ITERATIVE METHODS IN LINEAR ALGEBRA
R. Beauwens and P. de Groen (Editors)
Elsevier Science Publishers B.V. (North-Holland)

https://doi.org/10.20378/irbo-52624

SESSION 14: SOFTWARE DEVELOPMENTS

APPLICATION OF NUMERICAL SOLUTION METHODS FOR SINGULAR SYSTEMS IN THE FIELD OF COMPUTATIONAL PROBABILITY THEORY.

UDO R. KRIEGER* AND MICHAEL SCZITTNICK†

Abstract. First we discuss the numerical aspects of a computational approach for the performance evaluation of advanced telecommunication networks by finite continuous-time Markov chains. Then we illustrate by two examples the benefits and difficulties of direct and iterative methods that are applied for the solution of the singular systems arising in this field of computational probability theory.

1. Introduction. Recently, considerable attention has been devoted to the field of computational probability theory, especially to the analysis of Markov chains by numerical methods (cf. [24]). From a probabilistic point of view, the issue of computing the steady-state distribution of an ergodic, homogeneous Markov chain has already been solved since the stationary distribution may be calculated by the solution of a system of linear equations called steady-state balance equations. In practice, however, the application of probability theory in the field of modeling and performance analysis of telecommunication systems raises a lot of difficulties. The derived Markovian models indicate that it is necessary to apply computational solution methods for homogeneous, continuous-time Markov chains (CTMCs) with large but finite state spaces since analytical representations of the steady-state distributions are rarely available.

From a practical point of view, it is desirable that numerical solution techniques for Markov chains are supported by convenient software tools which facilitate the use of the various algorithms. For this reason a software package called MACOM has been developed (cf. [15], [23]). The package can cope with large Markovian models comprising up to 100000 states. It is written in C and running on a SUN workstation under UNIX. In this paper it is our objective to illustrate the benefits and difficulties of the solution methods implemented in MACOM.

The paper is organized as follows: In Section 2 we briefly outline the construction of Markovian models by MACOM. Section 3 is devoted to the mathematical background of numerical solution methods for finite CTMCs. In Section 4 we present the implemented direct methods and in Section 5 the iterative procedures that are based on splittings techniques. In Section 6 the benefits and difficulties of these procedures are illustrated by two examples. Finally, the findings and perspectives of the approach are pointed out in the conclusions.

2. The construction of Markovian models. MACOM is a performance evaluation tool that supports modeling and analysis of small parts of a B-ISDN by Markovian queueing net-

* Research Institute of the Deutsche Bundespost Telekom, Am Kavalleriesand 3, D-6100 Darmstadt, DBR. Email: xbr3de6y@ddathd21.bitnet
† Informatik IV, Universität Dortmund, P.O. Box 50 05 00, D-4600 Dortmund 50, DBR.

works. The networks consist of multi-class delay-loss stations of type $\sum_i \mathrm{MAP}_i^{d_i} / \overrightarrow{\mathrm{PH}} / n/m$ and allow state-dependent routing (cf. [23]). MACOM implements an analysis based on Markovian techniques. The basic idea is to evaluate the performance of a communication system, e.g., time- or arrival-stationary probabilities and throughputs or loss rates, by calculating the steady-state vector of the underlying finite CTMC of a model.

MACOM combines advanced software techniques and the technology of modern workstations. Its software design is governed by two major principles. First a convenient modeling environment defined in terms of a Markovian model world is combined with a powerful environment for the evaluation of the constructed models. The model world is adapted to the requirements of B-ISDN and provided by a graphical user interface (cf. [23]). Secondly, all solution techniques of MACOM are completely based on advanced numerical methods for homogeneous, continuous-time Markov chains with finite state spaces (cf. [15]). The specification of experiments, that determine the evaluation of the required model characteristics, and the representation of results is also supported by the graphical interface.

The performance evaluation based on the steady-state vector of the underlying CTMC is a complex process that must be performed in several steps. Therefore, MACOM has a layered software structure formed by three different levels of abstraction, called High (HLL), Medium (MLL) and Low Level Language (LLL) representation (cf. Fig. 1). The HLL reflects an application-oriented view of the model. It specifies the model and its relevant performance measures in terms of a graphical representation. The MLL arises from a construction-oriented view. It describes the model and its parameters in terms of an event-oriented representation, i.e., by states and transitions caused by all feasible events. The LLL is a mathematical-oriented view that represents the model in terms of the generator matrix of its underlying CTMC.

In the HLL the model and its required characteristics are specified by combining predefined graphical elements of the Markovian model world (cf. [23]). Then the HLL is transformed into the MLL substituting each graphical item by an associated vector of state variables, a set of initial states and transition rules describing the behavior of the component. In the MLL the model is described by a state-transition scheme. It comprises the products of all state descriptions and initial states and the finite set of all transition rules. These rules specify whether a transition of a specific type may occur in a state and how the new state is determined. Additionally nonnegative real numbers are attached to each transition. They depend on its type and describe the transition rates or branching probabilities associated with a transition. Performance measures are represented by random variables on the state space, i.e., by functions on the state variables and transitions between the corresponding states (cf. [22]).

The second transformation of MLL into LLL is logically divided into two steps, namely, the generation of the state space and the construction of the generator matrix $Q \in \mathbb{R}^{n \times n}$. This transformation generates the lists of all possible states of the CTMC and all feasible transitions between these states. Its result may be represented by a directed attributed graph (S, T) comprising the states as nodes and the transitions as edges. Attributes of a node are the values of the vector of state variables, attributes of an edge are the types of the occuring transitions including the corresponding rates. Loops and multiple links between states are admitted. Applying the transition rules successively to all initial states and their successors similar to attributed graph grammars, where the attributes of the state variables control the selection of all applicable transition rules, the state space of the underlying CTMC is incrementally generated (cf. [22]). This generation procedure may be considered as breadth-first-search in the fictitious graph representation of the state space. Starting in the initial states, all reachable states and transitions will be

generated and visited exactly once. In the case of a finite state space the procedure terminates. To determine whether a reached state is new, a search in the set of already generated states is required. This operation is the most complex issue of the algorithm that must be performed very efficiently. Therefore, hashing is used as basic technique in MACOM. The hash function is automatically generated based on the structure of the states. The efficiency of this process is estimated by the number of generated states or transitions per CPU second. The figures in Table 1 are obtained on a Sun 4/110. The experiments with MACOM show that the generation of the state space is less time consuming than the numerical solution phase.

TABLE 1
Generation rates of MACOM for the central server model (see Sect. 6.1)

(n,k)	#states	#transitions	CPU sec.	$\frac{\#states}{CPU sec.}$	$\frac{\#transitions}{CPU sec.}$
(7,1)	204	1120	9	22.7	124.4
(7,3)	484	2506	11	44.0	227.8
(7,8)	1184	5971	12	98.7	497.6
(17,1)	2109	13770	13	162.2	1054.2
(17,3)	5679	34476	18	315.5	1915.3
(17,8)	14604	86241	27	540.9	3194.1
(37,1)	19019	133570	31	613.5	4308.7
(37,3)	54169	350316	71	742.0	4789.8
(37,4)	71744	458689	99	724.7	4633.2
(39,4)	83760	536991	123	681.0	4365.8

Finally, the construction of the generator matrix Q consists of the enumeration of all states according to a fixed ordering, the evaluation of the matrix elements and their integration into a data structure arising from a given sparse matrix storage scheme. The enumeration is represented by a position mapping $p : S \longrightarrow \{1, ..., |S|\} = S_{\mathbb{N}} \subset \mathbb{N}$ on the set S with $|S| = n$ states. The simplest enumeration is determined by the order of generation. But macro structures of the model can also be taken into account. If the states of a macrostate are enumerated in a sequential order and macrostates are ordered in a lexicographical way, a block structured matrix Q results. This process may be represented by an appropriate partition and permutation on $S_{\mathbb{N}}$. Then the generator matrix Q is built determining first its zero structure, i.e., the incidence matrix. Each pair (s_s, s_d) of directly connected states yields a matrix entry in position $(p(s_s), p(s_d))$. The corresponding value is computed according to the type of the applied transition $s_s \rightarrow s_d$ and the attributes of the states. Values of multiple edges between states are added. The diagonal elements are set to the negative sums of the off-diagonal elements of the rows.

Similar to other applications communication models determine generator matrices Q that are large, sparse, nonsymmetric Metzler-Leontief matrices with zero row sums (cf. [2]). As it is more convenient to use the transposed generator matrix in numerical algorithms, Q^t is built up by MACOM in the construction process. It is represented by a row-oriented sparse matrix storage scheme based on a list of the rows of Q^t. One stored row contains the value of the diagonal element, an array for the nonzero off-diagonal elements of that row and an array for the corresponding column indices. This storage scheme is similar to one employed in ITPACK (cf. [13]), but it uses dynamic memory allocation.

The stationary distribution π of the CTMC is computed as solution of the linear system $Q^t \cdot \pi = 0, \sum_{i=1}^{n} \pi_i = 1$. It is unique if Q is irreducible. In MACOM this condition can be verified automatically using Tarjan's algorithm. Then the required performance characteristics, that are defined in terms of this steady-state vector π and specified by an evaluation description, are automatically computed by MACOM. Further details concerning this forward and backward transformations are discussed in [23] and [22].

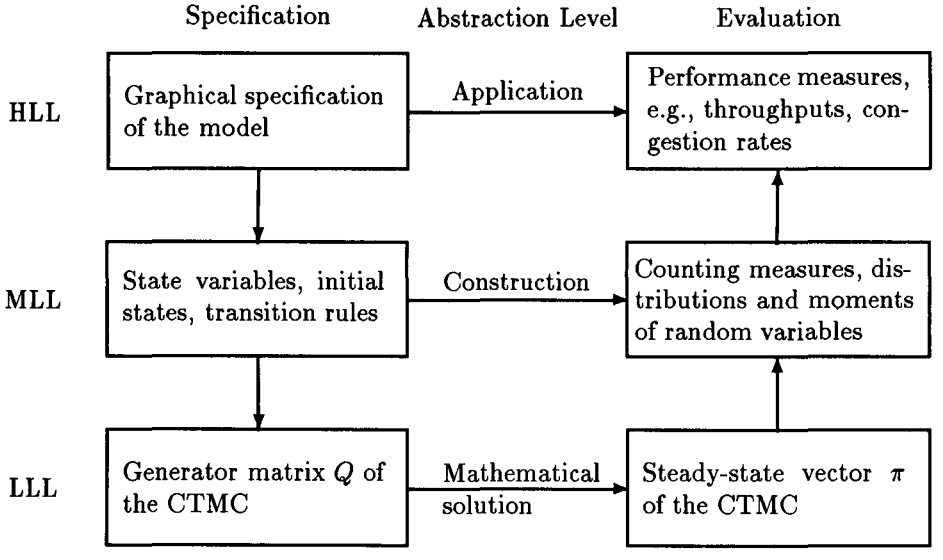

FIG. 1. *The structure of MACOM*

3. The mathematical background of computational solution methods for finite Markov chains. In the following we assume the reader to be familiar with the theory of M-matrices to the extent of the book of Berman and Plemmons [2] and we adopt the notation used therein (see also [15]).

In the previous section we have briefly outlined a methodology to evaluate the performance of telecommunication systems by means of Markovian queueing networks. From a mathematical point of view, this task may be reduced to the issue of calculating the stationary distribution π of the underlying finite CTMC by numerical methods. Let us assume without loss of generality that the CTMC is irreducible. Its corresponding irreducible, nonsymmetric generator matrix $Q \in \mathbb{R}^{n \times n}$ may be generated automatically by MACOM. Then we have the task to compute the solution of the linear system

$$A \cdot x = 0, \qquad x \in \mathbb{R}^n \tag{1}$$

$$\sum_{i=1}^{n} x_i = e^t \cdot x = 1 \tag{2}$$

where e denotes the vector of all ones and $A = -Q^t \in \mathbb{R}^{n \times n}$ is a singular, nonsymmetric, irreducible M-matrix with zero column sums, i.e., $e^t A = 0$. As $\lambda = 0$ is a simple eigenvalue of A, the equations (1), (2) have a unique positive, normalized solution $x \gg 0$ (cf. [2]). It coincides with the steady-state distribution π of underlying CTMC.

Numerical solution methods for the system (1) have been classified and intensively discussed in [15], [19], and [1]. Therefore, we subsequently describe only those direct and iterative methods

that were implemented in the tool MACOM.

4. Direct solution methods. Direct solution techniques based on Gaussian elimination without pivoting are employed by MACOM to calculate the positive solution vector $x \gg 0$ of the homogeneous system $-A \cdot x = Q^t \cdot x = 0$. The following algorithms are applied:

- Crout's LU decomposition algorithm according to Funderlic and Mankin [6] (cf. [15])
- modified Gaussian elimination according to Grassmann et al. [9], called Grassmann's algorithm (cf. [15])

Both algorithms first compute a decomposition $Q^t = L \cdot U$ of the irreducible, singular, diagonally dominant Metzler-Leontief (ML) matrix Q^t into a regular, lower triangular ML-matrix L and a singular, upper triangular M-matrix U with $U_{nn} = 0$ (cf. [2, Cor. 6.4.17]). In Grassmann's variant of Gaussian elimination, however, each pivot element is calculated as negative sum of all off-diagonal elements in the column below the pivot (cf. [9], [15]). Then the modified system $(U + e_n e_n^t) \cdot x = e_n$ is solved by back substitution, where e_n denotes the nth unit vector (cf. [6], [11]). Finally, a normalization of the resulting vector $x \gg 0$ yields the steady-state distribution $\pi = x/(e^t x)$.

From a computational point of view, both algorithms are stable. An error analysis of Crout's algorithm is provided by [11] and [7]. Grassmann's algorithm has the additional advantage that it uses only nonnegative numbers, thereby avoiding cancellation errors during the elimination process.

A row-oriented sparse matrix storage scheme, similar to one used in ITPACK, is employed as data structure in the LU decomposition algorithm (cf. [13]). Considering Grassmann's variant of Gaussian elimination, the storage scheme of the matrix must provide access by rows and columns. Therefore, a representation by two separate linked lists for rows and columns is used. Although this storage scheme includes a larger overhead compared to Crout's algorithm, Grassmann's procedure is generally recommended as direct solution method for matrices of order $n > 300$. It may be superior to Crout's algorithm in some cases due to the avoidance of cancellation errors.

An important problem of direct solution techniques is fill-in that strongly influences the requirements on storage and time of the solution process. In case of the automatic selection of a solution technique offered by MACOM, direct methods are only applied if a simple fill-in indicator $F_{Q^t}^b$ does not exceed a given bound. It is defined by

$$F_{Q^t}^b = \sum_{i=1}^{n} (\min(\max_{1 \le r \le i} (c_{Q^t}^{\max}(r) - r), n - i) + \max_{1 \le r \le i} (r - c_{Q^t}^{\min}(r)) + 1)$$

where n denotes the order of Q and $c_{Q^t}^{\min(\max)}(r)$ is the smallest (largest) column index of row r with a nonzero element. $F_{Q^t}^b$ is an upper bound for the fill-in. It may be computed with a storage requirement of $O(1)$ by scanning all rows.

Direct methods are recommended for the solution of a small system of order $n \le 500$ or a system with small bandwidth, provided that the underlying CTMC is not a nearly completely decomposable Markov chain (NCD-model) (cf. [5]). If it is of NCD-type, A is ill-conditioned since there exists a cluster of eigenvalues near the single eigenvalue $\lambda = 0$ (cf. [8], [16]). In this case, direct methods should be applied very carefully and, in general, iterative solution techniques are recommended.

5. Iterative solution methods.

5.1. Mathematical aspects. Normally a Markovian model generated by MACOM can have a very large state space comprising up to 100000 states. If the corresponding generator matrix Q is large and sparse, of NCD-type, or a banded matrix with large bandwidth, iterative methods based on a regular splitting $A = M - N$ of the singular, nonsymmetric, irreducible M-matrix $A = -Q^t$ are employed in MACOM. They generate the iterative scheme

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

based on the nonnegative iteration matrix $T = M^{-1} \cdot N$. Let $\rho(T)$ denote the spectral radius of T. As the iteration matrix has a simple eigenvalue $\rho(T) = 1$, it is necessary to apply the theory of semiconvergent matrices to prove the convergence of the scheme (3) (cf. [20]). A survey of necessary and sufficient conditions for convergence is given in [1] and [15].

In MACOM standard iterative schemes based on the point Jacobi and point or block Gauss-Seidel splitting, i.e., the point Jacobi, the extrapolated point Jacobi (JOR), the point Gauss-Seidel and point SOR as well as block Gauss-Seidel and block SOR procedure, are implemented. All schemes may be accelerated by inserting optional aggregation-disaggregation (A/D) steps during the iteration (cf. [14], [21]). A unifying mathematical framework for these A/D-accelerated iterative procedures derived from regular splittings $A = -Q^t = M - N$ is provided by [14]. In this paper an error analysis for iterative A/D methods and a stochastic interpretation of block iterative procedures are derived.

Regarding the application of iterative solution methods, the selection of a suitable splitting and its corresponding procedure is an important issue. Besides the numerical efforts to solve the systems $Mx^{(k+1)} = Nx^{(k)}$ during the iteration, the most important criterion is given by the rate of convergence. It is determined by the modulus of the subdominant eigenvalues $\delta(T) = \max\{|\lambda| \mid \lambda \in \sigma(T) \setminus \{1\}\}$ of the iteration matrix T. Here $\sigma(T)$ denotes the spectrum of T. Recall that $\delta(T)$ is less than one if the scheme is convergent (cf. [2, p. 152]). If A is a regular M-matrix, the choice of a splitting is governed by Varga's comparison theorem [26, Th. 3.15, p. 90]. Informally speaking it states that the rate of convergence increases if the term N of a regular splitting decreases. Its proof is based on the basic relation (cf. [26, Th. 3.13, p. 89]):

$$M^{-1}N = (I + A^{-1}N)^{-1}A^{-1}N \tag{4}$$

Regarding the class of regular splittings $A = M - N, M^{-1} \geq 0, N \geq 0$, of a singular, irreducible M-matrix A, this comparison result no longer holds (cf. [12]). But it is possible to prove a relation similar to (4).

Recall that the column sums of A are supposed to be zero. We denote the unique pair of normalized, positive left and right eigenvectors corresponding to the simple eigenvalue 0 of A by e, i.e., the vector of all ones, and by π. The normalization implies $\pi^t e = 1$. Then the following relations hold for the iteration matrix $T = M^{-1}N \geq 0$:

$$\begin{aligned} TAA^{\#} &= (I+Y)^{-1}Y \\ Y &= (A + M\pi e^t)^{-1}NAA^{\#} = (A + M\pi e^t)^{-1}N(I - \pi e^t) \\ &= A^{\#}N(I - \frac{\pi e^t N}{e^t N \pi}) - \pi e^t (I - \frac{N}{e^t N \pi}) \end{aligned} \tag{5}$$

Here, $A^{\#}$ denotes the group inverse of A and $AA^{\#} = I - \pi e^t$ is the projector on the range $R(A)$ of A along the null space $N(A)$ of A (cf. [2, p. 119]). Obviously, $A^{-} = (A + M\pi e^t)^{-1}$ is a

regular 1-inverse of A with $A^-A = AA^\#$. Moreover, $R(A^-A) = R(AA^\#) = R(A)$ and A^- is nonnegative on the range of A, i.e., $A^-u \geq 0$ for $u \in R(A), u \geq 0$ (cf. [2, La. 7.6.18, p. 199]). We feel that this result is a more natural extension of parts of Varga's theorem [26, Th. 3.13, p. 89] than the corresponding one derived by Meyer and Plemmons [17, Th. 1, p. 703] (cf. [2, Ths. 7.6.19, 7.6.20, p. 200]). Taking into account the relationship $\delta(T) = \rho(TAA^\#)$, we conclude from (5) that

$$(6) \qquad \delta(T) = \max_{\lambda \in \sigma(TAA^\#)} |\lambda| = \max_{\mu \in \sigma(A^-NAA^\#)} \left|\frac{\mu}{1+\mu}\right| \leq \max_{\mu \in \sigma(A^-NAA^\#)} \left|\frac{\mu}{1-|\mu|}\right|$$

holds. If $\sigma(T) \subseteq \mathbb{R}$, hence $\sigma(A^-NAA^\#) \subseteq \mathbb{R}$, holds, then it follows

$$\delta(T) \leq \max\{\frac{\mu^-}{1-\mu^-}, \frac{\mu^+}{1+\mu^+}\}$$

for

$$\mu^- = \max\{|\mu| \mid \mu \in \sigma(A^-NAA^\#), -1/2 \leq \mu < 0\},\ \mu^+ = \max\{\mu \mid \mu \in \sigma(A^-NAA^\#), 0 \leq \mu\}.$$

Although relation (6) reveals the influence of the splitting on the rate of convergence, we could not derive a simple comparison result such as [26, Th. 3.15, p. 90] in case of a general singular, irreducible M-matrix A. As other theoretical results are not available, the empirical comparison of different iterative schemes based on given test matrices seems to be the only alternative, especially if an acceleration by inserting A/D steps is used. Note that it is possible now that the Jacobi procedure converges faster than the Gauss-Seidel procedure. Moreover, the convergence and its speed strongly depend on the ordering of states, especially in the case of a p-cyclic M-matrix A (cf. [18], [12]).

5.2. Implementation aspects. Regarding the actual implementation of these iterative schemes in MACOM, we have to stress that all procedures are based on a point Jacobi splitting and a point or block Gauss-Seidel splitting of the singular, irreducible Metzler-Leontief matrix Q^t. For point iterative schemes a scaling $D^{-1}Q^t$ of the rows by a diagonal-positive matrix D, $D_{ii} = -Q_{ii} > 0, D_{ij} = 0, i \neq j$, is used to avoid time consuming divisions during the iteration. The basic data structure of the procedures is a row-oriented matrix storage scheme similar to one used in ITPACK (cf. [13]). The test of convergence is based on a stopping criterion that is defined in terms of the error estimate

$$e^{(k)} = d^{(k)} \cdot \max(1, \frac{r^{(k)}}{1-r^{(k)}}).$$

It takes into account the magnitude of both the absolute and the relative increments of the iteration vector denoted by

$$d^{(k)} = \|x^{(k)} - x^{(k-1)}\|_\infty \text{ and } r^{(k)} = \frac{\|x^{(k)} - x^{(k-1)}\|_\infty}{\|x^{(k-1)} - x^{(k-2)}\|_\infty}.$$

Note that we suppose $r^{(k)} < 1$ in the definition of $e^{(k)}$ and recall that $r^{(k)}$ converges to $\delta(T)$. Therefore, $r^{(k)}$ is called the reduction factor. In cases of very slow convergence the difference between $e^{(k)}$ and $d^{(k)}$ can be some orders of magnitude. To avoid overflows, the iteration vector $x^{(k)}$ is normalized whenever a convergence test is performed. As this check is not performed in each iteration step, the expensive normalization can be avoided. The iteration is stopped if one of the following three conditions set by a user is fulfilled:

1. The error estimate $e^{(k)}$ is less than the required accuracy.
2. An upper bound for the number of iteration steps is reached.
3. An upper bound for the used CPU time is exceeded.

In MACOM a suitable, positive initial vector $x^{(0)}$, e.g., the result of a preceding experiment in a series of experiments, can be used to speed up the convergence. Furthermore, iterative procedures may be accelerated either by applying relaxation techniques or by inserting some aggregation-disaggregation steps during the iteration. In case of SOR, the relaxation parameter is automatically adjusted based on the heuristic estimation procedure for consistently ordered 2-cyclic matrices developed by Hageman and Young [10, §9.5, p. 223].

The insertion of A/D steps is an alternative approach (cf. [14], [21]). It is based on the idea to approximate the new iteration vector $x^{(k+1)}$ by extending the solution $\alpha(x^{(k)}) \in \mathbb{R}^p$ of an aggregated, smaller system $B(x^{(k)})\alpha(x^{(k)}) = \alpha(x^{(k)})$ to the whole state space. In a way, B describes the behavior of a new system that is derived from the aggregation of the original state space into a smaller set $\{1,\ldots,p\}$ of macro states defined by the partition of the index set $S_{\mathbb{N}}$. This technique is determined by an appropriate ordering of states in the Markov chain. It yields a block partition $Q^t = (Q^t_{ij})_{1\le i,j\le p}$ of the generator matrix that is related to the structure of the Markovian model. In MACOM a default partition based on the number of customers in the network is defined. If no batch arrivals occur in the queueing network, the lexicographical ordering w.r.t. the population component yields a block tridiagonal generator matrix. Other predefined partitions, e.g., one based on the population of customer classes or on the phase of an arrival process, can be selected by the user.

It is the objective to find suitable partitions that yield fast iterative schemes. As no theoretical results regarding the optimal ordering of states and the partitioning of the state space are available, heuristic procedures are the only alternative. In MACOM A/D steps may be inserted automatically during the iteration if a block structure has been determined and the convergence is relatively slow, i.e., for $r^{(k)} > 0.9$. Between A/D steps some normal iteration steps are performed. If an A/D step does not change the iteration vector significantly, the number of iterations between these steps is increased. A/D steps and block iterative schemes require the solution of smaller systems defined in terms of the given block matrices. In MACOM LU factorization is used for the solution of subsystems. It can be performed very efficiently since the zero structures of the matrices L and U do not change during the iteration. Therefore, the data structures can be reused in each A/D step.

6. The application of numerical solution methods. In this section we illustrate by two examples the benefits and difficulties of the proposed numerical solution methods for finite CTMCs. Regarding the application of analysis algorithms in a tool like MACOM in general, it is very important that the procedures are suitable for an automatic selection by the tool itself. Moreover, their features should facilitate the setting of control parameters by defaults if users are unexperienced. On the other hand, the algorithms should possess some flexibility such that their adaptation to a given model and their tuning by experienced users are possible, for instance, by setting control parameters like the relaxation factor in an SOR scheme.

6.1. Selected communication models. Subsequently we use two well-known communication models to reveal the features of the described numerical solution methods. The first model, provided by [19], is a retrial queue that describes a telephone exchange (station $S2$) with impatient customers. Each customer waiting for a ringing tone may become impatient if his post-dialing delay is too long. Then he reattempts his call request after a random period

(modeled by station $S1$) or he gives up with a fixed probability $1-h$ (see Fig. 2). The number of customers in each station Si is denoted by $X_i, i \in \{1,2\}$. The arrival stream of new customers is modeled by a Poisson process with intensity A.

The exchange is described by a special $G/M_n/1/K2$ delay-loss system $S2$ where requests are lost if the system is occupied. If $X_2 = j$ requests are pending in $S2$, a successful customer gets the ringing tone and departs from the queueing network at $S2$ in an interval of infinitesimally small length dt with probability μdt. He does not get the tone and decides to reattempt with probability $j\tau dt$. This means, one of the $X_2 = j$ pending requests is processed with service rate μ and each corresponding customer may become impatient with rate τ. The fictitious orbit for impatient customers, where they are waiting before initiating a new call request, is a $G/M/K1/K1$ loss system $S1$. The time until the retrial of a customer is exponentially distributed with mean $1/\lambda$.

The parameters are set to $A = 0.6$, $\mu = 1.0$, $\tau = 0.05$, $h = 0.85$ and $\lambda = 5.0$. The capacities $K1$ and $K2$ can be varied yielding different sizes of the state space. They should be large enough such that congestion effects are negligible.

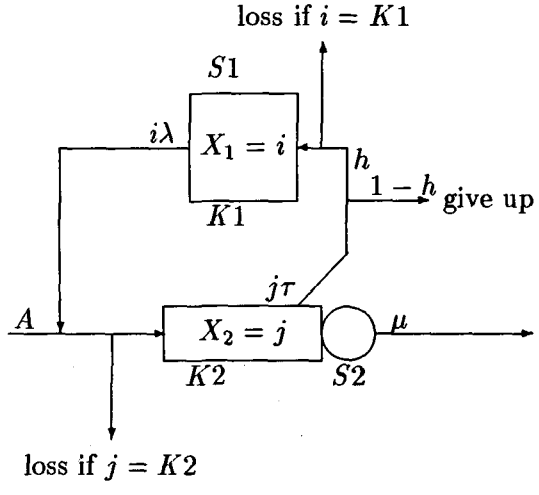

FIG. 2. *Queueing model with impatient customers*

The second model is a modified central server system that arises from modeling the exchange of information between an I/O subsystem and a computer (cf. [25, p. 382]). It is a closed queueing network with a fixed population of size n consisting of 4 stations $S1, \ldots, S4$ of type $\cdot/GI/1/\infty$ (see Fig. 3). The service times at $S1$ and $S4$ are exponentially distributed with rates μ_1 and μ_4, respectively. These of $S3$ are governed by an Erlang distribution with k phases and those of $S2$ by a Coxian distribution with two phases with the same rates. The coefficient of variation of this Coxian service-time distribution is an adjustable parameter of the model and set to 1.5. The mean service times at $S2$, $S3$ and $S4$ are set to 0.5. The size of the state space is determined by the number n of customers in the network and the number k of Erlangian phases at $S3$. The NCD property of the model is controlled by the routing probability p. To reduce the influence of p on the distribution of the population at $S1$, the service rate μ_1 is set to $1-p$.

6.2. Application of direct methods. In order to calculate the steady-state vector of a generated Markovian model, it is first necessary to decide whether a direct or iterative solution method should be applied. In MACOM this decision is controlled by the value of the fill-in

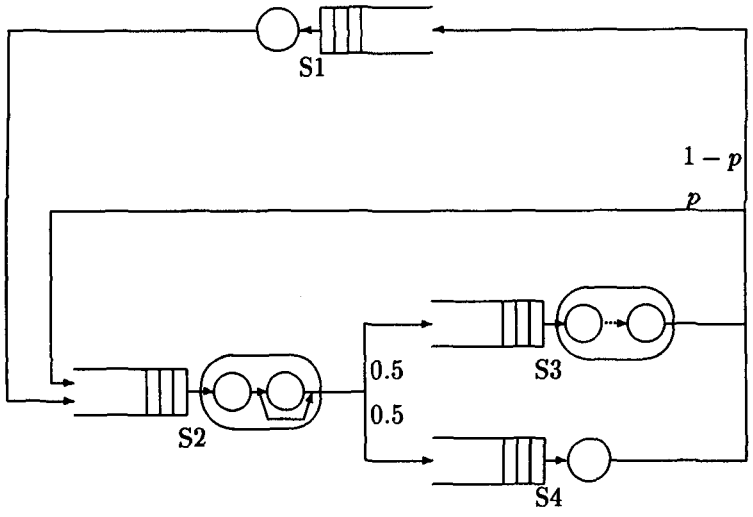

FIG. 3. *Modified central server model*

estimator $F^b_{Q^t}$. Of course, $F^b_{Q^t}$ strongly depends on the enumeration of states. Using four distinct enumeration schemes in the central server model with 2109 states, for instance, $F^b_{Q^t}$ varies between 543473 and 840664. The actual fill-in varies between 349360 and 446871. In most cases, the natural enumeration induced by the order of generation produces rather low fill-in since most models possess some kind of locality that is implicitly exploited during the automatic construction process.

Regarding the exchange model with impatient customers, given the capacities $K1 = 10$ and $K2 = 220$, LU factorization yields a solution vector with negative values due to cancellation errors. Considering the modified central server model with a low degree of coupling between $S1$ and the rest of the network for $p = 0.99999999$ and a state space with 2109 states, the result of LU factorization also becomes inaccurate due to the NCD property of the network. Moreover, the error depends on the used enumeration. If we apply, however, Grassmann's algorithm, no difficulties arise. It seems to be more stable, even for some NCD systems, if it is applied carefully (cf. [16]).

6.3. Application of iterative methods. Compared to direct solution techniques, iterative methods require much more monitoring and control w.r.t. the progress of the solution process. For instance, the stopping criterion for convergence must be evaluated, decisions on control parameters such as the relaxation parameter in SOR have to be taken, and after termination the accuracy of the approximate solution vector should be estimated. Furthermore, in the case of semiconvergent iteration matrices it is very important that the selected iterative procedures are carefully adapted to the algebraic structure of an investigated Markovian model to achieve a large asymptotic rate of convergence, i.e., to guarantee fast convergence. Compared to iterative methods with few dynamic control parameters such as preconditioned CG procedures (cf. [19]), iterative schemes derived from regular splittings of the singular M-matrix $-Q^t$, e.g., point and block SOR with optional A/D steps, require more monitoring and control activities. For unexperienced users or simple black box solvers it seems to be a disadvantage of these procedures that the control parameters have to be chosen very carefully. On the other hand, we are convinced that these features and the flexibility of splitting methods offer the chance to improve

the convergence of the corresponding iterative algorithms considerably if the control parameters are carefully adapted to the underlying algebraic and stochastic structure of a given Markovian model.

As mentioned previously, slow convergence is the main difficulty of iterative procedures. In most cases, it has the reason that the selected method and its control parameters, e.g., the relaxation factor of SOR, are not very well adapted to the structure of the model. This behavior may be caused by an undetected NCD property of the investigated CTMC. This problem can be illustrated by the modified central server model with a weak coupling of station $S1$ if $p = 0.9999$ is used (see Table 2). After 10000 iteration steps the throughput at station $S1$ still has a large

TABLE 2
Behavior of the Gauss-Seidel procedure in case of the central server model

#iteration steps k	$d^{(k)}$	$r^{(k)}$	$e^{(k)}$
20	7.2300e-4	0.96818	2.2002e-2
100	1.2736e-6	0.92055	1.4758e-5
500	1.6296e-6	0.99990	1.6176e-2
1000	1.5505e-6	0.99990	1.5758e-2
5000	1.0828e-6	0.99992	1.3238e-2
10000	7.4970e-7	0.99993	1.1328e-2

error of about 10% which is indicated by the large magnitude of $e^{(k)}$.

MACOM tries to overcome this difficulty by a simple strategy. The idea is to select a point or block SOR procedure with a suitable relaxation parameter and to insert several A/D steps during the iteration. The required partition of the state space, however, must be empirically adapted to the NCD structure of the model which is unknown in most cases.

The effect of three different partitions EQ_1, EQ_2, EQ_3 on the rate of convergence is shown by the central server model with $n = 17$ customers, $k = 3$ service phases at $S3$ and $p = 0.9999$. In this model the state space must be partitioned by a relation EQ_1 defined by the equality of the population at $S1$. EQ_2, that combines three macro states of EQ_1 to a new macro state, is coarser than EQ_1. The macrostates of EQ_3 are defined by the equality of the population at $S4$. Table 3 shows the influence of the chosen partition on the Gauss-Seidel and block Gauss-Seidel scheme. The block structure is derived from the partition used for A/D steps. The results in

TABLE 3
Solution of the central server model with several partitions on a SUN 4/110 by MACOM

Algorithm	partition	#iter. steps	#A/D steps	$d^{(k)}$	$r^{(k)}$	$e^{(k)}$	CPU (sec)
Gauss-Seidel	none	4000	0	1.28e-06	0.99991	1.50e-02	382.9
	EQ_1	181	8	5.14e-13	0.91929	5.85e-12	19.6
	EQ_2	4023	63	6.13e-06	0.99988	4.93e-02	385.5
	EQ_3	4086	84	1.07e-06	0.99992	1.28e-02	389.3
block Gauss-Seidel	EQ_1	80	1	3.76e-13	0.85984	2.31e-12	224.3
	EQ_2	60	1	7.96e-13	0.64910	1.47e-12	1069.8
	EQ_3	4040	80	2.92e-06	0.99998	1.35e-01	579.5

Table 3 show that partition EQ_1 improves the rate of convergence whereas the others do not, with

the exception of EQ_2 in the block Gauss-Seidel scheme. The latter scheme behaves well since it captures the NCD property in the diagonal blocks of the partitioned generator matrix and in the aggregated system. Obviously, the block Gauss-Seidel procedure can overcome slow convergence. Regarding, however, the CPU time used for the block scheme with large macrostates, we see that there is always a trade-off between the rate of convergence and the computational effort which is required to solve the large, linear systems defined by the diagonal matrices of the block scheme. As there is no general rule on the selection of a method, experiments performed on variants of a model with small state spaces are necessary to get insight into the behavior of the CTMC before starting a series of experiments.

The insertion of A/D steps during the iteration is a general strategy of MACOM to improve the rate of convergence, even if a model does not have an NCD structure. In the start-up phase of the iteration this technique may improve the search of a subspace containing the solution vector. This effect is shown by the first model with parameters $K1 = 30$ and $K2 = 550$. A comparison of different solution algorithms is provided by Table 4. The size of the state space is 17081 and all CPU times are obtained on a Sun 4/110. Two partitions are used for the block scheme and the A/D variants. In the first partition $P1$ states belong to the same macrostate if the corresponding total numbers of customers in the system, i.e., $X_1 + X_2 = i + j$, are equal. The second partition $P2$ is coarser. Here states belong to the same macrostate if $X_1 + X_2 = (i+j)/2$ holds where / denotes an integer division. We see that, although the model

TABLE 4

Comparison of the solution algorithms of MACOM on a SUN 4/110 based on the model with impatient customers

Algorithm	part-ition	relax. param.	#iter. steps k	#A/D steps	$d^{(k)}$	$r^{(k)}$	$e^{(k)}$	CPU (sec)
Grassmann	$i+j$							275.3
SOR		1.0	1000		3.03e-12	0.945	5.26e-11	211.8
		1.3	1000		3.72e-13	0.897	3.25e-12	223.4
		auto	920		4.64e-14	0.889	3.72e-13	200.9
SOR with A/D steps	$i+j$	1.0	100	4	1.13e-13	0.686	2.47e-13	54.6
		1.3	132	4	3.44e-15	0.663	6.79e-15	82.3
		auto	100	4	4.17e-14	0.688	9.19e-14	55.6
block SOR	$i+j$	1.0	850		8.49e-14	0.901	7.71e-13	380.5
		1.3	810		5.88e-14	0.811	2.53e-13	381.8
	$(i+j)/2$	1.0	430		7.19e-14	0.814	3.14e-13	236.8
		1.3	400		7.85e-14	0.633	1.36e-13	221.2
block SOR with A/D steps	$i+j$	1.0	68	2	3.21e-13	0.511	3.36e-13	63.4
		1.3	116	1	1.38e-13	0.877	9.82e-13	85.3

does not have an NCD structure, the iterative methods combined with A/D steps converge very well. The aggregated system with partition $P1$ has 581 states. Furthermore, the block SOR scheme converges faster than point SOR, but the CPU time increases, of course, with the block size. As partition $P1$ yields a matrix with small bandwidth 64 supporting the application of a direct solution method, some efficiency may be gained.

The selected examples have illustrated both the difficulties of the procedures implemented in MACOM and the strategies to overcome them. Last but not least, we want to stress that variants of the proposed algorithms based on regular multisplittings are most suitable for an

implementation on vector and multiprocessor computers. From our point of view, this potential of the proposed procedures and their flexibility of adaptation to the structure of a given model represent their major benefits. Therefore, they seem to be superior to preconditioned CG methods that are also applied to solve Markovian queueing models, but have serious drawbacks (cf. [3], [4]).

7. Conclusions. In this paper we have discussed the numerical aspects of a computational approach for modeling and analysis of advanced telecommunication systems by finite continuous-time Markov chains. The concepts were implemented in the tool MACOM. We have pointed out that the task to evaluate the performance of a system based on the steady-state vector of the underlying irreducible Markov chain is, from a mathematical point of view, equivalent to the solution of a linear system with a nonsymmetric, singular, irreducible M-matrix. Then we have described the direct and iterative solution techniques implemented in MACOM. Considering two well-known communication models, the benefits and difficulties of these methods were pointed out.

Regarding the solution of the singular systems arising in this special field of computational probability theory, we have seen that a lot of interesting problems, e.g., the determination of optimal state space partitions of a CTMC and suitable splittings for fast iterative methods, are still unsolved. Nevertheless, we are convinced that the proposed algorithms are most suitable solution methods for finite Markov chains since they possess a great potential of adaptation to the stochastic structure of a model and the features for an efficient implementation on a vector or multiprocessor computer.

REFERENCES

[1] V. A. BARKER, *Numerical solution of sparse singular systems of equations arsing from ergodic Markov chains*, Commun. Statist.-Stochastic Models, 5 (1989), pp. 335–381.

[2] A. BERMAN AND R. J. PLEMMONS, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[3] R. H. CHAN, *Iterative methods for overflow queueing models I*, Numerische Mathematik, 51 (1987), pp. 143–180.

[4] ——, *Iterative methods for overflow queueing models II*, Numerische Mathematik, 54 (1988), pp. 57–78.

[5] P. COURTOIS, *Decomposability*, Academic Press, New York, 1977.

[6] R. FUNDERLIC AND J. MANKIN, *Solution of homogeneous systems of linear equations arising from compartmental models*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 375–383.

[7] R. E. FUNDERLIC, M. NEUMANN, AND R. J. PLEMMONS, *LU decomposition of generalized diagonally dominant matrices*, Numerische Mathematik, 40 (1982), pp. 57–69.

[8] G. H. GOLUB AND C. D. MEYER JR, *Using the LU factorization and group inversion to compute, differentiate and estimate the sensitivity of stationary probabilities for Markov chains*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 273–280.

[9] W. GRASSMANN, M. TAKSAR, AND D. HEYMAN, *Regenerative analysis and steady-state distributions for Markov chains*, Operations Research, 33 (1985), pp. 1107–1116.

[10] L. HAGEMAN AND D. YOUNG, *Applied Iterative Methods*, Academic Press New York, 1981.

[11] W. J. HARROD AND R. J. PLEMMONS, *Comparison of some direct methods for computing stationary distributions of Markov chains*, SIAM J. Stat. Comput., 5 (1984), pp. 453–469.

[12] L. KAUFMAN, *Matrix methods for queueing problems*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 525–552.

[13] D. R. KINCAID, T. C. OPPE, AND D. M. YOUNG, *Vector computations for sparse linear systems*, SIAM J. Alg. Disc. Meth., 7 (1986), pp. 99–112.

[14] U. R. KRIEGER, *Analysis of a loss system with mutual overflow in a Markovian environment*, in [24].

[15] U. R. KRIEGER, B. MÜLLER-CLOSTERMANN, AND M. SCZITTNICK, *Modeling and analysis of communication systems based on computational methods for Markov chains*, IEEE Journal on Selected Areas in Communications, 8 (1990).

[16] C. D. MEYER, *Stochastic complementation, uncoupling Markov chains and the theory of nearly reducible systems*, SIAM Review, 31 (1989), pp. 240–272.

[17] C. D. MEYER AND R. J. PLEMMONS, *Convergent powers of a matrix with applications to iterative methods for singular linear systems*, SIAM J. Numer. Anal., 14 (1977), pp. 699–705.

[18] D. MITRA AND P. TSOUCAS, *Relaxations for the numerical solutions of some stochastic problems*, Commun. Statist.-Stochastic Models, 4 (1988), pp. 387–419.

[19] B. PHILIPPE, Y. SAAD, AND W. STEWART, *Numerical methods in Markov chain modeling*, Technical Report, TR-89-21, North Carolina State University, Raleigh, NC 27695, 1989.

[20] H. SCHNEIDER, *Theorems on M-splittings of a singular M-matrix which depend on graph structure*, Linear Algebra and its Applications, 58 (1984), pp. 407–424.

[21] P. SCHWEITZER, *A survey of aggregation-disaggregation in large Markov chains*, in [24].

[22] M. SCZITTNICK, *Techniques for the functional and quantitative analysis of Markovian models of computing systems (in German)*, Master's thesis, Universität Dortmund, F.R.G., 1987.

[23] M. SCZITTNICK AND B. MÜLLER-CLOSTERMANN, *MACOM - A tool for the Markovian analysis of communication systems*, in Proceedings of the Fourth International Conference on Data Communication Systems and their Performance, Barcelona, Spain, June 20 – 22, 1990.

[24] W. STEWART, ed., *Proceedings of the First International Conference on the Numerical Solution of Markov Chains, Raleigh, North Carolina, January 8–10, 1990*, Marcel Dekker, New York, 1990.

[25] H. TIJMS, *Stochastic Modeling And Analysis: A Computational Approach*, John Wiley, New York, 1986.

[26] R. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, 1962.