k)} = (b + \delta b)$, and it can also be written as $\|r^{(k)}\|/\|A\|$. The backward error is more practical to use than the forward error since it can be easily computed using $r^{(k)}$ and the coefficient matrix:

$$e^{(k)} = x^{(k)} - x = A^{-1}(Ax^{(k)} - b),$$

hence

$$\|e^{(k)}\| \leq \|A^{-1}\| \cdot \|r^{(k)}\| \tag{11}$$

$$= \|A^{-1}\| \cdot \|A\| \cdot \frac{\|r^{(k)}\|}{\|A\|}. \tag{12}$$

For a singular matrix A, the group inverse $A^{\#}$ can replace A^{-1} in equations (11) and (12). The expression $\|A^{-1}\| \cdot \|A\|$ is referred to as the condition number of A. From equation (11), we see that, if the algorithm stops due to the test $\|r^{(k)}\| \leq stop\_tol$, the forward error can be upper-bounded by $stop\_tol\,\|A^{-1}\|$. There also exist the concepts of relative forward error defined by $\|e^{(k)}\|/\|x^{(k)}\|$ and relative backward error defined by $\|r^{(k)}\|/(\|A\| \cdot \|x^{(k)}\|)$. Directly from equation (12) we can upper-bound the relative forward error in the following way:

$$\frac{\|e^{(k)}\|}{\|x^{(k)}\|} \leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|r^{(k)}\|}{\|A\| \cdot \|x^{(k)}\|}.$$

[1] The norm is not important as long as we are consistent.

Different stopping criteria has been suggested for the convergence test of iterative methods. Several criteria are discussed in [2, 14]. Unfortunately, there is no single stopping criterion known to be suitable for all iterative methods. Hence, selecting the most appropriate one is a difficult decision to make during the implementation of the solver. However, knowing the solvers and the implicit residual each produces helps. The amount of computation required by the convergence test is another constraint which should be taken into consideration.

3. Numerical Results. In this section, we discuss implementation issues and overview the results of the numerical experiments performed on the thirteen test cases. Details corresponding to each of the seven applications can be found in Appendices A through G. The applications are named *pushout*, *2D*, *ncd*, *telecom*, *qn*, *leaky*, and *mutex*.

3.1. Implementation Issues. Since we are dealing with large sparse systems[2], we are required to use a sparse storage scheme. We use the compact sparse row (CSR) Harwell-Boeing format [25, 34], which requires three arrays: one real and one integer of size nz (i.e., number of nonzero elements in the coefficient matrix), and one integer of size $n+1$. Unless otherwise specified, by reductions we mean row-reductions. This strategy is used to take full advantage of the row-by-row storage of the CSR format. We would like to remark that we generate and store all test matrices using the MARCA package. Later these files are used as input to the solvers.

All code is written in Fortran and compiled in double precision with *g77* on a SUN Sparcstation with 64 Mb RAM running Solaris 2.5. The numerical experiments are timed using a C function that reports CPU time. SOR, the two-level iterative solvers, the four partitioning algorithms, and the three ILU preconditioners are part of the MARCA software package. The Krylov subspace methods are implemented using two one-dimensional arrays defined at the beginning of the driver program to hold double precision and integer values.

In two-level iterative methods, we attempt to solve diagonal blocks, and the coupling matrix in IAD, directly by Gaussian elimination. The memory needed to solve the coupling matrix is set aside at the beginning and what is left is used for the diagonal blocks. If there is not enough space for solving the coupling matrix, the method fails. Blocks of order 1 and 2 are treated separately. We obtain the LU factorizations of as many diagonal blocks as possible given available memory and do this in such a way that smaller blocks are treated first, leaving the big blocks to be solved using SOR when there is insufficient memory. In order to accelerate this process we use a considerably large tolerance 10^{-3}, a maximum number of iterations of 100, and a relaxation parameter of 1.0 (hence, Gauss-Seidel) with the SOR algorithm when solving the remaining diagonal blocks. Furthermore, the block Gauss-Seidel correction [36] in the disaggregation step is replaced by BSOR. The results reported are always those that are obtained using the optimal relaxation parameter ω with one significant digit after the decimal point.

The *newncd* partitioning algorithm is implemented so that if there are states that are left in singletons after the NCD partition corresponding to a decomposability parameter

[2] The average order of the seven problems we experimented with is $33,278$; the largest matrix is of order $104,625$ and the smallest one is of order $8,258$.

γ' is determined, they are grouped into a single subset which forms the last NCD block. When choosing decomposability parameters γ for *ncdtest*, we report the smallest and largest values of γ (as 0.10 times a power of 10) for which there are at least two blocks in the partition. On the other hand, when experimenting with *newncd*, we had to work on a finer scale with γ' since there were not as many possibilities as γ of *ncdtest*. Hence, we report the smallest and largest values of γ' (in two decimal digits of precision times a power of 10) for which there are at least two blocks in the partition.

The dimension of the Krylov subspace we used for (restarted) GMRES is 20 (i.e., $m = 20$). The number of vectors kept in DQGMRES is 20 (i.e., $k = 20$) in all but one of the applications (*qn*), where we had to limit k to 7 or 9 depending on the preconditioner used. With each Krylov subspace solver, we used three different thresholds for the ILUTH preconditioner: 10^{-2}, 10^{-3}, and 10^{-5}. Due to the amount of fill-in and the computation time, it is futile to experiment with a threshold value of 10^{-5} in two of the applications (*qn* and *mutex*). In ILUK, we allowed a maximum of 10 nonzero elements per row of the preconditioned matrix (i.e., $K = 10$). In all the Krylov subspace methods implemented, we use left-preconditioning and take the ILU preconditioner as $M = \tilde{L}\hat{U}$ (see (10) in Section 2.3).

In order to regulate the amount of fill-in produced, ILUTH is implemented in such a way that before the reduction of a given row, the number of free entries in the double precision work array is divided by the number of remaining rows to be reduced. This gives us the maximum number of allowable nonzero elements that can be stored for the current row. If the reduction gives a higher number of nonzero elements than the allowable maximum, the threshold is multiplied by 10 and the dropping rule is applied again. This is repeated until the number of nonzero elements in a given row becomes less than or equal to the allowable maximum. The first row of the matrix is not reduced, and the method is forced to fail if the magnitude of any reduced diagonal element is less than 10^{-300}.

In ILUK, the Kth largest value in magnitude, say max, in the reduced row is determined. Then all elements having an absolute value less than max are set to zero. If the number of nonzero elements in the row is still higher than K, the reduced row is scanned from left to right and elements having an absolute value equal to max are set to zero until the number of nonzero elements becomes K. As in ILUTH, the reduction does not include the first row of the matrix and the method fails if any reduced diagonal element is found to be less than 10^{-300}.

In order to reduce the possibility of underflow and overflow, each row of the coefficient matrix is multiplied by the inverse of the largest value in magnitude in that row (i.e., absolute value of the diagonal element). This is a scaling operation and it transforms the system to a more suitable form without altering the global solution. Normalizing the solution vector at each iteration is an alternative way to limit the effect of underflow and overflow and up to a certain extent control the irregular convergence behavior of some iterative methods. The drawback of this strategy is that it may lead to considerable loss of precision due to rounding errors that occur at each iteration or to even divergence. In SOR, BSOR, and IAD, which are all part of MARCA, the coefficient matrix is scaled and the solution vector is normalized at each iteration. As for the Krylov subspace methods

we implemented, the coefficient matrix is not scaled and the solution vector is normalized only upon termination.

The stopping criteria we use in SOR and two-level solvers are respectively

$$\textbf{stop if} \quad k \geq maxit \quad \textbf{or} \quad \|x^{(k)} - x^{(k-1)}\|_\infty \leq stop\_tol$$

and

$$\begin{aligned}\textbf{stop if} \quad & k \geq maxit \quad \textbf{or} \quad \|x^{(k)} - x^{(k-1)}\|_\infty \leq stop\_tol \quad \textbf{or} \\ & \Big(\|x^{(k)} - x^{(k-1)}\|_\infty \leq stop\_tol_1 \quad \textbf{and} \\ & \quad |(\|x^{(k)} - x^{(k-1)}\|_\infty - \|x^{(k-1)} - x^{(k-2)}\|_\infty)| \leq stop\_tol_2\Big),\end{aligned}$$

where k is the iteration count, $maxit$ is the maximum number of iterations the algorithm will be permitted to perform, and $stop\_tol$ is the user-specified stopping tolerance, which should be less than 1 and greater than machine epsilon.

The stopping criterion we use in the Krylov subspace methods of interest is

$$\begin{aligned}\textbf{stop if} \quad & k \geq maxit \quad \textbf{or} \quad \|r^{(k)}\|_\infty \leq stop\_tol \quad \textbf{or} \\ & \Big(\|r^{(k)}\|_\infty \leq stop\_tol_1 \quad \textbf{and} \quad |(\|r^{(k)}\|_\infty - \|r^{(k-1)}\|_\infty)| \leq stop\_tol_2\Big).\end{aligned}$$

The stopping tolerance, $stop\_tol$, is set to 10^{-10}, meaning we consider the entries of A (our right-hand side is 0) to have errors in the range $\pm 10^{-10}\|A\|$. The use of $stop\_tol_1$ and $stop\_tol_2$ forces the solver to terminate when the norm of the residual is decreasing too slowly while the difference between two successive iterates is small enough. In the experiments, we set $stop\_tol_1$ and $stop\_tol_2$ to 10^{-6} and 10^{-12}, respectively. As for $maxit$, we use 100, 500, or 1,000 depending on the solver and the particular problem at hand.

For each problem solved (see Table 3.1), the true residual and the relative backward error in the solution are computed (see [14, 2]). The true residual is computed as $\|A\hat{x}\|_\infty$, where $\hat{x}$ is the normalized approximate solution upon termination. The relative backward error is computed as $\|A\hat{x}\|_\infty / (\|A\|_\infty \|\hat{x}\|_\infty)$. If a Krylov subspace solver terminates at iteration k, then $\|r\|$ denotes $\|r^{(k)}\|_\infty$. These norms are byproducts of all Krylov subspace methods except GMRES and DQGMRES, and need not be computed separately. Due to this, we compare $\|r^{(k)}\|_2$ (and not $\|r^{(k)}\|_\infty$) with $stop\_tol$ at each (inner) iteration of GMRES. At the end of each restart the true residual is computed explicitly from the unnormalized current approximation and then compared with $stop\_tol$. As for DQGMRES, it is the scalar gamma that is compared with $stop\_tol$ at each iteration (see the algorithm in [28]). Hence, in these two solvers, $\|r\|$ upon termination is explicitly computed using the solution vector. If BCGStab converges due to the convergence test $\|s\|_\infty \leq stop\_tol$ (see the BCGStab algorithm on p.27 in [4]), then $\|r\|$ stands for $\|s\|_\infty$ upon termination. In this case a superscript "s" (i.e., ^{s}) is inserted in the corresponding cell. A dagger (i.e., †) in the same column indicates termination due to $stop\_tol_2$ (see Section 2.4) although the residual norm $\|r\|$ upon termination was found to be less than 10^{-10}. On the contrary, a double dagger (i.e., ‡) is used to denote those cases where the termination was normal but $\|r\|$ turned out to be larger than 10^{-10}. An asterisk (i.e., *) following the iteration number means the solver failed to converge in that many

iterations. Hence, the existence of a residual norm $\|r\|$ larger than 10^{-10} and the absence of an asterisk in the #it column denotes those cases where the termination was due to $stop\_tol_2$. The column heading $\|\Delta x\|$ denotes the infinity norm of the difference between the last two approximations of SOR, BSOR, and IAD.

TABLE 3.1
Notation Used in the Tables of Results.

n	Order of the coefficient matrix
nz	Number of nonzero elements in the coefficient matrix
$nzlu$	Number of nonzero elements in the incomplete LU factorization
ω	Optimal relaxation parameter for SOR, BSOR, and IAD
γ/γ'	Decomposability parameter for *ncdtest*/*newncd*
Time	Time (in seconds) taken by the method or the preconditioner
#it	Number of iterations performed
$\|r\|$	Infinity norm of the residual incurred by the method upon termination (exception for GMRES and BCGStab)
$\|A\hat{x}\|$	True residual upon termination
Bk.Error	Relative backward error upon termination
$\|\Delta x\|$	Infinity norm of the difference between the last two iterates
Blocks	Number of diagonal blocks solved iteratively
Partition.	Partitioning technique used

3.2. Overview of Results. We consider seven models, six of which appear in [35] and one is discussed in [1]. All seven models arise in Markov chain applications. Three of these models (i.e., *pushout*, *ncd*, and *mutex*) are chosen and two more test problems for each one generated (namely, *medium*, *hard*, *ncd_alt*1, *ncd_alt*2, *mutex_alt*1, and *mutex_alt*2) giving us a total of thirteen test problems. The original *pushout* test problem is given the name *easy* as will be explained later. In the appendices, we provide information about each test matrix and their bandwidth[3]. We also give the degree of coupling for each partitioning (i.e., $\|E\|_\infty$ for the partitioning in (2) of Section 1) and the coefficient of asymmetry (i.e., (a_0, a_1), where $a_0 = 0.5\|A + A^T\|_1, a_1 = 0.5\|A - A^T\|_1$, and $a_o \ll a_1$ implies high asymmetry) for each test matrix. The majority of these matrices would be ranked among the largest of the matrices considered in the Matrix Market [19].

None of the thirteen test matrices is highly asymmetric. Two of the problems (i.e., *ncd* and *mutex*) give test matrices with symmetric nonzero structure. Furthermore, the *ncdtest* algorithm is unable to find partitionings in NCD normal form as in (2) except for the *ncd* test matrices. In fact, the degree of coupling values corresponding to various partitionings in all the other test matrices are notoriously large (see the discussion in 2.2.3). The symmetric nonzero structure of the *ncd* test problem seems to have helped the *ncdtest* algorithm. Finally, we would single out *leaky*, *ncd_alt*2, *ncd*, *ncd_alt*1, and *telecom* as NCD test cases based on the smallest decomposability parameter that could be used with the *newncd* partitioning algorithm. These test cases have degree of coupling values ranging (in the given order) from $0.2e-101$ to $0.9e-2$. The test cases *medium*, *qn*,

[3] We adopt the convention that higher and lower bandwidths do not include the diagonal.

hard, *easy*, and $2D$ have degree of coupling values between $0.7e+0$ and $0.1e+0$. Hence, they are not as NCD. The remaining test cases *mutex*, *mutex_alt*1, and *mutex_alt*2 lie somewhere in between, all with degree of coupling $0.2e-1$.

The time spent for partitioning using *equal* and *other* is negligible. On the other hand, the time to partition a given test matrix using the *ncdtest* and *newncd* algorithms has two components: time spent to determine the partition and time to permute the coefficient matrix according to the ordering of states in the computed partition. The time taken by the *ncdtest* and *newncd* partitionings used in our experiments does not exceed respectively 1 and 1.8 seconds except for matrices generated from the *qn*, *leaky*, and *mutex* test problems. The time spent by *ncdtest* and *newncd* for the *qn* test matrix is no more than 5.4 and 6.8 seconds, respectively. The time spent by *ncdtest* and *newncd* for the *leaky* test matrix is no more than 2.8 and 8.5 seconds, respectively. Finally, the time spent by *ncdtest* and *newncd* for the *mutex* test matrices is no more than 5.5 and 5.8 seconds, respectively.

Numerical experiments show that two-level iterative solvers are, in general, superior to SOR and Krylov subspace solvers with the chosen preconditioners. Out of ten test matrices for which two-level iterative solvers are winners, BSOR is the fastest solver for seven test matrices, three times with *equal* (*ncd_alt*2, *mutex_alt*1, *mutex_alt*2), two times with *other* (*easy*, *ncd_alt*1), one time with *ncdtest* (*qn*) and *newncd* (*leaky*) each. IAD is the fastest solver for three test matrices, two times with *newncd* (*hard*, *ncd*) and one time with *equal* (*medium*). CGS and BCGStab with ILUTH(10^{-5}) each turns out to be the fastest solver for one test matrix (*telecom* and $2D$, respectively). SOR is the winner for one test matrix (*mutex*).

It is noticed that the more balanced, in terms of the order of blocks, is the partitioning, the better two-level iterative solvers take advantage of the divide-and-conquer notion, and hence the faster they converge. For those test cases in which two-level solvers are winners, none of the diagonal blocks are solved iteratively. The IAD algorithm proves to be competitive with BSOR. When the coupling matrix is of reasonable size, IAD usually gives good performance. We especially recommend IAD for those cases that have a small degree of coupling. However, the drawback of IAD is that it may fail if the coupling matrix is reducible or require an unreasonably long time to converge when the coupling matrix is large. Straightforward partitionings, especially *equal*, are very competitive with those of *newncd*. Out of ten test matrices, the *equal* and *other* partitionings provide winners for respectively four and two test matrices.

SOR does not give satisfactory results; it converges in less than 1,000 iterations in only eight of the test matrices. Interestingly, the optimal relaxation parameter for SOR and BSOR always happens to be equal to or larger than 1.0. For IAD, the optimal relaxation parameter turns out to be 0.9 for a few test matrices, otherwise it is larger. In most of the experiments, 1.0 is the optimal choice.

Among the Krylov subspace methods of interest, it is clear that BCGStab performs the best. It converges for all the test matrices with at least one preconditioner. Its total solution time is always the shortest or very close to that of an outperforming Krylov subspace solver. CGS comes second and GMRES third, the latter being more costly in terms of memory requirements and number of flops per iteration. QMR is also

competitive in some cases; however, it almost always terminates with the alternative stopping criterion. There are cases in which DQGMRES takes a smaller number of iterations than the corresponding GMRES solver, but even in those cases its solution time is almost always longer. BCG performs very poorly and converges only for a few test matrices.

We should point out that the ILU0 preconditioner leads to better total solution time than all the other ILU preconditioners for five test matrices (*easy*, *qn*, *mutex*, *mutex_alt*1, *mutex_alt*2) out of twelve with which we could use ILU preconditioners. As for the threshold preconditioners, ILUTH(10^{-3}) is the best preconditioner for two test matrices (*medium*, *hard*) whereas ILUTH(10^{-5}) is the best preconditioner for five test matrices (*2D*, *ncd*, *ncd_alt*1, *ncd_alt*2, *telecom*). There are cases which show that a denser preconditioner is not always the better preconditioner. The problem with ILUK is the long time overhead to form the preconditioner. The ILUTH(10^{-3}) and ILUTH(10^{-5}) preconditioners are superior to ILU0 for test matrices that are of medium order (around $20,000$ states), have narrow bandwidth (such as *2D*) or are relatively more ill-conditioned (such as *ncd_alt*2, *ncd*, *ncd_alt*1, *telecom*, *hard*, *medium*). When the Markov chain is ill-conditioned (such as *leaky*), incomplete *LU* factorization may fail causing preconditioned Krylov subspace methods to fail as well. Moreover, we see that the ILU0 preconditioner may be very effective if the coefficient matrix is quite large, but sparse (such as *qn*), or dense (such as *mutex*). For the latter type of matrices, SOR is recommended. It is clear that Krylov subspace solvers are affected adversely with higher ill-conditioning. However, higher ill-conditioning does not always imply poorer performance. It is noticed in some cases that it may even help a solver, especially IAD, to converge faster.

We executed the TPABLO partitioning algorithm on five of the test matrices and recorded the computed partitionings for $\alpha = \beta = 0.5$, $minbs = 10$, $maxbs = 200$. Then we solved for the stationary distribution using both BSOR and IAD with the recorded block structure and the optimal threshold value γ of TPABLO (see Section 2.2.3), which we picked from $\{0.10e{+}0, 0.10e{-}1, 0.10e{-}2, 0.10e{-}3\}$. When choosing the test matrices, we tried to form a representative set of problems with different degrees of difficulty and sparsity patterns. Unfortunately, it was not possible to use TPABLO with the *qn* and *mutex* test matrices. The winners of these experiments are given in Table 3.2. The figures in the P.Time column indicate partitioning times. Note that the last column has more information than the same column of the tables in the appendices. The first integer again stands for the number of blocks solved iteratively. The integer after the colon is the number of blocks in the partition, and the arguments inside the parantheses are respectively the order of the smallest and largest blocks.

TABLE 3.2

Solvers with TPABLO ordering, $\alpha = \beta = 0.5, minbs = 10, maxbs = 200$.

Matrix	Solver	ω	P.Time	Time	#it	Bk.Error	Blocks
easy	IAD, $\gamma = 0.10e-3$	1.0	6.3	8.2	9	$0.24e-13$	0 : 103(8 – 200)
medium	IAD, $\gamma = 0.10e-2$	1.1	6.5	8.6	7	$0.78e-13$	0 : 103(8 – 200)
2D	IAD, $\gamma = 0.10e+0$	1.1	9.2	4.8	19	$0.18e-11$	0 : 129(129 – 129)
ncd	BSOR, $\gamma = 0.10e-1$	1.5	6.7	16.2	67	$0.61e-15$	0 : 1,040(10 – 52)
telecom	IAD, $\gamma = 0.10e-1$	1.0	5.2	23.7	49	$0.99e-13$	0 : 196(6 – 200)

Note that TPABLO gives mostly balanced partitionings for the chosen parameters, and it turns out to be the case that, when input to the two-level solvers of MARCA, all diagonal blocks in these partitionings are solved directly. Also, in the partitionings TPABLO computes, it can come up with blocks of order less than *minbs* (or larger than *maxbs*, something observed in our experiments). None of the TPABLO solvers considered provide a winner when compared with the results in the appendices though BSOR with TPABLO $\gamma = 0.10e-1$ is competitive with *ncdtest* $\gamma = 0.10e-3$ for the *ncd* test matrix. The partitioning provided by TPABLO for the $2D$ matrix is an *equal* partitioning, however, with a different ordering of the states. The solution time for $2D$ is better than its IAD with *equal* counterpart if we exclude the partitioning time. However, in both the *ncd* and $2D$ test matrices, there is a faster IAD solver with a *newncd* partitioning. Our conclusion regarding TPABLO is that it may give faster converging orderings, but with a set of five parameters, it is quite difficult to fine-tune.

4. Acknowledgments. We thank Wail Gueaieb for providing the framework to carry out the experiments with the Krylov subspace solvers, Yousef Saad for his comments on this work, and Daniel Szyld for supplying the TPABLO routines and his comments on an earlier version.

References

[1] Ö. Aras and T. Dayar. Complete buffer sharing with pushout thresholds in ATM networks under bursty arrivals. *Proceedings of the First Symposium on Computer Networks, 30-31 May '96, İstanbul, Turkey* (1996), 144–156.

[2] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, UK, 1994.

[3] V. A. Barker. Numerical solution of sparse singular systems of equations arising from ergodic Markov chains. *Comm. Statist. Stochastic Models*, 5:355–381, 1989.

[4] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems.* SIAM, Philadelphia, PA, 1994.

[5] M. Benzi and M. Tuma. A comparision of some preconditioning techniques for general sparse matrices. In S. Margenov and P. Vassilevski (Eds.), *Iterative Methods in Linear Algebra, II, IMACS Series in Computational and Applied Mathematics, Vol. 3*, IMACS, NJ (1996), 191–203.

[6] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* SIAM, Philadelphia, PA, 1994.

[7] J. J. Buoni. Incomplete factorization of singular M-matrices. *SIAM J. Alg. Disc. Meth.*, 7:193–198, 1986.

[8] H. Choi and D. B. Szyld. Application of threshold partitioning of sparse matrices to Markov chains. *Proceedings of the IEEE International Computer Performance and Dependability Symposium IPDS'96*, Urbana, IL (1996) 158–165.

[9] T. Dayar. Permuting Markov chains to nearly completely decomposable form. *SIAM J. Matrix. Anal. Appl.*, submitted for publication.

[10] T. Dayar and W. J. Stewart. On the effects of using the Grassmann-Taksar-Heyman method in iterative aggregation-disaggregation. *SIAM J. Sci. Comput.*, 17:287–303, 1996

[11] R. W. Freund and M. Hochbruck. On the use of two QMR algorithms for solving singular systems and applications in Markov chain modelling. *Numer. Linear Algebra with Applic.*, 1:403–420, 1994.

[12] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[13] D. Gross, B. Gu, and R. M. Soland. The biconjugate gradient method for obtaining the steady-state probability distributions of Markovian multiechelon repairable item inventory systems. In W. J. Stewart (Ed.), *Numerical Solution of Markov Chains*, M. Dekker, Inc., New York (1991), 473–489.

[14] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA (1996).
[15] S. T. Leutenegger and G. H. Horton. On the utility of the multi-level algorithm for the solution of nearly completely decomposable Markov chains. In W. J. Stewart (Ed.), *Computations with Markov Chains: Proceedings of the Second International Workshop on the Numerical Solution of Markov Chains*, Kluwer, Boston, MA (1995), 425–442.
[16] I. Marek and D. B. Szyld. Iterative and semi-iterative methods for computing stationary probability vectors of Markov operators. *Math. Comp.*, 61:719–731, 1993.
[17] I. Marek and D. B. Szyld. Local convergence of the (exact and inexact) iterative aggregation method for linear systems and Markov operators. *Numer. Math.*, 69:61–82, 1994.
[18] I. Marek and P. Mayer. Convergence theory of an aggregation/disaggregation iteration method. *J. Comp. Appl. Math.*, 62: unassigned, 1997.
[19] Matrix Market. A repository of test matrices at the National Institute of Standards and Technology. `http://math.nist.gov/MatrixMarket`.
[20] 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.*, 31:148–162, 1977.
[21] C. D. Meyer. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Rev.*, 31:240–272, 1989.
[22] V. Migallón, J. Penadés, and D. B. Szyld. Block two-stage methods for singular systems and Markov chains. *Numer. Linear Algebra with Applic.*, 3:413–426, 1996.
[23] J. O'Neil and D. B. Szyld. A block ordering method for sparse matrices. *SIAM J. Sci. Stat. Comput.*, 11:811–823, 1990.
[24] B. Philippe, Y. Saad, and W. J. Stewart. Numerical methods in Markov chain modelling. *Opns. Res.*, 40:1156–1179, 1992.
[25] Y. Saad. SPARSKIT: A Basic Tool Kit for Sparse Matrix Computation. Tech. Report CSRD TR 1029, Center for Supercomputing Research and Development, University of Illinois at Urbana Champaign, 1990.
[26] Y. Saad. Projection methods for the numerical solution of Markov chain models. In W. J. Stewart (Ed.), *Numerical Solution of Markov Chains*, M. Dekker, Inc., New York (1991), 455–471.
[27] Y. Saad. Preconditioned Krylov subspace methods for the numerical solution of Markov chains. In W. J. Stewart (Ed.), *Computations with Markov Chains: Proceedings of the Second International Workshop on the Numerical Solution of Markov Chains*, Kluwer, Boston, MA (1995), 49–64.
[28] Y. Saad. *Iterative Solution of Sparse Linear Systems.* PWS Publishing, New York, 1996.
[29] Y. Saad and M. H. Schultz. GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.
[30] Y. Saad and K. Wu. DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization. *Numer. Linear Algebra with Applic.*, 3:329–343, 1996.
[31] P. Sonneveld. CGS: A fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.
[32] G. W. Stewart, W. J. Stewart and D. F. McAllister. A two-stage iteration for solving nearly completely decomposable Markov chains. In G. H. Golub, A. Greenbaum, and M. Luskin (Eds.), *The IMA Volumes in Mathematics and its Applications 60: Recent Advances in Iterative Methods*, Springer-Verlag, New York (1994) 201–216.
[33] W. J. Stewart. *MARCA*: Markov Chain Analyzer. A software package for Markov modelling. In W. J. Stewart (Ed.), *Numerical Solution of Markov Chains*, M. Dekker, Inc., New York (1991), 37–62.
[34] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, NJ, 1994.
[35] W. J. Stewart. Marca models: a database of Markov chain models. `http://www.csc.ncsu.edu/faculty/WStewart/MARCA_Models/MARCA_Models.html`.
[36] W. J. Stewart and W. Wu. Numerical experiments with iteration and aggregation for Markov chains. *ORSA J. Comput.*, 4:336–350, 1992.
[37] H. A. van der Vorst. BI-CGSTAB: A fast and smootly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

Appendix

A. Complete Buffer Sharing with Pushout Thresholds in ATM Networks. Broadband Integrated Services Digital Networks (B-ISDNs) are to support multiple types of traffic such as voice, video, and data. The Asynchronous Transfer Mode (ATM) is the support technique of choice for B-ISDNs by the standards committees. In this mode of operation, all information is carried using fixed size packets (called 'cell's) so as to share the network among multiple classes of traffic. Since multiclass traffic will be carried on B-ISDNs, different quality of service requirements will be imposed by different applications.

One type of congestion control for ATM networks deals with discarding cells at ATM buffers in order to guarantee a prespecified cell loss rate. One bit in each ATM header is reserved to assign the space priorities of cells. This bit indicates whether the given cell is high or low priority. Priority cell discarding is a buffer management scheme in which higher priority cells are favored in receiving buffer space. An efficient technique for determining the cells to be discarded when congestion occurs is the complete buffer sharing scheme with pushout thresholds.

In the system under consideration, there are two classes of traffic arriving to an ATM buffer of size K. Time is divided into fixed size slots of length equal to one cell transmission time. The arrival of traffic class l $(= 1, 2)$ to the buffer is modeled as a Bernoulli process with probability of cell arrival p_l in a slot.

The states of the corresponding queueing system may be represented by the ordered pair (i, j), where i and j are the number of class 1 and class 2 cells in the buffer, respectively [1]. Let k $(= i + j)$ denote the total number of cells in the buffer at state (i, j). Then, a natural state space ordering that places the states with the same number of total cells in the buffer (i.e., k) consecutively, gives rise to a block matrix with $\sum_{k=0}^{K}(k+1) = (K+1)(K+2)/2$ states. The first block consists of the state (0,0) (i.e., the state in which the buffer is empty), the second block has states (0,1), (1,0), the third block has states (0,2),(1,1),(2,0), and so on. The kth block has $k+1$ states. That is, we have the following ordering:

$$(0,0) \prec (0,1) \prec (1,0) \prec (0,2) \prec (1,1) \prec (2,0) \prec \cdots \prec (K,0)$$

During a time slot, no cells, one cell, or two cells may arrive. If one or two cells arrive, then this happens at the beginning of a slot. A cell departure occurs by the end of the slot if the buffer has at least one cell at the beginning of the slot. Hence, an arriving cell cannot be transmitted before the end of the next slot. With these assumptions, a cell is discarded if and only if two cells arrive to a full buffer. The pushout threshold for class 2 cells is given by T_2 and the pushout threshold of class 1 cells is given by T_1 $(= K - T_2)$. If two cells arrive to a full buffer (i.e., $i + j = K$), then a class 2 cell is discarded if $j > T_2$, otherwise a class 1 cell is discarded if $j < T_2$. When $j = T_2$, the lower priority traffic class cell is discarded. One may view the system as if there is temporary space to store up to two arrivals while the buffer is full and a decision as to which class of cell will be discarded is made.

It is assumed that at steady-state the head of the queue (i.e., the cell that will be leaving the buffer at the end of the current time slot—if there was one to begin with) is a type 1 cell with probability $i/(i+j)$ and it is a type 2 cell with probability $j/(i+j)$.

The discrete-time Markov chain corresponding to these assumptions is block tridiagonal (with the exception of the first row of blocks) where each diagonal block is tridiagonal and has a different block size (see Figure A.1). Depending on the selected threshold, the nonzero elements in the last row of blocks change making it difficult to apply analytical solution techniques to such a system with control.

To study the effect of the threshold, three test cases are generated. In all test cases, K and T_2 are fixed to 200 and 20, respectively. In the first test case, which we call *easy*, we set $p_1 = 0.99$ and $p_2 = 0.99$. The second test case, *medium*, is generated by choosing $p_1 = 0.1$ and $p_2 = 0.5$. Setting p_1 and p_2 respectively to 0.1 and 0.9 gives us a third test case which we call *hard*. The coefficient matrices of the three test cases are of the same order $n = 20,301$ and have the same number of nonzero elements $nz = 140,504$, bandwidth, and nonzero structure. Tables A.2, A.3 and A.4 show the results of *ncdtest*, *equal*, *other*, and *newncd* partitionings applied to the three coefficient matrices. As can be seen from Table A.2, choosing $\gamma = 0.10e-1$ causes the coefficient matrix to be partitioned into blocks of order 1 with the exception of one block which is of order 2. Such cases are not interesting and we do not consider them in our experiments. The possible values of γ' for the *newncd* partitioning algorithm on the *easy* test matrix range between $0.11e+0$ and $0.98e+0$ with $0.11e+0$ being the smallest degree of coupling corresponding to the values of γ' we tested in the mentioned range. The time to partition the *pushout* test matrices using *ncdtest* and *newncd* is not larger than 1 and 1.4 seconds, respectively. Table A.5 gives information about the symmetric nonzero structure and bandwidth of the three test matrices.

The first thing we notice in the results of the *easy* test matrix is that IAD with *ncdtest* $\gamma = 0.10e-1$ requires an unreasonably long solution time due to the order of the coupling matrix (see Table A.6). However, BSOR performs very well especially with the *other* and *ncdtest* $\gamma = 0.10e-2$ partitionings. The former happens to be the winning solver for this test matrix. This may be explained by examining the nonzero structure of the *pushout* matrices (see Figure A.1). The *easy* test matrix is block tridiagonal (with the exception of the first row of blocks) where diagonal blocks are tridiagonal with increasing block sizes as we move down the matrix. Hence, the matrix is narrow banded in the first few rows and the bandwidth increases down the matrix. This enables the *other* partitioning to gather most of the nonzero elements within diagonal blocks. The *ncdtest* partitioning with $\gamma = 0.10e-2$ gave a similar block structure to that of *other* and hence close performance. We just want to show by experimenting with *ncdtest* $\gamma = 0.10e-1$ that a partitioning may lead to poor performance if it does not take advantage of the divide-and-conquer nature of two-level iterative methods. For this particular partitioning, all diagonal blocks are of order 1 except one which is of order 2 (see number of blocks for $\gamma = 0.10e-1$ in Table A.2).

TABLE A.1
Characteristics of the Pushout Threshold Problem.

n	nz	symmetric nz structure
20,301	140,504	no

TABLE A.2
Partitioning Results for the easy Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-3$	308	1	5,050	$0.99e+0$
$0.10e-2$	4,060	1	162	$0.10e+1$
$0.10e-1$	20,300	1	2	$0.10e+1$

	number of blocks	last block size	degree of coupling
equal	143	137	$0.99e+0$
other	201	0	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.11e+0$	3	22	20,048	$0.11e+0$
$0.25e+0$	31	22	19,026	$0.25e+0$
$0.50e+0$	258	4	4,951	$0.99e+0$
$0.75e+0$	333	2	10,357	$0.99e+0$
$0.98e+0$	398	2	19,499	$0.99e+0$

The *newncd* partitionings are not competitive with *ncdtest*, *equal*, and *other* when BSOR is used though for all values of γ' they give solution in less than one minute. When IAD is used, the *newncd* partitioning for $\gamma' = 0.25e+0$ is rather competitive with *ncdtest* partitionings, but not with *equal* and *other* partitionings. For the other two values of γ' for the *newncd* partitioning, IAD fails to produce a solution due the reducibility of the corresponding coupling matrices. The results of CGS, GMRES, and BCGStab with ILU0 are very competitive for this test case. All Krylov subspace solvers except BCG converge with all the preconditioners used, however they require longer time, in most of the converging cases, than SOR and BSOR due to the preconditioning time overhead for ILUTH(10^{-3}), ILUTH(10^{-5}), and ILUK(10). We should remark that CGS and BCGStab

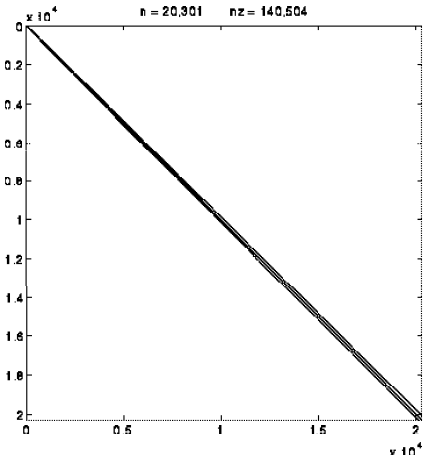

FIGURE A.1 *Pushout Threshold.*

TABLE A.3
Partitioning Results for the medium Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-2$	4	1	20,295	$0.95e+0$
$0.10e-1$	720	1	19,477	$0.95e+0$
$0.10e+0$	20,300	1	2	$0.95e+0$

	number of blocks	last block size	degree of coupling
equal	143	137	$0.93e+0$
other	201	0	$0.55e+0$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.23e+0$	77	3	14,198	$0.72e+0$
$0.25e+0$	86	3	10,220	$0.71e+0$
$0.30e+0$	67	4	7,430	$0.69e+0$
$0.35e+0$	46	4	10,901	$0.73e+0$
$0.50e+0$	2	2	20,299	$0.95e+0$

with ILUK(10) converge in only 1 iteration whereas 2 iterations are necessary for GMRES and DQGMRES. This suggests that the preconditioned matrix formed by ILUK (10) is very well-conditioned.

The *medium* and *hard* test matrices seem to be somewhat more "stiff" versions of the *easy* test matrix although the smallest degree of coupling that is computed for our choices of γ' for both of these matrices turns out to be larger than that of the *easy* test matrix (see Tables A.2, A.3, and A.4). Comparing the results of *easy*, *medium*, and *hard* test matrices, we see that Krylov subspace solvers perform worse as we move from *easy* to *hard* (see Tables A.6, A.7, and A.8). Nevertheless, BCGStab is affected the least among them and it becomes the next to best solver for the *hard* test matrix due to the poor performance of its competitors. IAD with the *newncd* $\gamma' = 0.20e+0$ partitioning is the winner for the *hard* test matrix. BCG converges for only one test matrix (i.e., *hard*) of the pushout threshold problem. The residual infinity norm that comes as a byproduct of BCG is observed to be unstable and too much oscillating which illustrates the irregular convergence behavior of BCG. It converges for the *hard* test matrix with ILUTH(10^{-5}) like all other Krylov subspace solvers since the chosen threshold value gives a dense and strong preconditioner. From the pushout threshold example, it seems that "stiffness" affects the performance of Krylov subspace solvers and SOR adversely. Performance degradation is also observed for BSOR. On the other hand, even though the solution time might have increased, the number of iterations taken to convergence by BSOR and IAD with the *ncdtest* and *newncd* partitionings for a fixed decomposability parameter decrease when we go from *easy* to *medium*, then to *hard*. The behavior of the two-level solvers with the *equal* and *other* partitionings is rather irregular, and IAD with the *equal* partitioning fails due to a reducible coupling matrix for the *hard* test matrix although

TABLE A.4
Partitioning Results for the hard Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-3$	2	1	20,300	$0.99e+0$
$0.10e-2$	134	1	20,168	$0.99e+0$
$0.10e-1$	2,286	1	18,016	$0.99e+0$

	number of blocks	last block size	degree of coupling
equal	143	137	$0.96e+0$
other	201	0	$0.91e+0$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.82e-1$	2	12	20,289	$0.26e+0$
$0.20e+0$	201	3	4,860	$0.38e+0$
$0.50e+0$	201	2	12,306	$0.68e+0$
$0.90e+0$	2	2	20,299	$0.99e+0$

the same partitioning had provided a winner with BSOR for *medium*.

TABLE A.5
Lower, Higher Bandwidths and Coefficients of Asymmetry of the pushout Test Matrices.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
easy	201	201	(2.97,1.96)
medium	201	201	(1.90,0.94)
hard	201	201	(1.97,0.99)

TABLE A.6

Numerical Results for easy ($n = 20,301$, $nz = 140,504$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.71e-10$	4.8	31	$0.69e-12$	$0.21e-12$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-3$	1.0	$0.61e-11$	6.1	7	$0.60e-15$	$0.18e-15$	0
$\gamma = 0.10e-2$	1.0	$0.28e-10$	2.1	7	$0.15e-14$	$0.47e-15$	0
$\gamma = 0.10e-1$	1.0	$0.36e-10$	8.0	24	$0.34e-12$	$0.10e-12$	0
equal	1.0	$0.11e-10$	3.9	7	$0.10e-14$	$0.31e-15$	0
other	1.0	$0.84e-11$	1.9	7	$0.81e-15$	$0.25e-15$	0
$\gamma' = 0.25e+0$	1.1	$0.11e-10$	41.0	15	$0.21e-12$	$0.64e-13$	0
$\gamma' = 0.50e+0$	1.0	$0.52e-10$	26.1	52	$0.49e-12$	$0.15e-12$	0
$\gamma' = 0.75e+0$	1.1	$0.44e-10$	17.3	16	$0.42e-12$	$0.13e-12$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-3$	0.9	$0.65e-10$	22.1	37	$0.84e-13$	$0.26e-13$	0
$\gamma = 0.10e-2$	0.9	$0.99e-10$	249.0	52	$0.12e-12$	$0.36e-13$	0
$\gamma = 0.10e-1$	0.9	requires unreasonably long time (large coupling matrix)					
equal	0.9	$0.90e-10$	16.4	41	$0.14e-12$	$0.42e-13$	0
other	0.9	$0.53e-10$	15.4	52	$0.10e-12$	$0.32e-13$	0
$\gamma' = 0.25e+0$	1.0	$0.14e-10$	34.5	9	$0.30e-13$	$0.91e-14$	0
$\gamma' = 0.50e+0$	1.0	failed (reducible coupling matrix)					
$\gamma' = 0.75e+0$	1.0	failed (reducible coupling matrix)					

TABLE A.6 continued
Numerical Results for easy ($n = 20,301$, $nz = 140,504$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	140,504	0.5	0.2
ILUTH(10^{-3})	110,010	6.1	0.5
ILUTH(10^{-5})	166,503	6.3	0.8
ILUK(10)	201,187	18.5	9.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	2.6	7	$0.65e-12$	$0.23e-15$	$0.69e-16$
	ILUTH(10^{-3})	1.8	5	$0.65e-13$	$0.40e-13$	$0.12e-13$
	ILUTH(10^{-5})	1.4	3	$0.26e-14$	$0.27e-14$	$0.82e-15$
	ILUK(10)	1.0	2	$0.60e-12$	$0.26e-13$	$0.81e-14$
DQGMRES ($k = 20$)	ILU0	4.1	7	$0.65e-12$	$0.23e-15$	$0.69e-16$
	ILUTH(10^{-3})	3.0	5	$0.65e-13$	$0.40e-13$	$0.12e-13$
	ILUTH(10^{-5})	2.2	3	$0.26e-14$	$0.27e-14$	$0.82e-15$
	ILUK(10)	1.8	2	$0.60e-12$	$0.26e-13$	$0.81e-14$
BCG	ILU0	297.9	500*	$0.13e-06$	$0.40e-09$	$0.12e-09$
	ILUTH(10^{-3})	270.6	500*	$0.46e-03$	$0.51e-04$	$0.16e-04$
	ILUTH(10^{-5})	308.5	500*	$0.13e-07$	$0.17e-08$	$0.51e-09$
	ILUK(10)	348.0	500*	$0.13e-01$	$0.19e-02$	$0.85e-03$
CGS	ILU0	2.4	4	$0.62e-10$	$0.37e-13$	$0.11e-13$
	ILUTH(10^{-3})	1.7	3	$0.45e-13$	$0.26e-13$	$0.79e-14$
	ILUTH(10^{-5})	1.5	2	$0.33e-16$	$0.21e-16$	$0.63e-17$
	ILUK(10)	0.8	1	$0.13e-10$	$0.31e-12$	$0.95e-13$
BCGStab	ILU0	2.6	4	$0.47e-12$	$0.96e-15$	$0.29e-15$
	ILUTH(10^{-3})	1.5	3	$0.11e-10^{s}$	$0.64e-11$	$0.19e-11$
	ILUTH(10^{-5})	0.8	1	$0.17e-10$	$0.25e-10$	$0.75e-11$
	ILUK(10)	0.8	1	$0.11e-12$	$0.58e-14$	$0.18e-14$
QMR	ILU0	5.7	8	$0.42e-08$	$0.43e-11$	$0.13e-11$
	ILUTH(10^{-3})	4.0	6	$0.44e-09$	$0.32e-09$	$0.99e-10$
	ILUTH(10^{-5})	3.6	4	$0.52e-08$	$0.78e-08$	$0.24e-08$
	ILUK(10)	4.5	5	$0.46e-10$	$0.18e-10$	$0.55e-11$

TABLE A.7
Numerical Results for medium ($n = 20,301$, $nz = 140,504$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.1	$0.68e-10$	51.4	352	$0.40e-10$	$0.44e-10$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-2$	1.0	$0.11e-15$	211.9	2	$0.21e-16$	$0.23e-16$	0
$\gamma = 0.10e-1$	1.0	$0.93e-10$	343.0	24	$0.21e-10$	$0.23e-10$	1
$\gamma = 0.10e+0$	1.1	$0.97e-10$	19.6	57	$0.34e-11$	$0.37e-11$	0
equal	1.0	$0.18e-11$	4.5	2	$0.28e-15$	$0.31e-15$	0
other	1.2	$0.38e-10$	43.5	218	$0.17e-10$	$0.18e-10$	0
$\gamma' = 0.25e+0$	1.9	$0.34e-10$	27.8	8	$0.62e-11$	$0.68e-11$	0
$\gamma' = 0.30e+0$	1.3	$0.26e-10$	77.8	78	$0.47e-11$	$0.52e-11$	0
$\gamma' = 0.35e+0$	1.2	$0.47e-10$	105.3	70	$0.91e-11$	$0.10e-10$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-2$	1.0	$0.56e-16$	211.0	2	$0.31e-16$	$0.34e-16$	0
$\gamma = 0.10e-1$	1.0	$0.99e-10$	318.8	27	$0.36e-10$	$0.40e-10$	1
$\gamma = 0.10e+0$	1.0	requires unreasonably long time (large coupling matrix)					
equal	1.0	$0.59e-15$	5.5	2	$0.59e-16$	$0.65e-16$	0
other	1.0	$0.40e-10$	4.7	11	$0.72e-11$	$0.79e-11$	0
$\gamma' = 0.25e+0$	1.3	$0.94e-10$	27.3	6	$0.31e-12$	$0.34e-12$	0
$\gamma' = 0.30e+0$	1.2	$0.86e-10$	37.9	24	$0.13e-11$	$0.15e-11$	0
$\gamma' = 0.35e+0$	1.2	$0.55e-10$	58.7	24	$0.29e-11$	$0.32e-11$	0

TABLE A.7 continued

Numerical Results for medium ($n = 20,301$, $nz = 140,504$.

Preconditioner	$nzlu$	Time	MFlops
ILU0	140,504	0.5	0.2
ILUTH(10^{-3})	275,253	6.4	1.5
ILUTH(10^{-5})	554,148	8.8	5.0
ILUK(10)	201,189	19.8	10.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	29.9	64	$0.86e-10^{\dagger}$	$0.23e-10$	$0.25e-10$
	ILUTH(10^{-3})	4.9	9	$0.15e-09$	$0.45e-16$	$0.50e-16$
	ILUTH(10^{-5})	4.6	7	$0.22e-08$	$0.30e-11$	$0.33e-11$
	ILUK(10)	5.4	12	$0.73e-11$	$0.12e-13$	$0.13e-13$
DQGMRES ($k = 20$)	ILU0	63.4	72	$0.13e-11$	$0.40e-12$	$0.44e-12$
	ILUTH(10^{-3})	13.8	19	$0.82e-04^{\ddagger}$	$0.86e-08$	$0.95e-08$
	ILUTH(10^{-5})	7.4	9	$0.11e-05^{\ddagger}$	$0.76e-06$	$0.84e-06$
	ILUK(10)	7.3	12	$0.73e-11$	$0.12e-13$	$0.13e-13$
BCG	ILU0	298.7	500*	$0.17e+03$	$0.15e-02$	$0.12e+00$
	ILUTH(10^{-3})	396.7	500*	$0.39e-02$	$0.26e-08$	$0.28e-08$
	ILUTH(10^{-5})	598.1	500*	$0.21e-01$	$0.40e-09$	$0.44e-09$
	ILUK(10)	343.4	500*	$0.15e+00$	$0.10e-03$	$0.12e-03$
CGS	ILU0	291.6	500*	$0.16e+15$	$0.23e-02$	$0.19e+00$
	ILUTH(10^{-3})	4.0	5	$0.17e-10$	$0.35e-16$	$0.38e-16$
	ILUTH(10^{-5})	577.5	500*	$0.24e+05$	$0.13e-08$	$0.15e-08$
	ILUK(10)	5.5	8	$0.11e-11$	$0.19e-14$	$0.20e-14$
BCGStab	ILU0	17.6	30	$0.35e-10^{s}$	$0.92e-11$	$0.10e-10$
	ILUTH(10^{-3})	3.2	4	$0.14e-10$	$0.73e-16$	$0.80e-16$
	ILUTH(10^{-5})	2.5	2	$0.59e-11$	$0.42e-16$	$0.46e-16$
	ILUK(10)	4.5	7	$0.34e-10^{s}$	$0.57e-13$	$0.63e-13$
QMR	ILU0	366.3	500*	$0.25e-05$	$0.35e-06$	$0.39e-06$
	ILUTH(10^{-3})	6.5	7	$0.76e-06$	$0.30e-12$	$0.33e-12$
	ILUTH(10^{-5})	663.8	500*	$0.92e-10$	$0.12e-12$	$0.13e-12$
	ILUK(10)	9.8	12	$0.12e-07$	$0.20e-10$	$0.22e-10$

TABLE A.8

Numerical Results for hard ($n = 20,301$, $nz = 140,504$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.57e-06$	147.0	$1,000^*$	$0.97e-07$	$0.42e-04$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-3$	1.0	$0.29e-16$	119.9	2	$0.41e-16$	$0.46e-14$	0
$\gamma = 0.10e-2$	1.0	$0.96e-15$	181.7	2	$0.23e-17$	$0.26e-15$	0
$\gamma = 0.10e-1$	1.0	$0.68e-10$	122.0	8	$0.59e-14$	$0.67e-12$	0
equal	1.0	$0.77e-06$	338.7	$1,000^*$	$0.33e-07$	$0.52e-05$	0
other	1.0	$0.78e-06$	185.1	$1,000^*$	$0.72e-07$	$0.23e-04$	0
$\gamma' = 0.20e+0$	1.6	$0.90e-10$	54.6	146	$0.15e-10$	$0.17e-08$	0
$\gamma' = 0.50e+0$	1.2	$0.64e-10$	74.7	27	$0.46e-13$	$0.52e-11$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-3$	1.0	$0.20e-16$	120.0	2	$0.41e-16$	$0.46e-16$	0
$\gamma = 0.10e-2$	1.0	$0.36e-15$	182.0	2	$0.11e-17$	$0.13e-15$	0
$\gamma = 0.10e-1$	1.0	$0.32e-10$	304.5	8	$0.29e-14$	$0.33e-12$	0
equal	1.0	failed (reducible coupling matrix)					
other	1.3	$0.98e-10$	37.0	141	$0.48e-11$	$0.54e-09$	0
$\gamma' = 0.20e+0$	1.4	$0.91e-10$	29.8	59	$0.12e-11$	$0.14e-09$	0
$\gamma' = 0.50e+0$	1.1	$0.18e-10$	60.2	13	$0.35e-13$	$0.40e-11$	0

TABLE A.8 continued
Numerical Results for hard ($n = 20,301$, $nz = 140,504$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	140,504	0.5	0.2
ILUTH(10^{-3})	860,386	12.4	10.5
ILUTH(10^{-5})	1,750,723	42.4	50.3
ILUK (10)	201,187	17.7	8.4

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	207.1	480	$0.96e-10$	$0.45e-10$	$0.52e-08$
	ILUTH(10^{-3})	35.2	40	$0.13e-11$	$0.38e-12$	$0.43e-10$
	ILUTH(10^{-5})	22.8	16	$0.40e-11$	$0.79e-12$	$0.90e-10$
	ILUK(10)	163.2	340	$0.86e-10$	$0.34e-10$	$0.38e-08$
DQGMRES ($k = 20$)	ILU0	479.2	500*	$0.72e-09$	$0.34e-09$	$0.38e-07$
	ILUTH(10^{-3})	50.3	41	$0.19e-12$	$0.57e-13$	$0.65e-11$
	ILUTH(10^{-5})	25.5	16	$0.40e-11$	$0.79e-12$	$0.90e-10$
	ILUK (10)	491.0	500*	$0.21e-11$	$0.99e-12$	$0.11e-09$
BCG	ILU0	297.1	500*	$0.82e+06$	$0.14e-02$	$0.64e-01$
	ILUTH(10^{-3})	839.7	500*	$0.47e+00$	$0.67e-03$	$0.55e-01$
	ILUTH(10^{-5})	171.0	58	$0.29e-10$	$0.18e-10$	$0.21e-08$
	ILUK(10)	346.0	500*	$0.74e+03$	$0.77e-03$	$0.57e-01$
CGS	ILU0	291.2	500*	$0.30e+07$	$0.34e-03$	$0.98e-01$
	ILUTH(10^{-3})	37.2	23	$0.14e-10$	$0.42e-11$	$0.48e-09$
	ILUTH(10^{-5})	34.3	12	$0.60e-10$	$0.12e-10$	$0.14e-08$
	ILUK(10)	337.8	500*	$0.11e+15$	$0.11e-02$	$0.12e+00$
BCGStab	ILU0	43.1	73	$0.31e-10^{s}$	$0.15e-10$	$0.17e-08$
	ILUTH(10^{-3})	22.7	14	$0.98e-10$	$0.29e-10$	$0.33e-08$
	ILUTH(10^{-5})	24.4	9	$0.60e-10^{s}$	$0.12e-10$	$0.14e-08$
	ILUK (10)	40.2	59	$0.55e-10^{s}$	$0.26e-10$	$0.30e-08$
QMR	ILU0	358.6	500*	$0.27e-04$	$0.25e-04$	$0.30e-01$
	ILUTH(10^{-3})	50.4	28	$0.21e-07$	$0.63e-08$	$0.71e-06$
	ILUTH(10^{-5})	57.1	18	$0.41e-08$	$0.83e-09$	$0.94e-07$
	ILUK(10)	410.6	500*	$0.26e-04$	$0.21e-04$	$0.44e-01$

B. A Two-Dimensional Markov Chain Model. As the title suggests, in this problem we consider a two-dimensional Markov chain. In the first dimension of the chain, the state variable assumes all values from 0 through N_x. Similarly, in the second dimension, the state variable takes on values from 0 through N_y. The state space is sketched in Figure B.1.

This two-dimensional Markov chain model allows for transitions from any non-boundary state to adjacent states in the North, South, East, West, North-East, North-West, South-East, and South-West directions. However, in the model we used in our experiments, only transitions to the South, East and North-West are permitted (taking the others to be 0). From any non-boundary state (u, v), transitions to the South are assigned the value v, transitions to the East are assigned the value 2025.0, and transitions to the North-West are assigned the value u [33]. The state space of the Markov chain is of size $(N_x + 1)(N_y + 1)$. The values of N_x and N_y are both set to 128, yielding a matrix, $2D$, of order $n = 16,641$ and number of nonzero elements $nz = 66,049$. The partitioning results of the $2D$ test matrix are illustrated in Table B.2. The time taken by the *ncdtest* and *newncd* partitioning algorithms does not exceed 1 second for the $2D$ matrix. Interestingly, for this test matrix the *newncd* algorithm finds a partition with equal size blocks and a degree of coupling of $0.11e + 0$. The characteristics of the test matrix and bandwidth information are reported in Tables B.1 and B.3.

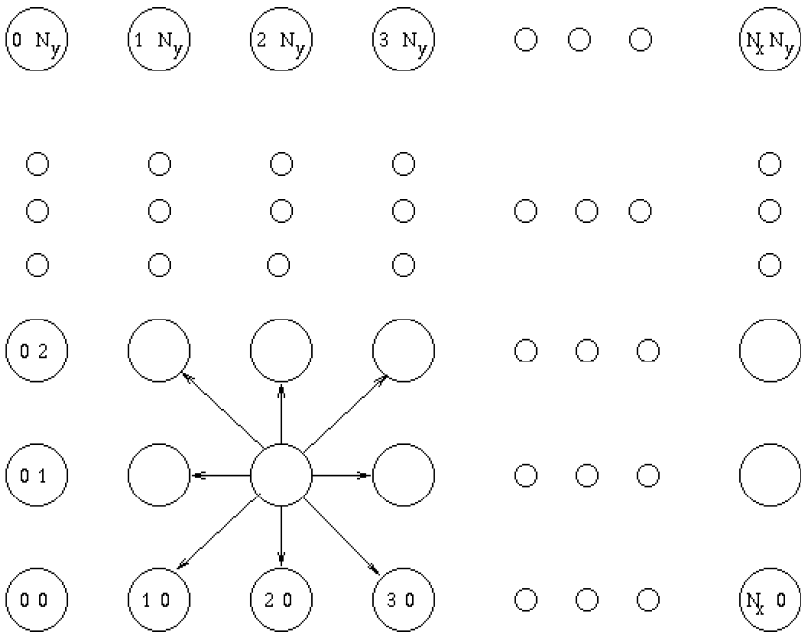

FIGURE B.1 *A Two-Dimensional Markov Chain Model Model.*

TABLE B.1
Characteristics of the Two-Dimensional Markov Chain Problem.

n	nz	symmetric nz structure
16,641	66,049	no

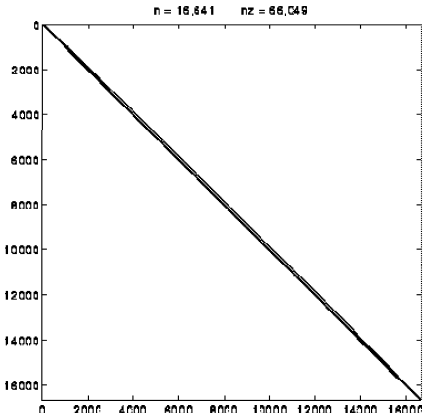

FIGURE B.2 *2D.*

TABLE B.2
Partitioning Results for the 2D Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-2$	513	1	16,129	$0.94e+0$
$0.10e-1$	5,913	1	11,449	$0.95e+0$

	number of blocks	last block size	degree of coupling
equal	129	0	$0.98e+0$
other	182	170	$0.97e+0$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.57e-1$	129	129	129	$0.11e+0$
$0.88e+0$	129	129	129	$0.11e+0$

TABLE B.3
Lower, Higher Bandwidths and Coefficient of Asymmetry of the 2D Test Matrix.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
$2D$	65	129	(2.00,1.00)

For the $2D$ test matrix, BCGStab with ILUTH (10^{-5}) outperforms all other solvers (see Table B.4). CGS with ILUTH(10^{-5}) comes a close second. GMRES with ILUTH(10^{-5}) and BCGStab with ILUTH(10^{-3}) follow as close third. The major inconvenience of BCG that is observed in $2D$ and most of the test cases is its convergence behavior. The residual infinity norm of BCG tends to decrease in the first few iterations but stagnates or increases thereafter. This behavior is observed in most of the test matrices with all the preconditioners used. The residual may not be an accurate indicator of the number of correct digits in the approximate solution (see [24], p. 1168). Note also that the ILU0 preconditioner is generally quite weak for the $2D$ test matrix. As for

two-level solvers, it can be seen from Table B.4 that BSOR and IAD require relatively long time to converge with the *ncdtest* partitionings $\gamma = 0.10e-2$ and $\gamma = 0.10e-1$, although they both take 2 iterations to converge. The *ncdtest* algorithm with $\gamma = 0.10e-2$ partitions the matrix to 512 diagonal blocks of order 1 and one last block of order 16,129, and similarly *ncdtest* with $\gamma = 0.10e-1$ partitions the matrix to 5,192 diagonal blocks of order 1 and one last block of order 11,449. These partitionings do not take advantage of the divide-and-conquer nature of two-level iterative methods; we can easily infer that the 2 iterations are entirely used to solve the large block in each partitioning. This may explain the relation between the low iteration count and the long time to converge. The *equal* and *other* partitionings take larger number of iterations but converge in less time as they partition the matrix more uniformly. IAD with the *ncdnew* $\gamma' = 0.57e-1$ partitioning is the fastest two-level solver for this problem. Also, the same *newncd* partitioning is much better than those of *ncdtest* and it outperforms *equal* and *other* partitionings for BSOR. Finally, the performance of SOR is poor for this test matrix.

TABLE B.4
Numerical Results for 2D ($n = 16,641$, $nz = 66,049$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.4	$0.96e-10$	29.6	314	$0.71e-11$	$0.57e-10$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-2$	1.0	$0.26e-15$	62.3	2	$0.22e-17$	$0.17e-16$	0
$\gamma = 0.10e-1$	1.0	$0.57e-15$	35.1	2	$0.30e-17$	$0.24e-16$	0
equal	1.3	$0.95e-10$	33.9	211	$0.72e-11$	$0.58e-10$	0
other	1.2	$0.98e-10$	35.1	205	$0.79e-11$	$0.63e-10$	0
$\gamma' = 0.57e-1$	1.8	$0.87e-10$	22.4	180	$0.15e-11$	$0.12e-10$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-2$	1.0	$0.74e-15$	54.9	2	$0.22e-17$	$0.17e-16$	0
$\gamma = 0.10e-1$	1.0	$0.37e-15$	106.6	2	$0.22e-17$	$0.17e-16$	0
equal	1.1	$0.77e-10$	9.9	43	$0.23e-11$	$0.18e-10$	0
other	1.0	$0.88e-10$	11.6	51	$0.30e-11$	$0.24e-10$	0
$\gamma' = 0.57e-1$	1.0	$0.84e-10$	5.6	27	$0.62e-11$	$0.49e-10$	0

TABLE B.4 continued
Numerical Results for 2D ($n = 16,641$, $nz = 66,049$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	66,049	0.2	0.05
ILUTH(10^{-3})	138,392	2.7	0.60
ILUTH(10^{-5})	250,897	3.3	1.40
ILUK(10)	165,819	9.0	4.80

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	67.4	240	$0.63e-10$	$0.26e-09$	$0.20e-08$
	ILUTH(10^{-3})	2.9	10	$0.54e-13$	$0.79e-14$	$0.63e-13$
	ILUTH(10^{-5})	1.6	4	$0.97e-13$	$0.15e-12$	$0.12e-11$
	ILUK(10)	4.2	13	$0.17e-12$	$0.47e-12$	$0.37e-11$
DQGMRES ($k = 20$)	ILU0	139.9	200	$0.98e-12$	$0.17e-11$	$0.14e-10$
	ILUTH(10^{-3})	4.6	10	$0.54e-13$	$0.79e-14$	$0.63e-13$
	ILUTH(10^{-5})	2.4	4	$0.97e-13$	$0.15e-12$	$0.12e-11$
	ILUK(10)	6.5	13	$0.17e-12$	$0.47e-12$	$0.37e-11$
BCG	ILU0	166.6	500*	$0.77e+04$	$0.70e-02$	$0.99e-01$
	ILUTH(10^{-3})	230.5	500*	$0.36e-03$	$0.98e-04$	$0.10e-02$
	ILUTH(10^{-5})	2.0	3	$0.74e-10$	$0.34e-10$	$0.27e-09$
	ILUK(10)	252.1	500*	$0.18e+02$	$0.11e-02$	$0.15e-01$
CGS	ILU0	164.4	500*	$0.25e+14$	$0.11e-01$	$0.11e+00$
	ILUTH(10^{-3})	2.4	5	$0.42e-11$	$0.52e-12$	$0.41e-11$
	ILUTH(10^{-5})	1.3	2	$0.24e-11$	$0.11e-11$	$0.87e-11$
	ILUK(10)	4.1	8	$0.19e-10$	$0.32e-10$	$0.26e-09$
BCGStab	ILU0	12.7	38	$0.66e-10^{s}$	$0.70e-10$	$0.55e-09$
	ILUTH(10^{-3})	2.2	5	$0.36e-10^{s}$	$0.44e-11$	$0.35e-10$
	ILUTH(10^{-5})	1.0	2	$0.73e-10^{s}$	$0.75e-10$	$0.59e-09$
	ILUK(10)	3.8	8	$0.65e-11^{s}$	$0.11e-10$	$0.90e-10$
QMR	ILU0	217.3	500*	$0.30e-04$	$0.26e-04$	$0.12e-02$
	ILUTH(10^{-3})	6.2	11	$0.36e-08$	$0.43e-09$	$0.34e-08$
	ILUTH(10^{-5})	2.6	3	$0.74e-10$	$0.34e-10$	$0.27e-09$
	ILUK(10)	7.9	13	$0.15e-09$	$0.25e-09$	$0.20e-08$

C. An NCD Queueing Network of the Central Server Type. The model illustrated in Figure C.1 represents the system architecture of a time-shared, multiprogrammed, paged, virtual memory computer. The system [33] consists of

- a set of N_t terminals from which N_t users generate commands,
- a central processing unit (CPU),
- a secondary memory device (SM),
- a filing device (FD).

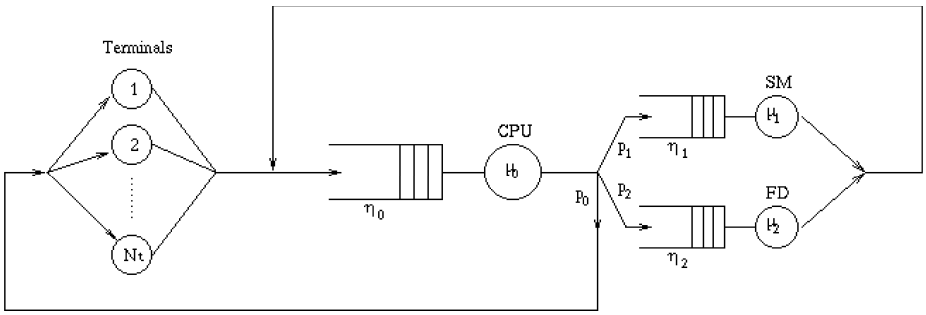

FIGURE C.1 *An NCD Queueing Network of the Central Server Type Model.*

A queue of requests is associated with each device and the scheduling is assumed to be FCFS (first-come, first-served). When a user generates a command at the terminal, it remains inactive until the system responds. Symbolically, this user enters the CPU queue. The system behaves in such a way that after a certain time period, called the compute time, either a page fault or an input/output (file request) occurs. In the case of a page fault, the process currently in the system enters the SM queue, otherwise, in the case of a file request, it joins the FD queue. Processes that terminate their service at the SM or FD queue return to the CPU queue. A command commit is symbolically represented by a departure of the process from the CPU to the terminals.

Let n_0, n_1 and n_2 respectively be the number of processes in the CPU, SM and FD queues at a certain time. Then the degree of multiprogramming at that moment is given by $\eta = n_0 + n_1 + n_2$. Let $(\mu_0(\eta))^{-1}$, $q(\eta)$ and $r(\eta)$ respectively denote the mean service time at the CPU, the mean compute time between two page faults, and the mean compute time between two input/output requests. It follows that the probabilities that a process leaving the CPU will be directed to the SM or to the FD queue are respectively given by $p_1(\eta) = (\mu_0(\eta)q(\eta))^{-1}$ and $p_2(\eta) = (\mu_0(\eta)r(\eta))^{-1}$. The probability that a process leaving the CPU to the terminals is given by $p_0(\eta) = (\mu_0(\eta)c(\eta))^{-1} = 1 - (p_1(\eta) + p_2(\eta))$, where $c(\eta)$ is the mean compute time of a process [24].

We assign a specific value to each parameter. The rate at which processes move from the CPU queue to the SM device is taken to be $p_1(\eta)\mu_0(\eta) = 100(\eta/128)^{1.5}$. The mean compute time between two input/output requests $r(\eta)$ is taken as 20 ms so that $p_2(\eta)\mu_0(\eta) = 0.05$, and the mean compute time of a process $c(\eta)$ is equal to 500 ms giving $p_0(\eta)\mu_0(\eta) = 0.002$. The mean think-time of a user at a terminal is estimated to be on the order of $\lambda^{-1} = 10$ s. The mean service time of the SM is taken as $(\mu_1(\eta))^{-1} = 5$ ms and that of the FD to be $(\mu_2(\eta))^{-1} = 30$ ms. The total number of users in the system, N_t, is

set to 50 yielding a matrix, *ncd*, of order $n = \begin{pmatrix} N_t + 4 - 1 \\ 4 - 1 \end{pmatrix} = 23,426$ and number of nonzero elements $nz = 156,026$.

Two more test cases are generated from this model. The first one, *ncd_alt*1, is obtained by setting the mean service time of the FD to $(\mu_2(\eta))^{-1} = 3,000\ s$. The second test case, *ncd_alt*2, is more ill-conditioned than *ncd_alt*1 and is generated by setting the mean think-time of a user at a terminal to $\lambda^{-1} = 10,000\ s$. The smallest degree of coupling we came across for the *ncd_alt*2 test matrix for the values of γ' we used is $0.81e-4$. Naturally, the three test matrices, *ncd*, *ncd_alt*1, and *ncd_alt*2, have the same order, number of nonzero elements, and nonzero structure. The characteristics, partitioning results, and bandwidths of all the test matrices for this model are reported in Tables C.1 through C.5. The time to partition the *ncd* test matrices using *ncdtest* and *newncd* does not exceed 1 and 1.8 seconds, respectively.

TABLE C.1
Characteristics of the NCD Queueing Network Problem.

n	nz	symmetric nz structure
23,426	256,026	yes

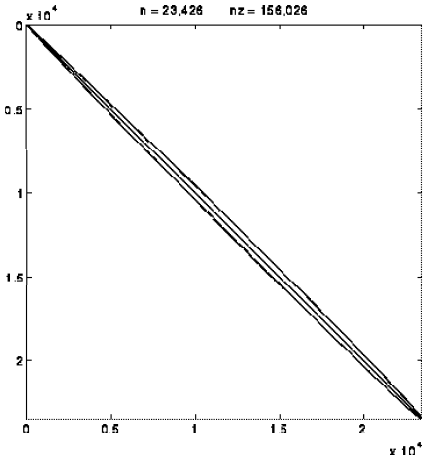

FIGURE C.2 *ncd.*

The *equal* partitioning of the *ncd* test matrix leads to a reducible coupling matrix causing IAD to fail (see Table C.6). So does *newncd* $\gamma' = 0.10e-2$ for the *ncd* and *ncd_alt*1 test matrices (see Tables C.6 and C.7). With the *ncdtest* $\gamma = 0.10e-3$ and *other* partitionings, IAD outperforms BSOR though both methods solve the same blocks iteratively. This shows the advantage of IAD over BSOR in solving the coupling matrix directly, in the aggregation step, when it is not too large. The winning solver for the *ncd* test matrix is, however, IAD with *newncd* $\gamma' = 0.10e+0$. For this partitioning, the coupling matrix is of order 1,221. This is also one of several test cases which demonstrates the superiority of BCGStab over other Krylov subspace solvers. The performance of QMR on this problem is commendable as well.

TABLE C.2
Partitioning Results for the ncd Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-4$	3	1,275	20,825	$0.85e-4$
$0.10e-3$	51	1	1,326	$0.28e-3$
$0.10e-2$	51	1	1,326	$0.28e-3$

	number of blocks	last block size	degree of coupling
equal	154	17	$0.10e+1$
other	216	206	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.81e-4$	21	528	5,456	$0.16e-3$
$0.10e-2$	51	1	1,326	$0.28e-3$
$0.10e+0$	1,221	2	326	$0.36e-2$
$0.98e+0$	51	2	22,101	$0.35e-2$

The performance of SOR in all parts of the *ncd* test problem is very poor and uninteresting.

The *ncd_alt*2 test matrix is a more ill-conditioned version of the *ncd* test matrix. The same cannot be said of *ncd_alt*1 since the smallest degree of coupling of the *newncd* partitionings for *ncd_alt*1 is the same as the corresponding one for *ncd*. See the degree of couplings for various partitionings of these three test matrices in Tables C.2, C.3, and C.4. Close investigation of the results in Tables C.6, C.7, and C.8 also confirms this situation. All Krylov subspace solvers with the ILUK(10) preconditioner perform better for *ncd_alt*1 than they do for *ncd*. For the other preconditioners, the performance of Krylov subspace solvers become worse. Note that for *ncd_alt*1, the ILUK preconditioner preserves the same density, but the ILUTH preconditioners become sparser. On the other hand, the performance of two-level solvers in general improve considerably for *ncd_alt*1, BSOR with the *other* partitioning becoming the fastest solver. BSOR with the *ncdtest* $\gamma = 0.10e-3$ partitioning comes a close second (see Table C.7). As for the *ncd_alt*2 test matrix, Krylov subspace solvers show very poor performance, possibly because the coefficient matrix is more ill-conditioned than both *ncd* and *ncd_alt*1 and the preconditioners used are not sufficiently robust. We see improvements among the two-level solvers with the *equal* and *newncd* $\gamma' = 0.10e+0$ partitionings. BSOR with the *equal* partitioning is the winner for this test matrix followed by BSOR with *newncd* $\gamma' = 0.10e+0$ as a close second. The decomposability parameter $0.10e-7$ with *ncdtest* partitions the *ncd_alt*2 test matrix to 3 diagonal blocks of order 1,275, 1,326 and 20,825. This unbalanced partitioning and solving the largest block iteratively with a tolerance of 10^{-3} are most likely the reasons behind the poor performance of BSOR and IAD for this particular partitioning (see Table C.8).

TABLE C.3
Partitioning Results for the ncd_alt1 Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-6$	51	1	1,326	$0.20e-2$
$0.10e-5$	51	1	1,326	$0.20e-2$
$0.10e-4$	150	1	1,225	$0.21e-2$
$0.10e-3$	1,326	1	51	$0.23e-2$
$0.10e-2$	1,326	1	51	$0.23e-2$

	number of blocks	last block size	degree of coupling
equal	154	17	$0.10e+1$
other	216	206	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.82e-4$	21	528	5,456	$0.16e-3$
$0.10e-2$	51	1	1,326	$0.28e-3$
$0.10e+0$	1,221	2	326	$0.23e-2$
$0.98e+0$	51	2	22,101	$0.21e-2$

TABLE C.4
Partitioning Results for the ncd_alt2 Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-7$	3	1,275	20,825	$0.81e-4$
$0.10e-6$	25	406	3,654	$0.81e-4$
$0.10e-5$	51	1	1,326	$0.81e-4$
$0.10e-4$	51	1	1,326	$0.81e-4$
$0.10e-3$	51	1	1,326	$0.81e-4$
$0.10e-2$	51	1	1,326	$0.81e-4$

	number of blocks	last block size	degree of coupling
equal	154	17	$0.10e+1$
other	216	206	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.81e-4$	51	1	1,326	$0.81e-4$
$0.10e-1$	1,273	2	58	$0.35e-2$
$0.10e+0$	1,221	2	326	$0.35e-2$
$0.98e+0$	51	2	22,101	$0.35e-2$

TABLE C.5
Lower, Higher Bandwidths and Coefficients of Asymmetry of the ncd Test Matrices.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
ncd	460	460	(2.00,0.98)
*ncd_alt*1	460	460	(2.00,0.98)
*ncd_alt*2	460	460	(2.00,0.98)

TABLE C.6
Numerical Results for ncd ($n = 23,426$, $nz = 156,026$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.36e-04$	173.4	$1,000^*$	$0.30e-06$	$0.18e-05$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-3$	1.0	$0.45e-10$	27.2	21	$0.23e-16$	$0.14e-16$	0
equal	1.0	$0.12e-04$	222.1	$1,000^*$	$0.54e-07$	$0.35e-07$	0
other	1.0	$0.97e-04$	208.1	$1,000^*$	$0.33e-06$	$0.30e-06$	0
$\gamma' = 0.10e-2$	1.0	$0.49e-10$	25.3	21	$0.29e-16$	$0.17e-16$	0
$\gamma' = 0.10e+0$	1.4	$0.42e-10$	18.9	80	$0.57e-14$	$0.34e-14$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-3$	1.0	$0.60e-13$	21.5	5	$0.39e-16$	$0.23e-16$	0
equal	1.0	failed (reducible coupling matrix)					
other	1.4	$0.98e-10$	187.2	623	$0.17e-12$	$0.10e-12$	0
$\gamma' = 0.10e-2$	1.0	failed (reducible coupling matrix)					
$\gamma' = 0.10e+0$	1.0	$0.17e-11$	10.5	7	$0.38e-16$	$0.23e-16$	0

TABLE C.6 continued
Numerical Results for ncd ($n = 23,426$, $nz = 156,026$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	156,026	0.5	0.2
ILUTH(10^{-3})	154,747	19.2	0.7
ILUTH(10^{-5})	282,825	19.8	1.5
ILUK(10)	233,882	40.3	14.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	248.2	500*	$0.45e-05$	$0.14e-05$	$0.29e-05$
	ILUTH(10^{-3})	253.7	500*	$0.19e-06$	$0.57e-07$	$0.35e-07$
	ILUTH(10^{-5})	5.9	10	$0.58e-14$	$0.42e-15$	$0.25e-15$
	ILUK(10)	177.4	321	$0.48e-08$	$0.38e-07$	$0.23e-07$
DQGMRES ($k = 20$)	ILU0	547.9	500*	$0.38e-05$	$0.11e-05$	$0.14e-05$
	ILUTH(10^{-3})	533.8	500*	$0.58e-09$	$0.17e-09$	$0.10e-09$
	ILUTH(10^{-5})	7.9	10	$0.58e-14$	$0.42e-15$	$0.25e-15$
	ILUK(10)	574.1	500*	$0.52e-07$	$0.99e-06$	$0.21e-05$
BCG	ILU0	339.7	500*	$0.82e-01$	$0.22e-04$	$0.91e-03$
	ILUTH(10^{-3})	337.9	500*	$0.17e-01$	$0.52e-04$	$0.18e-03$
	ILUTH(10^{-5})	16.7	18	$0.83e-11$	$0.12e-10$	$0.74e-11$
	ILUK(10)	404.5	500*	$0.46e-01$	$0.21e-05$	$0.11e-05$
CGS	ILU0	329.5	500*	$0.58e+03$	$0.25e-04$	$0.24e-03$
	ILUTH(10^{-3})	328.7	500*	$0.93e+05$	$0.12e-04$	$0.26e-04$
	ILUTH(10^{-5})	4.7	5	$0.17e-10$	$0.12e-11$	$0.75e-12$
	ILUK(10)	54.6	69	$0.17e-11$	$0.57e-11$	$0.34e-11$
BCGStab	ILU0	45.9	69	$0.82e-10^{s}$	$0.20e-11$	$0.12e-11$
	ILUTH(10^{-3})	36.5	55	$0.50e-10^{s}$	$0.62e-11$	$0.37e-11$
	ILUTH(10^{-5})	3.8	4	$0.78e-10$	$0.57e-11$	$0.34e-11$
	ILUK(10)	102.9	129	$0.40e-08$	$0.36e-07$	$0.22e-07$
QMR	ILU0	409.0	500*	$0.65e-05$	$0.29e-05$	$0.28e-04$
	ILUTH(10^{-3})	63.8	78	$0.13e-07$	$0.90e-09$	$0.54e-09$
	ILUTH(10^{-5})	12.5	11	$0.27e-08$	$0.20e-09$	$0.12e-09$
	ILUK(10)	86.7	91	$0.54e-07$	$0.13e-05$	$0.80e-06$

TABLE C.7
Numerical Results for ncd_alt1 ($n = 23,426$, $nz = 156,026$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.39e-03$	167.7	$1,000^*$	$0.80e-08$	$0.87e-08$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-6$	1.0	$0.38e-11$	15.6	7	$0.48e-16$	$0.24e-16$	0
$\gamma = 0.10e-4$	1.0	$0.14e-10$	19.4	16	$0.41e-16$	$0.21e-16$	0
$\gamma = 0.10e-3$	1.0	$0.18e-10$	4.5	17	$0.39e-16$	$0.20e-16$	0
equal	1.0	$0.35e-03$	222.9	$1,000^*$	$0.45e-07$	$0.71e-07$	0
other	1.0	$0.71e-10$	2.4	9	$0.78e-15$	$0.39e-15$	0
$\gamma = 0.10e-2$	1.0	$0.63e-10$	19.5	14	$0.59e-17$	$0.29e-17$	0
$\gamma = 0.10e+0$	1.0	$0.61e-10$	6.1	23	$0.76e-16$	$0.38e-16$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-6$	1.0	$0.34e-11$	19.2	4	$0.48e-16$	$0.24e-16$	0
$\gamma = 0.10e-4$	1.0	$0.13e-10$	16.6	3	$0.48e-16$	$0.24e-16$	0
$\gamma = 0.10e-3$	1.0	$0.12e-11$	4.9	3	$0.48e-16$	$0.24e-16$	0
equal	1.0	$0.31e-05$	285.8	$1,000^*$	$0.45e-08$	$0.22e-08$	0
other	1.0	$0.46e-10$	5.8	7	$0.31e-15$	$0.15e-15$	0
$\gamma = 0.10e-2$	1.0	failed (reducible coupling matrix)					
$\gamma = 0.10e+0$	1.0	$0.56e-10$	33.7	28	$0.52e-16$	$0.26e-16$	0

TABLE C.7 continued

Numerical Results for ncd_alt1 ($n = 23,426$, $nz = 156,026$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	156,026	0.5	0.2
ILUTH(10^{-3})	89,732	19.1	0.4
ILUTH(10^{-5})	198,984	19.7	1.0
ILUK(10)	234,088	30.1	4.1

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	248.6	500*	$0.31e-06$	$0.20e-07$	$0.12e-07$
	ILUTH(10^{-3})	224.0	500*	$0.13e-05$	$0.31e-06$	$0.24e-06$
	ILUTH(10^{-5})	7.1	13	$0.14e-12$	$0.59e-15$	$0.29e-15$
	ILUK(10)	7.8	14	$0.21e-14^{\dagger}$	$0.26e-14$	$0.13e-14$
DQGMRES ($k = 20$)	ILU0	554.8	500*	$0.81e-06$	$0.94e-07$	$0.67e-07$
	ILUTH(10^{-3})	537.0	500*	$0.51e-06$	$0.74e-07$	$0.46e-07$
	ILUTH(10^{-5})	9.8	13	$0.14e-12$	$0.59e-15$	$0.29e-15$
	ILUK(10)	17.2	20	$0.11e-08^{\ddagger}$	$0.72e-09$	$0.36e-09$
BCG	ILU0	339.0	500*	$0.64e-03$	$0.74e-07$	$0.44e-07$
	ILUTH(10^{-3})	280.2	500*	$0.34e-06$	$0.19e-09$	$0.95e-10$
	ILUTH(10^{-5})	15.3	19	$0.60e-10$	$0.40e-09$	$0.20e-09$
	ILUK(10)	9.0	11	$0.44e-11$	$0.60e-13$	$0.30e-13$
CGS	ILU0	328.9	500*	$0.16e+15$	$0.12e-05$	$0.52e-05$
	ILUTH(10^{-3})	271.2	500*	$0.18e+10$	$0.13e-05$	$0.62e-05$
	ILUTH(10^{-5})	5.7	7	$0.16e-11$	$0.67e-14$	$0.33e-14$
	ILUK(10)	3.3	4	$0.62e-10$	$0.48e-16$	$0.24e-16$
BCGStab	ILU0	55.4	83	$0.49e-10^{s}$	$0.13e-13$	$0.63e-14$
	ILUTH(10^{-3})	65.3	117	$0.21e-10^{s}$	$0.93e-14$	$0.46e-14$
	ILUTH(10^{-5})	4.5	6	$0.53e-10^{s}$	$0.22e-12$	$0.11e-12$
	ILUK(10)	3.3	4	$0.40e-10$	$0.47e-16$	$0.23e-16$
QMR	ILU0	408.8	500*	$0.13e-04$	$0.41e-05$	$0.15e-03$
	ILUTH(10^{-3})	151.3	218	$0.82e-08$	$0.15e-11$	$0.73e-12$
	ILUTH(10^{-5})	11.0	11	$0.13e-08$	$0.56e-11$	$0.28e-11$
	ILUK(10)	10.8	11	$0.43e-11$	$0.78e-13$	$0.39e-13$

TABLE C.8

Numerical Results for ncd_alt2 ($n = 23,426$, $nz = 156,026$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.28e-03$	176.2	$1,000^*$	$0.17e-07$	$0.17e-07$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-7$	1.0	$0.32e-02$	$1,766.0$	100^*	$0.63e-08$	$0.42e-08$	1
$\gamma = 0.10e-6$	1.0	$0.15e-10$	164.6	2	$0.24e-17$	$0.12e-17$	0
$\gamma = 0.10e-5$	1.0	$0.68e-10$	48.2	20	$0.13e-16$	$0.67e-17$	0
equal	1.0	$0.69e-10$	2.0	5	$0.32e-14$	$0.16e-14$	0
other	1.0	$0.24e-03$	207.6	$1,000^*$	$0.18e-07$	$0.13e-07$	0
$\gamma' = 0.10e-1$	1.8	$0.87e-10$	58.6	271	$0.18e-13$	$0.93e-14$	0
$\gamma' = 0.10e+0$	1.1	$0.49e-10$	1.9	2	$0.13e-15$	$0.64e-16$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-7$	1.0	$0.32e-02$	$1,792.0$	100^*	$0.63e-08$	$0.42e-08$	1
$\gamma = 0.10e-6$	1.0	$0.14e-12$	169.7	2	$0.32e-17$	$0.16e-17$	0
$\gamma = 0.10e-5$	1.0	$0.27e-12$	34.3	4	$0.34e-17$	$0.17e-17$	0
equal	1.2	$0.29e-10$	35.5	108	$0.47e-15$	$0.24e-15$	0
other	1.0	$0.70e-10$	204.5	739	$0.34e-12$	$0.17e-12$	0
$\gamma' = 0.10e-1$	1.2	$0.81e-10$	15.4	11	$0.65e-15$	$0.33e-15$	0
$\gamma' = 0.10e+0$	1.0	$0.79e-11$	4.9	2	$0.74e-17$	$0.38e-17$	0

TABLE C.8 continued

Numerical Results for ncd_alt2 ($n = 23,426$, $nz = 156,026$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	156,026	0.5	0.2
ILUTH(10^{-3})	154,747	17.7	0.7
ILUTH(10^{-5})	241,259	18.3	1.2
ILUK(10)	234,073	45.3	18.6

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	269.4	500*	$0.18e-06$	$0.65e-08$	$0.47e-07$
	ILUTH(10^{-3})	267.3	500*	$0.48e-06$	$0.10e-06$	$0.13e-06$
	ILUTH(10^{-5})	24.0	40	$0.58e-12$	$0.32e-13$	$0.16e-13$
	ILUK(10)	48.5	83	$0.36e-09$	$0.65e-15$	$0.32e-15$
DQGMRES ($k = 20$)	ILU0	558.2	500*	$0.20e-06$	$0.98e-08$	$0.64e-07$
	ILUTH(10^{-3})	536.7	500*	$0.56e-06$	$0.16e-06$	$0.52e-06$
	ILUTH(10^{-5})	58.7	56	$0.45e-15$	$0.18e-16$	$0.91e-17$
	ILUK(10)	561.4	500*	$0.35e-09$	$0.63e-15$	$0.32e-15$
BCG	ILU0	342.2	500*	$0.12e+05$	$0.15e-04$	$0.45e-03$
	ILUTH(10^{-3})	340.0	500*	$0.36e-03$	$0.52e-06$	$0.10e-05$
	ILUTH(10^{-5})	18.5	23	$0.30e-10$	$0.14e-11$	$0.72e-12$
	ILUK(10)	398.3	500*	$0.36e-01$	$0.35e-10$	$0.18e-10$
CGS	ILU0	335.5	500*	$0.74e+08$	$0.14e-04$	$0.92e-04$
	ILUTH(10^{-3})	335.3	500*	$0.28e+07$	$0.68e-06$	$0.13e-05$
	ILUTH(10^{-5})	14.9	19	$0.96e-10$	$0.43e-11$	$0.22e-11$
	ILUK(10)	212.1	273	$0.30e-10$	$0.13e-17$	$0.65e-18$
BCGStab	ILU0	302.3	443	$0.60e-10^{s}$	$0.41e-12$	$0.20e-12$
	ILUTH(10^{-3})	125.0	183	$0.41e-10^{s}$	$0.12e-11$	$0.58e-12$
	ILUTH(10^{-5})	13.9	18	$0.93e-10^{s}$	$0.53e-11$	$0.27e-11$
	ILUK(10)	24.6	31	$0.17e-09$	$0.30e-16$	$0.15e-16$
QMR	ILU0	422.5	500*	$0.23e-05$	$0.39e-06$	$0.35e-05$
	ILUTH(10^{-3})	418.7	500*	$0.16e-05$	$0.46e-07$	$0.26e-07$
	ILUTH(10^{-5})	21.5	22	$0.50e-10$	$0.19e-11$	$0.96e-12$
	ILUK(10)	477.0	500*	$0.62e-05$	$0.53e-11$	$0.27e-11$

D. A Telecommunication Model. A model of a telecommunication problem is considered to study the effect of impatient telephone customers on a computerized telephone exchange [33]. The model is shown in Figure D.1. In this model each customer makes a request for service. Then the customer has to wait a certain period for a reply. If the reply has not arrived at the end of that period, the customer has the right to either give up and leave the network, or wait for some period of time before trying again.

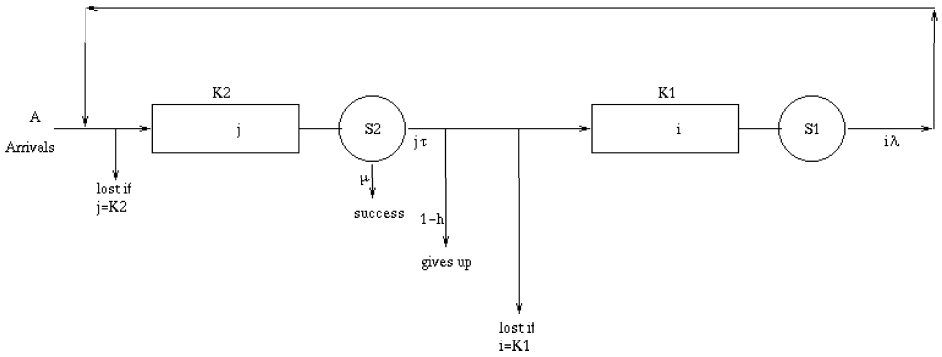

FIGURE D.1 *Telecommunication Model.*

All customers have to pass by station $S2$ which is dedicated to a special processing task. These customers are processed by a single server according to a processor sharing discipline. Each customer may wait in $S2$ for a certain time which is defined as an upper bound on its service duration: whenever its patience is exhausted, the customer simply gives up processing (with a fixed probability $1 - h$).

In case the customer decides to keep trying, it joins an infinite server station $S1$ where it remains for a certain period, called the thinking-time, before joining back station $S2$ for another attempt.

We are interested in studying the number of customers in $S1$ and $S2$ in the long run. Let i and j be the number of customers respectively in $S1$ and $S2$. Then the state of the network may be described by the pair (i, j). When $j \geq 1$, the rate of

- service completions in $S2$ is μ,
- departures due to impatience is $j\tau$.

When $i \geq 1$, the rate of departures from $S1$ is $i\lambda$. External arrivals to $S2$ are assumed to have a Poisson distribution of rate A.

As we are interested in finite Markov chains, we let $K1$ and $K2$ be the maximum sizes of $S1$ and $S2$, respectively. Customers arriving to a full station are lost. It is important to choose large values for $K1$ and $K2$ so that the probability of saturation is negligible. In that case, the truncation of the state space will have little effect, and hence, the resulting steady-state probabilities may be taken as an accurate approximation of those of the infinite capacity network.

The following values are taken from [24] to be used in our experiments:

$$A = 0.6, \quad \mu = 1.0, \quad \tau = 0.05, \quad h = 0.85, \quad \lambda = 5.0.$$

The state space of the Markov chain is of size $(K1 + 1)(K2 + 1)$. We set $K1 = 30$ and

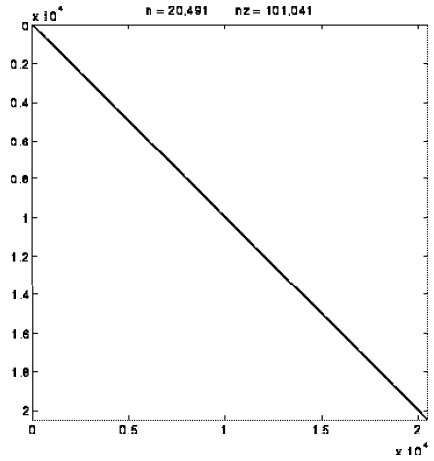

FIGURE D.2 *telecom.*

TABLE D.1
Characteristics of the Telecommunication Problem.

n	nz	symmetric nz structure
20,491	101,041	no

TABLE D.2
Partitioning Results for the telecom Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-1$	1,981	1	31	$0.10e+1$
$0.10e+0$	14,389	1	28	$0.10e+1$

	number of blocks	last block size	degree of coupling
equal	144	42	$0.99e+0$
other	202	190	$0.99e+0$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.55e-2$	3	1	20,488	$0.87e-2$
$0.15e+0$	563	2	3,257	$0.26e+0$
$0.81e+0$	661	2	19,170	$0.82e+0$

$K2 = 660$ which gives the *telecom* matrix of order $n = 20,491$ and number of nonzero elements $nz = 101,041$. Results of the *ncdtest*, *equal*, *other*, and *newncd* partitionings are shown in Table D.2. The smallest degree of coupling of the *telecom* test matrix for the values of γ' used with the *newncd* partitioning algorithm is $0.87e-2$. The time taken by the *ncdtest* and *newncd* partitioning algorithms does not exceed 1 and 1.2 seconds for the *telecom* test matrix, respectively. Tables D.1 and D.3 provide information about the symmetric nonzero structure and bandwidth of the test matrix.

TABLE D.3
Lower, Higher Bandwidths and Coefficient of Asymmetry of the telecom Test Matrix.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
telecom	31	60	(2.27,1.14)

The *telecom* matrix is a test case in which some of the Krylov subspace solvers turn out to be the fastest solvers. CGS with ILUTH(10^{-5}) is the winner for *telecom*. BCGStab and GMRES with ILUTH(10^{-5}) follow as close second and third, respectively (see Table D.4). The ILU0 preconditioner and, to a certain extent, the ILUK(10) preconditioner are very ineffective for this test problem. The *ncdtest* algorithm does not give balanced partitionings for the *telecom* test matrix (see Table D.2), and two-level solvers do not perform well with *ncdtest* when $\gamma = 0.10e-1$ and $\gamma = 0.10e+0$ are used. The performance of IAD with the *equal* and *other* partitionings is clearly superior to that of BSOR. Nevertheless, BSOR with the *newncd* $\gamma' = 0.15e+0$ partitioning is the fastest two-level solver. The performance of SOR for this test matrix is very poor.

TABLE D.4
Numerical Results for telecom ($n = 20,491$, $nz = 101,041$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.63e-04$	122.9	1,000*	$0.11e-05$	$0.12e-05$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-1$	1.6	$0.86e-10$	55.6	303	$0.38e-12$	$0.36e-12$	0
$\gamma = 0.10e+0$	1.2	$0.49e-08$	253.2	956	$0.18e-08$	$0.17e-08$	0
equal	1.4	$0.17e-08$	99.0	185	$0.15e-10$	$0.14e-10$	0
other	1.6	$0.64e-10$	221.4	465	$0.10e-12$	$0.95e-13$	0
$\gamma' = 0.55e-2$	1.0	$0.40e-10$	21.2	7	$0.21e-13$	$0.20e-13$	0
$\gamma' = 0.15e+0$	1.9	$0.57e-10$	5.7	13	$0.99e-13$	$0.93e-13$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-1$	0.9	$0.84e-10$	55.2	32	$0.40e-13$	$0.37e-13$	0
$\gamma = 0.10e+0$	1.0	$0.56e-16$	116.2	2	$0.69e-17$	$0.65e-17$	0
equal	1.0	$0.99e-10$	10.1	8	$0.16e-12$	$0.15e-12$	0
other	1.0	$0.73e-10$	18.9	28	$0.46e-13$	$0.43e-13$	0
$\gamma' = 0.55e-2$	1.0	$0.28e-10$	21.4	7	$0.15e-13$	$0.14e-13$	0
$\gamma' = 0.15e+0$	1.7	$0.32e-10$	8.1	12	$0.13e-12$	$0.12e-12$	0

TABLE D.4 continued
Numerical Results for telecom ($n = 20,491$, $nz = 101,041$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	101,041	0.3	0.1
ILUTH(10^{-3})	181,126	1.5	0.7
ILUTH(10^{-5})	318,749	2.3	1.8
ILUK(10)	204,807	5.3	3.2

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	197.1	500*	$0.25e-05$	$0.57e-05$	$0.89e-04$
	ILUTH(10^{-3})	224.2	500*	$0.13e-05$	$0.12e-05$	$0.26e-05$
	ILUTH(10^{-5})	3.3	7	$0.97e-15$	$0.72e-16$	$0.68e-16$
	ILUK(10)	231.5	500*	$0.43e-06$	$0.88e-06$	$0.25e-05$
DQMRES ($k = 20$)	ILU0	455.5	500*	$0.18e-05$	$0.23e-05$	$0.32e-05$
	ILUTH(10^{-3})	294.5	312	$0.12e-13$	$0.89e-14$	$0.84e-14$
	ILUTH(10^{-5})	4.6	7	$0.97e-15$	$0.72e-16$	$0.68e-16$
	ILUK(10)	485.2	500*	$0.30e-06$	$0.22e-06$	$0.28e-06$
BCG	ILU0	240.1	500*	$0.79e+04$	$0.17e-03$	$0.13e-01$
	ILUTH(10^{-3})	307.3	500*	$0.36e+01$	$0.11e-03$	$0.15e-03$
	ILUTH(10^{-5})	11.3	14	$0.86e-10$	$0.13e-10$	$0.12e-10$
	ILUK(10)	328.9	500*	$0.19e+00$	$0.12e-03$	$0.28e-02$
CGS	ILU0	236.0	500*	$0.40e+11$	$0.34e-03$	$0.10e-01$
	ILUTH(10^{-3})	26.9	44	$0.97e-10$	$0.16e-09$	$0.16e-09$
	ILUTH(10^{-5})	2.5	3	$0.22e-10$	$0.16e-11$	$0.15e-11$
	ILUK(10)	321.3	500*	$0.13e+16$	$0.26e-03$	$0.18e-01$
BCGStab	ILU0	241.6	500*	$0.32e-02$	$0.20e-04$	$0.79e-03$
	ILUTH(10^{-3})	44.4	72	$0.26e-10^{s}$	$0.10e-10$	$0.99e-11$
	ILUTH(10^{-5})	2.5	3	$0.26e-11$	$0.20e-11$	$0.19e-11$
	ILUK(10)	328.8	500*	$0.14e-06$	$0.76e-07$	$0.93e-07$
QMR	ILU0	299.4	500*	$0.39e-05$	$0.55e-05$	$0.27e-03$
	ILUTH(10^{-3})	39.8	54	$0.77e-07$	$0.31e-07$	$0.29e-07$
	ILUTH(10^{-5})	7.0	7	$0.11e-09$	$0.84e-11$	$0.79e-11$
	ILUK(10)	97.4	125	$0.15e-07$	$0.71e-09$	$0.67e-09$

E. A Queueing Network with Blocking and Priority Service Model. The model we shall discuss now is an open queueing network of three finite capacity queues and two customer classes. Class 1 customers arrive from the exterior to queue 1 according to a Poisson process with parameter λ_1. Similarly, class 2 customers arrive from outside the network to queue 2 according to a Poisson process, but this time at rate λ_2. A customer (from either class) will be lost if upon arrival it finds the corresponding queue full. The servers at queues 1 and 2 provide exponential service respectively at rates μ_1 and μ_2. After being served, customers in both of these queues try to join queue 3. If queue 3 is full, class 1 customers are blocked (after service) and the server at queue 1 must halt. This server cannot resume serving the next customer unless a slot becomes available in the buffer of queue 3 and the blocked customer is transferred. On the other hand, when a class 2 customer has been served at queue 2 and finds the buffer at queue 3 full, it is simply lost. Queue 3 provides exponential service at rate μ_{3_1} to class 1 customers and at rate μ_{3_2} to class 2 customers. Customers departing after service from queue 3 leave the network. Figure E.1 illustrates this model. $C_k - 1$, $k = 1, 2, 3$ denote the finite buffer capacity at queue k.

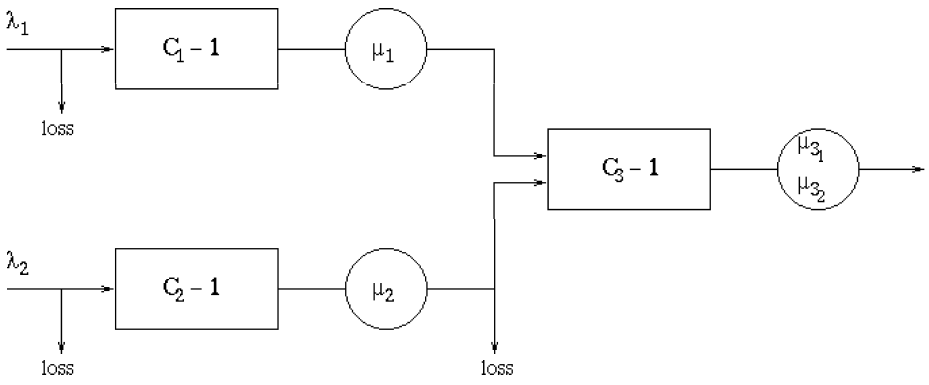

FIGURE E.1 *A Queueing Network Model.*

The states of the Markov chain underlying this model may be represented by four-component vectors [33]. Components 1 and 2 may be used to denote respectively the number of customers in queues 1 and 2. Components 3 and 4 may be used to represent respectively the number of class 1 and class 2 customers present in queue 3.

We assign the following values to the parameters in Figure E.1:

$$\lambda_1 = 1.0, \quad \lambda_2 = 2.0, \quad \mu_1 = 3.0, \quad \mu_2 = 4.0, \quad \mu_{3_1} = 5.0, \quad \mu_{3_2} = 6.0.$$

The state space is of size $C_1C_2C_3(C_3 + 1)/2$; hence setting each of C_1 and C_2 to 15 and C_3 to 30 gives the qn matrix of order $n = 104,625$ and number of nonzero elements $nz = 593,115$. The time to partition the qn test matrix using *ncdtest* and *newncd* does not take more than 5.4 and 6.8 seconds, respectively. The smallest degree of coupling we came across when using the *newncd* partitioning algorithm is $0.25e + 0$. Additional information about the test matrix is given in Tables E.1, E.2, and E.3.

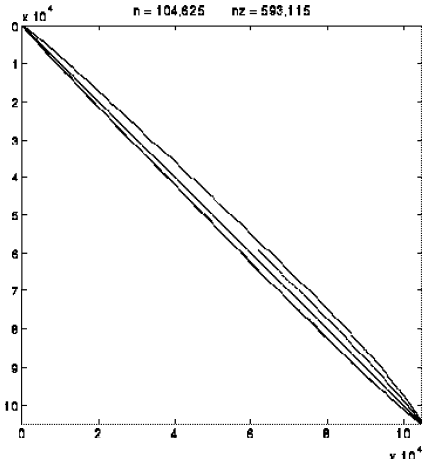

FIGURE E.2 *qn.*

TABLE E.1
Characteristics of the Queueing Network Problem.

n	nz	symmetric nz structure
104,625	593,115	no

TABLE E.2
Partitioning Results for the qn Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e+0$	91,800	1	450	$0.94e+0$

	number of blocks	last block size	degree of coupling
equal	324	296	$0.10e+1$
other	457	429	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.19e+0$	15	6,975	6,975	$0.25e+0$
$0.30e+0$	225	465	465	$0.62e+0$
$0.37e+0$	226	30	97,875	$0.62e+0$

The *qn* test case is the largest problem in this study. The number of blocks in the partition obtained using *ncdtest* $\gamma = 10^{-1}$, and hence the order of the coupling matrix for IAD, is very large (i.e., 91,800, see Table E.2). This causes the solution time of IAD to be unreasonably long for the *qn* test matrix when *ncdtest* is used (see Table E.4). Nevertheless, we still can find at least one partitioning for BSOR having satisfactory convergence time. For the *qn* test matrix, BSOR with *ncdtest* $\gamma = 0.10e+0$ emerges as the fastest solver, SOR a distant second, and IAD with *equal* a distant third. Two-level

TABLE E.3
Lower, Higher Bandwidths and Coefficients of Asymmetry of the qn Test Matrices.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
qn	2,728	5,385	(2.06,1.06)

solvers with the *newncd* partitionings do not perform well on this (rather) non-NCD problem. The performance of Krylov subspace solvers except BCG are satisfactory in terms of the number of iterations taken to convergence. However, the order of the test matrix and its wide bandwidth (see Table E.3) cause the computation of the ILUTH and ILUK preconditioners to be very expensive and the total solution time unduly long. We would like to remark that we use 10^{-2} and 10^{-3} as threshold values for the ILUTH preconditioner, and do not experiment with 10^{-5}. On the other hand, CGS, BCGStab, and QMR with ILU0 preconditioning emerge as the fastest Krylov subspace solvers.

TABLE E.4
Numerical Results for qn ($n = 104,625$, $nz = 593,115$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.2	$0.85e-10$	93.4	134	$0.24e-10$	$0.73e-10$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e+0$	1.1	$0.66e-10$	76.6	44	$0.32e-11$	$0.98e-11$	91,350
equal	1.2	$0.90e-10$	111.6	113	$0.22e-10$	$0.66e-10$	0
other	1.2	$0.86e-10$	130.1	136	$0.24e-10$	$0.73e-10$	2
$\gamma' = 0.19e+0$	1.4	$0.20e-09^{\dagger}$	205.2	46	$0.74e-10$	$0.23e-09$	7
$\gamma' = 0.30e+0$	1.3	$0.10e-09$	126.7	136	$0.86e-11$	$0.26e-10$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e+0$	1.0	requires unreasonably long time (large coupling matrix)					
equal	1.2	$0.78e-10$	104.0	55	$0.11e-10$	$0.34e-10$	0
other	1.2	$0.72e-10$	117.1	65	$0.24e-11$	$0.72e-11$	2
$\gamma' = 0.19e+0$	1.2	$0.96e-10$	456.2	58	$0.48e-11$	$0.15e-10$	8
$\gamma' = 0.30e+0$	1.3	$0.88e-10$	246.9	110	$0.77e-11$	$0.23e-10$	0

TABLE E.4 continued
Numerical Results for qn ($n = 104,625$, $nz = 593,115$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	593,115	2.0	0.5
ILUTH(10^{-2})	1,020,335	798.4	6.6
ILUTH(10^{-3})	1,073,171	1,064.9	11.0
ILUK(10)	1,046,092	952.4	12.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	1,117.0	500*	$0.27e-04$	$0.18e-03$	$0.14e-02$
	ILUTH(10^{-2})	204.3	80	$0.30e-10$	$0.20e-10$	$0.62e-10$
	ILUTH(10^{-3})	207.0	80	$0.47e-10$	$0.32e-10$	$0.95e-10$
	ILUK(10)	130.0	40	$0.39e-10$	$0.19e-10$	$0.56e-10$
(9) DQGMRES(7) (7) (7)	ILU0	477.3	146	$0.75e-11$	$0.50e-11$	$0.15e-10$
	ILUTH(10^{-2})	384.6	119	$0.29e-11$	$0.20e-11$	$0.60e-11$
	ILUTH(10^{-3})	347.3	106	$0.24e-11$	$0.16e-11$	$0.48e-11$
	ILUK(10)	314.5	96	$0.15e-11$	$0.71e-12$	$0.21e-11$
BCG	ILU0	1,395.0	500*	$0.27e-01$	$0.43e-02$	$0.11e+00$
	ILUTH(10^{-2})	1,759.0	500*	$0.20e+01$	$0.22e-02$	$0.25e+00$
	ILUTH(10^{-3})	1,787.0	500*	$0.73e-06$	$0.64e-06$	$0.19e-05$
	ILUK(10)	1,768.0	500*	$0.20e-01$	$0.10e-02$	$0.16e+00$
CGS	ILU0	105.2	38	$0.33e-10$	$0.22e-10$	$0.66e-10$
	ILUTH(10^{-2})	87.7	25	$0.21e-10$	$0.14e-10$	$0.43e-10$
	ILUTH(10^{-3})	86.1	24	$0.64e-10$	$0.45e-10$	$0.14e-09$
	ILUK(10)	81.6	23	$0.54e-11$	$0.26e-11$	$0.79e-11$
BCGStab	ILU0	125.3	45	$0.78e-10^{s}$	$0.52e-10$	$0.16e-09$
	ILUTH(10^{-2})	87.1	25	$0.17e-10^{s}$	$0.11e-10$	$0.34e-10$
	ILUTH(10^{-3})	87.3	24	$0.80e-10$	$0.54e-10$	$0.16e-09$
	ILUK(10)	86.4	24	$0.93e-11$	$0.45e-11$	$0.14e-10$
QMR	ILU0	177.4	52	$0.32e-08$	$0.60e-08$	$0.18e-07$
	ILUTH(10^{-2})	173.7	42	$0.38e-08$	$0.36e-08$	$0.11e-07$
	ILUTH(10^{-3})	185.6	44	$0.83e-09$	$0.55e-09$	$0.17e-08$
	ILUK(10)	138.3	33	$0.49e-09$	$0.91e-09$	$0.28e-08$

F. A Multiplexing Model of a Leaky Bucket in Tandem. One of the major problems in ATM networks is to control the congestion of intermediate buffers with fast and simple mechanisms. Several policies have been proposed and evaluated with diverse probabilistic hypothesis. The simplest mechanism is the leaky bucket. The problem is to determine the behavior of this mechanism under external arrivals [33]. An evaluation of this mechanism will enable its comparison with other, more complex mechanisms.

The traffic source is of an $M/D/1/C$ type. The external arrival stream is modeled as a Poisson process with rate λ. The queue is of size C cells and has a single server with service time D, which will be taken as unit time. Hence, the model may be viewed as a one-dimensional discrete-time Markov chain with time slots of length D and state descriptor N_p denoting the number of cells produced by the Poisson source at time t,

The leaky bucket has a finite size of K cells and a service time given by $T'D = TD(1-\epsilon)$, where T' is an integer. The state of the system is described by the state variable k, which is the buffer occupancy (in terms of the number of cells).

The values used for the above parameters are

$$C = K = 64, \quad T = 4, \quad \lambda = 0.85, \quad \epsilon = 0.4959.$$

The *leaky* matrix we generated from this model has order $n = C \times K \times T(1-\epsilon) = 8,258$ and number of nonzero elements $nz = 197,474$. This matrix is severely ill-conditioned (with a degree of coupling $0.19e-101$), although the degree of coupling values computed for none of the *ncdtest* partitionings reflects this fact. Information about the nonzero structure of the *leaky* test matrix and its various partitionings are provided in Tables F.1,

TABLE F.1
Characteristics of the Leaky-Bucket Problem.

n	nz	symmetric nz structure
8,258	197,474	no

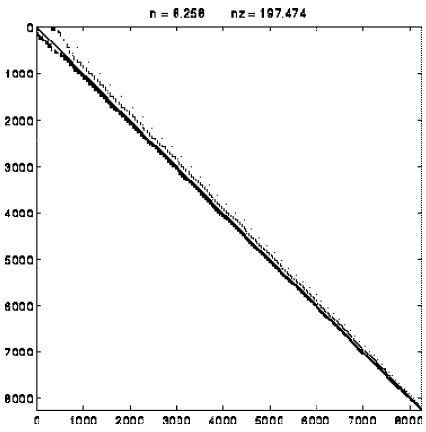

FIGURE F.1 *leaky.*

TABLE F.2
Partitioning Results for the leaky Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-102$	5	1	8,254	$0.14e-15$
$0.10e-008$	193	1	8,057	$0.86e+00$
$0.10e-007$	225	1	8,016	$0.86e+00$
$0.10e-006$	225	1	8,016	$0.86e+00$
$0.10e-005$	265	1	7,976	$0.86e+00$
$0.10e-004$	318	1	7,923	$0.86e+00$
$0.10e-003$	400	1	7,832	$0.86e+00$
$0.10e-002$	531	1	7,692	$0.86e+00$
$0.10e-001$	778	1	7,427	$0.86e+00$
$0.10e+000$	7,386	1	507	$0.86e+00$

	number of blocks	last block size	degree of coupling
equal	91	158	0.10e+1
other	129	2	0.10e+1

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.19e-101$	2	1	8,257	$0.19e-101$
$0.10e-014$	2	4	8,254	$0.14e-015$
$0.64e+000$	64	4	8,005	$0.50e+000$

TABLE F.3
Lower, Higher Bandwidths and Coefficient of Asymmetry of the leaky Test Matrix.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
leaky	191	435	(2.46,1.46)

F.2, and F.3. The time to partition the *leaky* test matrix using the *ncdtest* and *newncd* algorithms does not exceed 2.8 and 8.5 seconds, respectively.

The *leaky* test matrix is the most interesting case in our test suite. All ILU factorization attempts of this test matrix fail which prevents us from experimenting with Krylov subspace solvers using ILU preconditioners. When we do not use any preconditioner, none of the Krylov subspace solvers except BCGStab, which terminates at iteration 359 with a residual of $0.75e-10^{s}$ after 102.8 seconds, converge in 500 iterations. In the *ncdtest* $\gamma = 0.10e-8$ partitioning, the matrix is partitioned to 193 diagonal blocks; the largest block is of order 8,057 and there are 192 blocks of very small order (see Table F.2). Choosing $\gamma = 0.10e+0$ leads to a partitioning of 7,386 diagonal blocks; the largest block is of order 507 and the rest of the blocks are very small. As a consequence of these two unbalanced partitionings, BSOR and IAD cannot benefit from the divide-and-conquer nature of two-level solvers. Hence, the time taken for solving the diagonal blocks is bi-

ased towards solving the largest block in both partitionings. Going back to aggregation in IAD, these two unbalanced partitionings make aggregation a detrimental step rather than an accelerator for convergence causing IAD with *ncdtest* $\gamma = 0.10e+0$ to take an unreasonably long time. The *equal* and *other* partitionings, as always, provide more balanced partitionings. However, the winner is BSOR/IAD with the *newncd* $\gamma' = 0.10e-14$ partitioning that has a total solution time of 16.8 seconds (of which 4.4 seconds is for partitioning). IAD with *equal* and IAD with *other* are the next fastest solvers. Finally, we should note that SOR has below average performance among other solvers for this test matrix.

TABLE F.4
Numerical Results for leaky ($n = 8,258$, $nz = 197,474$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.97e-10$	57.8	409	$0.75e-10$	$0.38e-09$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-008$	1.0	$0.28e-16$	39.7	3	$0.98e-16$	$0.50e-15$	0
$\gamma = 0.10e+000$	1.0	$0.94e-10$	33.3	142	$0.11e-10$	$0.55e-10$	0
equal	1.1	$0.96e-10$	59.1	255	$0.83e-10$	$0.42e-09$	89
other	1.0	$0.99e-10$	73.8	313	$0.77e-10$	$0.39e-09$	106
$\gamma' = 0.19e-101$	1.0	$0.42e-16$	22.5	2	$0.46e-16$	$0.23e-15$	0
$\gamma' = 0.10e-014$	1.0	$0.14e-16$	12.4	2	$0.14e-16$	$0.69e-16$	0
$\gamma' = 0.64e+000$	1.0	$0.14e-13$	19.0	2	$0.48e-14$	$0.25e-13$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-008$	1.1	$0.40e-10$	37.6	2	$0.40e-10$	$0.20e-09$	0
$\gamma = 0.10e+000$	1.0	requires unreasonably long time (large coupling matrix)					
equal	1.0	$0.92e-10$	19.3	72	$0.14e-11$	$0.74e-11$	89
other	1.0	$0.86e-10$	23.4	82	$0.12e-11$	$0.61e-10$	106
$\gamma' = 0.19e-101$	1.0	$0.28e-16$	21.1	2	$0.46e-16$	$0.23e-15$	0
$\gamma' = 0.10e-014$	1.0	$0.42e-16$	12.4	2	$0.14e-16$	$0.69e-16$	0
$\gamma' = 0.64e+000$	1.0	$0.14e-13$	19.4	2	$0.49e-14$	$0.25e-13$	0

G. Mutex—A Resource Sharing Model. In this model, M distinguishable processes share a certain resource [33]. Each of these processes alternates between a sleeping state and a resource using state. However, only P processes may concurrently use the resource, where $1 \leq P \leq M$. If a process currently in the sleeping state tries to move to the resource using state while there are P processes already using the resource, it simply fails to access the resource and remains in the sleeping state. Notice that when $P = 1$ this model reduces to the usual mutual exclusion problem, whereas when $P = N$ all the processes are independent. Let λ_i be the rate at which process i awakes from the sleeping state wishing to access the resource and let μ_i be the rate at which this same process releases the resource when it has a possession of it. Figure G.1 provides a graphical illustration of this model. Each process i is modeled by a two-state automaton A_i. The function f takes the value 1 when access is permitted to the resource and takes the value 0 otherwise.

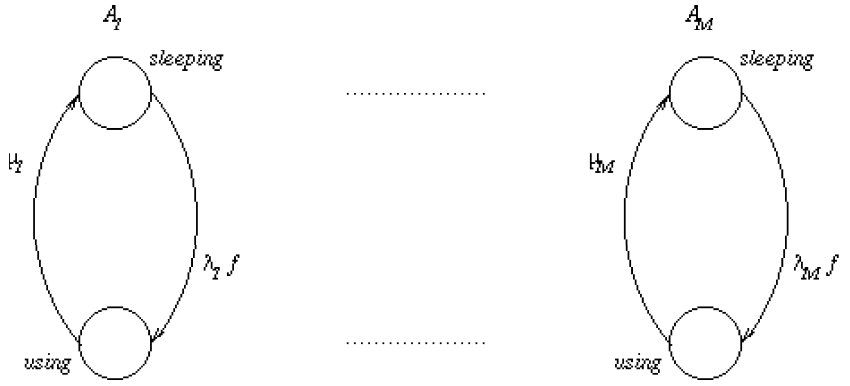

FIGURE G.1 *A Resource Sharing Model (Mutex).*

We set $\lambda_i = 1/i$ and $\mu_i = i$, for $i = 1, 2, \ldots, M$. Parameters P and M are fixed to 8 and 16, respectively. These values yield the *mutex* matrix of order $n = \sum_{p=0}^{P} \binom{M}{p} = 39,203$ and number of nonzero elements $nz = 563,491$.

Two more test cases are generated from this model. The *mutex_alt*1 test matrix is obtained by setting $\mu_i = 10^3\, i$, whereas the *mutex_alt*2 test matrix is generated by setting $\lambda_i = 10^{-3}/i$ and $\mu_i = 10^3\, i$. As the values of P and M are fixed, the three test cases have the same order, number of nonzero elements, and nonzero structure. The partitioning results of these matrices are shown in Tables G.2, G.3, and G.4. The time to partition the *mutex* test matrices using *ncdtest* and *ncdnew* does not exceed 5.5 and 5.8 seconds, respectively. The smallest degree of coupling we came across while experimenting with the *newncd* partitioning is $0.19e-1$ for all three matrices. Tables G.1 and G.5 provide information about the nonzero structure and bandwidth of the three test matrices generated from this model.

The *mutex* test matrix is the densest of this study with an average of about 15 nonzero elements per row (see Figure G.2). Unsurprisingly, SOR outperforms all other

solvers in terms of total solution time for the *mutex* test matrix (see Table G.6). This is the only test matrix where SOR is superior to all the other solvers considered. The Krylov subspace solvers CGS, BCGStab, GMRES, and DQGMRES with ILU0 preconditioning give very competitive results on all the *mutex* test matrices (see Tables G.6 through G.8).

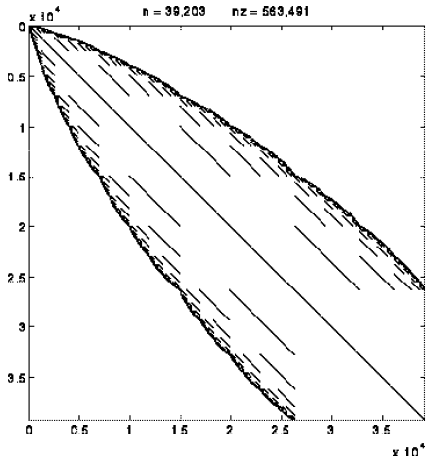

FIGURE G.2 *mutex.*

TABLE G.1
Characteristics of the Mutex Problem.

n	nz	symmetric nz structure
39,203	563,491	yes

TABLE G.2
Partitioning Results for the mutex Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-2$	256	1	256	$0.10e+1$

	number of blocks	last block size	degree of coupling
equal	198	394	$0.10e+1$
other	280	143	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.19e-1$	2	16,384	22,819	$0.19e-1$
$0.75e-1$	128	10	511	$0.32e+0$
$0.10e+0$	503	8	128	$0.48e+0$
$0.15e+0$	16,385	2	6,435	$0.93e+0$

TABLE G.3
Partitioning Results for the mutex_alt1 Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-5$	256	1	256	$0.10e+1$

	number of blocks	last block size	degree of coupling
equal	198	394	$0.10e+1$
other	280	143	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.19e-1$	2	16,384	22,819	$0.19e-1$
$0.75e-1$	128	10	511	$0.32e+0$
$0.10e+0$	503	8	128	$0.48e+0$
$0.15e+0$	16,385	2	6,435	$0.93e+0$

TABLE G.4
Partitioning Results for the mutex_alt2 Test Matrix.

γ *ncdtest*	number of blocks	smallest block size	largest block size	degree of coupling
$0.10e-8$	256	1	256	$0.10e+1$

	number of blocks	last block size	degree of coupling
equal	198	394	$0.10e+1$
other	280	143	$0.10e+1$

γ' *newncd*	number of blocks	smallest block size	largest block size	degree of coupling
$0.19e-1$	2	16,384	22,819	$0.19e-1$
$0.75e-1$	128	10	511	$0.32e+0$
$0.10e+0$	503	8	128	$0.48e+0$
$0.15e+0$	16,385	2	6,435	$0.93e+0$

However, it is BSOR with the *equal* and *other* partitionings that follows SOR respectively as close second and third for the *mutex* test matrix. The wide bandwidth and the order of each test matrix cause the ILUTH and ILUK preconditioning times to be extremely long. The solvers obtained therefrom do not perform well, and there is no need to experiment with a threshold value of 10^{-5}.

The test matrices *mutex_alt*1 and *mutex_alt*2 are not more ill-conditioned than the *mutex* test matrix. See the degree of coupling values corresponding to γ' in Tables G.2, G.3, and G.4. For the *mutex_alt*1 test matrix, BSOR with *equal* is the fastest solver followed by SOR as close second and BSOR with *other* as close third. All Krylov subspace solvers except BCG perform better than they do for the *mutex* matrix (compare the

TABLE G.5
Lower, Higher Bandwidths and Coefficients of Asymmetry of the mutex Test Matrices.

matrix	lower bandwidth	higher bandwidth	coefficient of asymmetry
mutex	13,495	13,495	(1.64,0.69)
*mutex_alt*1	13,495	13,495	(1.61,0.70)
*mutex_alt*2	13,495	13,495	(1.61,0.70)

results in Tables G.6 and G.7). Results of the *mutex_alt*2 test matrix also demonstrate similar improvement, except for the ILU0 and ILUK(10) preconditioners, in a few of the solvers (see Table G.8). The ILUTH preconditioners corresponding to threshold values 10^{-2} and 10^{-3} are exactly the same for the *mutex_alt*1 and *mutex_alt*2 test matrices. That is why we report the results corresponding to a single threshold value for these two matrices. The winner for *mutex_alt*2 is BSOR with *equal*. SOR comes a close second and BSOR with *other* a third just as in *mutex_alt*1. We should also remark that BCG with ILU0 and ILUK(10) converge for the *mutex_alt*2 test matrix when it does not within 500 iterations for the *mutex* and *mutex_alt*1 test matrices. As for SOR, BSOR, and IAD, they all take less time to converge as we move from *mutex* to *mutex_alt*1 and then to *mutex_alt*2. The performance of two-level iterative solvers with the *ncdtest* and *newncd* partitionings for the *mutex* test matrices is below average.

TABLE G.6
Numerical Results for mutex ($n = 39,203$, $nz = 563,491$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.1	$0.40e-10$	9.7	19	$0.34e-12$	$0.41e-12$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-2$	1.1	$0.16e-10$	123.6	12	$0.74e-13$	$0.90e-13$	3
equal	1.1	$0.54e-10$	11.6	15	$0.25e-13$	$0.30e-13$	0
other	1.1	$0.36e-10$	11.6	18	$0.40e-12$	$0.48e-12$	0
$\gamma' = 0.75e-1$	1.1	$0.33e-10$	227.3	14	$0.58e-12$	$0.71e-12$	63
$\gamma' = 0.10e+0$	1.1	$0.16e-10$	53.0	15	$0.44e-12$	$0.53e-12$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-2$	1.1	$0.12e-10$	193.4	12	$0.87e-13$	$0.11e-12$	10
equal	1.0	$0.18e-10$	115.2	12	$0.51e-13$	$0.62e-13$	0
other	1.0	$0.71e-10$	96.8	14	$0.72e-12$	$0.88e-12$	0
$\gamma' = 0.75e-1$	1.0	$0.42e-10$	288.0	12	$0.64e-12$	$0.78e-12$	66
$\gamma' = 0.10e+0$	1.0	$0.17e-10$	135.3	12	$0.21e-12$	$0.25e-12$	0

TABLE G.6 continued
Numerical Results for mutex ($n = 39,203$, $nz = 563,491$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	563,491	2.5	0.6
ILUTH(10^{-2})	301,347	814.8	0.9
ILUTH(10^{-3})	496,693	876.9	3.0
ILUK(10)	392,037	1,215.4	15.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	11.8	10	$0.27e-11$	$0.24e-11$	$0.30e-11$
	ILUTH(10^{-2})	9.6	10	$0.35e-11$	$0.77e-11$	$0.93e-11$
	ILUTH(10^{-3})	13.7	12	$0.27e-11$	$0.27e-11$	$0.33e-11$
	ILUK(10)	15.9	15	$0.54e-11$	$0.48e-11$	$0.58e-11$
DQMRES ($k = 20$)	ILU0	15.3	10	$0.27e-11$	$0.24e-11$	$0.30e-11$
	ILUTH(10^{-2})	13.3	10	$0.35e-11$	$0.77e-11$	$0.93e-11$
	ILUTH(10^{-3})	18.1	12	$0.27e-11$	$0.27e-11$	$0.33e-11$
	ILUK(10)	22.1	15	$0.54e-11$	$0.48e-11$	$0.58e-11$
BCG	ILU0	1,030.0	500*	$0.95e-02$	$0.49e-02$	$0.11e-01$
	ILUTH(10^{-2})	775.1	500*	$0.32e-01$	$0.61e-02$	$0.85e-01$
	ILUTH(10^{-3})	972.9	500*	$0.82e-05$	$0.82e-05$	$0.99e-05$
	ILUK(10)	859.0	500*	$0.28e+00$	$0.13e-01$	$0.43e-01$
CGS	ILU0	9.8	5	$0.58e-10$	$0.52e-10$	$0.63e-10$
	ILUTH(10^{-2})	7.9	5	$0.15e-10$	$0.32e-10$	$0.39e-10$
	ILUTH(10^{-3})	11.4	6	$0.93e-10$	$0.93e-10$	$0.11e-09$
	ILUK(10)	13.4	8	$0.50e-10$	$0.44e-10$	$0.54e-10$
BCGStab	ILU0	9.9	5	$0.37e-10$	$0.33e-10$	$0.40e-10$
	ILUTH(10^{-2})	7.2	5	$0.92e-10^{s}$	$0.20e-09$	$0.25e-09$
	ILUTH(10^{-3})	11.4	6	$0.34e-10$	$0.33e-10$	$0.41e-10$
	ILUK(10)	12.7	8	$0.90e-10^{s}$	$0.79e-10$	$0.96e-10$
QMR	ILU0	26.4	12	$0.16e-08$	$0.14e-08$	$0.17e-08$
	ILUTH(10^{-2})	19.6	11	$0.46e-09$	$0.10e-08$	$0.12e-08$
	ILUTH(10^{-3})	28.2	13	$0.63e-09$	$0.63e-09$	$0.76e-09$
	ILUK(10)	38.8	20	$0.10e-08$	$0.92e-09$	$0.11e-08$

TABLE G.7
Numerical Results for mutex_alt1 ($n = 39,203$, $nz = 563,491$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.82e-10$	5.2	10	$0.12e-14$	$0.82e-15$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-5$	1.0	$0.27e-11$	114.8	8	$0.94e-16$	$0.67e-16$	3
equal	1.0	$0.27e-10$	4.5	3	$0.76e-13$	$0.54e-13$	0
other	1.0	$0.82e-10$	6.8	10	$0.12e-14$	$0.82e-15$	0
$\gamma' = 0.75e-1$	1.0	$0.25e-11$	213.6	9	$0.46e-15$	$0.33e-15$	63
$\gamma' = 0.10e+0$	1.0	$0.58e-13$	45.7	10	$0.35e-18$	$0.25e-18$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-5$	1.0	$0.15e-10$	178.0	8	$0.63e-14$	$0.45e-14$	10
equal	1.0	$0.36e-11$	90.9	2	$0.41e-14$	$0.29e-14$	0
other	1.0	$0.83e-11$	84.2	4	$0.11e-15$	$0.76e-16$	0
$\gamma' = 0.75e-1$	1.0	$0.48e-10$	281.8	8	$0.18e-14$	$0.13e-14$	66
$\gamma' = 0.10e+0$	1.0	$0.61e-11$	87.3	4	$0.79e-16$	$0.56e-16$	0

TABLE G.7 continued
Numerical Results for mutex_alt1 ($n = 39,203$, $nz = 563,491$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	563,491	2.6	0.6
ILUTH(10^{-2})	301,347	814.4	0.9
ILUK(10)	392,037	1,163.9	15.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	7.2	6	$0.26e-12$	$0.10e-12$	$0.74e-13$
	ILUTH(10^{-2})	3.3	3	$0.27e-12$	$0.64e-12$	$0.45e-12$
	ILUK(10)	11.6	11	$0.21e-13^{\dagger}$	$0.20e-16$	$0.14e-16$
DQMRES ($k = 20$)	ILU0	9.8	6	$0.26e-12$	$0.10e-12$	$0.74e-13$
	ILUTH(10^{-2})	5.0	3	$0.27e-12$	$0.64e-12$	$0.45e-12$
	ILUK(10)	24.7	16	$0.88e-11$	$0.84e-14$	$0.60e-14$
BCG	ILU0	1,028.0	500*	$0.28e-02$	$0.14e-02$	$0.11e-02$
	ILUTH(10^{-2})	779.1	500*	$0.30e-07$	$0.87e-07$	$0.62e-07$
	ILUK(10)	858.6	500*	$0.19e+00$	$0.21e-04$	$0.15e-04$
CGS	ILU0	6.3	3	$0.40e-11$	$0.16e-11$	$0.11e-11$
	ILUTH(10^{-2})	3.4	2	$0.79e-16$	$0.19e-15$	$0.13e-15$
	ILUK(10)	10.5	6	$0.13e-11$	$0.13e-14$	$0.90e-15$
BCGStab	ILU0	5.4	3	$0.78e-10^{s}$	$0.31e-10$	$0.22e-10$
	ILUTH(10^{-2})	2.7	2	$0.71e-13^{s}$	$0.17e-12$	$0.12e-12$
	ILUK(10)	9.5	6	$0.51e-11^{s}$	$0.49e-14$	$0.35e-14$
QMR	ILU0	15.6	7	$0.25e-09$	$0.99e-10$	$0.70e-10$
	ILUTH(10^{-2})	5.5	3	$0.55e-09$	$0.13e-08$	$0.93e-09$
	ILUK(10)	21.6	11	$0.24e-07$	$0.23e-10$	$0.16e-10$

TABLE G.9
Numerical Results for mutex_alt2 ($n = 39,203$, $nz = 563,491$).

SOR

ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error
1.0	$0.15e-12$	4.7	9	$0.50e-16$	$0.35e-16$

BSOR

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-8$	1.0	$0.51e-13$	101.0	3	$0.81e-17$	$0.58e-17$	0
equal	1.0	$0.22e-15$	3.9	2	$0.17e-17$	$0.12e-17$	0
other	1.0	$0.15e-12$	6.2	9	$0.50e-16$	$0.35e-16$	0
$\gamma' = 0.75e-1$	1.0	$0.65e-11$	191.2	8	$0.49e-16$	$0.35e-16$	58
$\gamma' = 0.10e+0$	1.1	$0.53e-10$	40.8	8	$0.14e-11$	$0.10e-11$	0

IAD

Partition.	ω	$\|\|\Delta x\|\|$	Time	#it	$\|\|A\hat{x}\|\|$	Bk.Error	Blocks
$\gamma = 0.10e-8$	1.0	$0.80e-10$	162.1	5	$0.32e-13$	$0.23e-13$	7
equal	1.0	$0.79e-15$	91.9	2	$0.37e-17$	$0.26e-17$	0
other	1.0	$0.13e-13$	82.7	3	$0.50e-16$	$0.35e-16$	0
$\gamma' = 0.75e-1$	1.0	$0.58e-11$	252.5	7	$0.97e-14$	$0.69e-14$	61
$\gamma' = 0.10e+0$	1.0	$0.46e-13$	80.1	3	$0.50e-16$	$0.35e-16$	0

TABLE G.9 continued
Numerical Results for mutex_alt2 ($n = 39,203$, $nz = 563,491$).

Preconditioner	$nzlu$	Time	MFlops
ILU0	563,491	2.6	0.6
ILUTH(10^{-2})	301,347	803.7	0.9
ILUK(10)	392,037	1,158.4	15.8

Method	Preconditioner	Time	#it	$\|\|r\|\|$	$\|\|A\hat{x}\|\|$	Bk.Error
GMRES ($m = 20$)	ILU0	7.1	6	$0.15e-12^{\dagger}$	$0.90e-16$	$0.64e-16$
	ILUTH(10^{-2})	2.5	2	$0.48e-13$	$0.11e-12$	$0.81e-13$
	ILUK(10)	11.7	11	$0.24e-12^{\dagger}$	$0.23e-18$	$0.16e-18$
DQMRES ($k = 20$)	ILU0	17.7	11	$0.32e-12$	$0.28e-15$	$0.20e-15$
	ILUTH(10^{-2})	3.9	2	$0.48e-13$	$0.11e-12$	$0.81e-13$
	ILUK(10)	24.6	16	$0.10e-11$	$0.22e-18$	$0.16e-18$
BCG	ILU0	23.1	11	$0.37e-10$	$0.14e-09$	$0.97e-10$
	ILUTH(10^{-2})	774.3	500*	$0.61e-04$	$0.26e-04$	$0.18e-04$
	ILUK	26.0	15	$0.95e-10$	$0.51e-10$	$0.36e-10$
CGS	ILU0	6.3	3	$0.93e-12$	$0.55e-15$	$0.39e-15$
	ILUTH(10^{-2})	1.9	1	$0.76e-15$	$0.18e-14$	$0.13e-14$
	ILUK(10)	21.6	13	$0.56e-15$	$0.66e-14$	$0.47e-14$
BCGStab	ILU0	5.4	3	$0.40e-11^{s}$	$0.24e-14$	$0.17e-14$
	ILUTH(10^{-2})	1.9	1	$0.17e-15$	$0.40e-15$	$0.28e-15$
	ILUK(10)	15.3	9	$0.63e-11$	$0.37e-15$	$0.26e-15$
QMR	ILU0	13.8	6	$0.10e-08$	$0.61e-12$	$0.43e-12$
	ILUTH(10^{-2})	3.6	2	$0.19e-08$	$0.46e-08$	$0.33e-08$
	ILUK(10)	31.7	16	$0.24e-09$	$0.79e-11$	$0.56e-11$