A Comparison of Numerical Splitting-based Methods for Markovian Dependability and Performability Models *

Víctor Suñé and Juan A. Carrasco

Departament d'Enginyeria Electrònica,
Universitat Politècnica de Catalunya,
Diagonal 647, plta. 9, 08028 Barcelona, Spain

Abstract. Iterative numerical methods are an important ingredient for the solution of continuous time Markov dependability models of fault-tolerant systems. In this paper we make a numerical comparison of several splitting-based iterative methods. We consider the computation of steady-state reward rate on rewarded models. This measure requires the solution of a singular linear system. We consider two classes of models. The first class includes failure/repair models. The second class is more general and includes the modeling of periodic preventive test of spare components to reduce the probability of latent failures in inactive components. The periodic preventive test is approximated by an Erlang distribution with enough number of stages. We show that for each class of model there is a splitting-based method which is significantly more efficient than the other methods.

1 Introduction

Continuous time Markov chains (CTMCs) are widely used for dependability modeling. For these models, several measures of interest can be computed from the solution vector of a linear system of equations. Typically, such a system is sparse and may have hundreds of thousands of unknowns, so it must, in general, be solved numerically using an iterative method.

Several currently available tools allow us to solve dependability models. These are, among others, SAVE [7], SPNP [3], UltraSAN [5] and SURF-2 [2]. SPNP uses Successive Overrelaxation (SOR) with dynamic adjustment of the relaxation parameter ω [4]. SAVE uses SOR for the computation of the steady-state probability vector and SOR combined with an acceleration technique [10] for computation of mean time to failure (*MTTF*) like measures. UltraSAN offers a direct method with techniques to reduce the degree of fill-in and SOR, being ω selected by the user. Finally, SURF-2 uses the gradient-conjugate method (see, for instance, [14]).

* This work has been supported by the Comisión Interministerial de Ciencia y Tecnología (CICYT) under the research grant TIC95–0707–C02–02.

R. Puigjaner et al. (Eds.): Tools'98, LNCS 1469, pp. 154–164, 1998.

Several papers have compared numerical methods for solving the linear systems of equations which arise when solving CTMC models. In an early paper [8], performance models are considered and several iterative methods are compared for the computation of the stationary probability vector of an ergodic Markov chain. These methods include Gauss-Seidel (GS), SOR, block SOR and Chebyshev acceleration with GS preconditioning. For SOR, an algorithm based on the theory of p-cyclic matrices [17] is used to select a value for ω. In [14], failure/repair models are considered and SOR with dynamic adjustment of ω also based on the theory of p-cyclic matrices is compared with GS and the power methods, showing that SOR is considerably more efficient specially for the linear systems arising in $MTTF$ computations. In [11] a number of direct and iterative methods are reviewed in the context of performance models. Among others, three spliting-base methods are considered: GS, SOR and symmetric SOR. In [6] the generalized minimal residual method and two variants of the quasi-minimal residual algorithm are compared. In [9], direct and splitting-based iterative methods are considered for solving CTMC models arising in communication systems and the authors suggest to use SOR with suitable values for ω in combination with some aggregation/disaggregation steps.

In this paper we compare splitting-based iterative methods for the solution of linear systems which arise in the computation of the steady-state reward rate ($SSRR$) defined over rewarded CTMC models. We start by defining formally the measure and establishing the linear system which has to be solved. Let $X = \{X(t); t \geq 0\}$ be a finite irreducible CTMC. X has state space Ω and infinitesimal generator $\boldsymbol{Q} = (q_{ij})_{i,j \in \Omega}$. Let r_i, $i \in \Omega$ be a reward rate structure defined over X. The steady-state reward rate is defined as:

$$SSRR = \lim_{t \to \infty} E[r_{X(t)}]$$

and can be computed as

$$SSRR = \sum_{i \in \Omega} r_i \pi_i \ ,$$

where $\boldsymbol{\pi} = (\pi_i)_{i \in \Omega}$ is the steady-state probability distribution vector of X, which is the only normalized ($\|\boldsymbol{\pi}\|_1 = 1$) solution of:

$$\boldsymbol{Q}^T \boldsymbol{\pi} = \boldsymbol{0} \ , \tag{1}$$

where matrix $\boldsymbol{Q}^T$ is singular and the superscript T indicates transpose. The steady-state unavailability is a particular case of $SSRR$ obtained by defining a reward rate structure $r_i = 0$, $i \in U$, $r_i = 1$, $i \in D$, where U is the subset of Ω including the up (operational) states and D is the subset of Ω including the down states.

In this paper we are concerned with numerical iterative methods to solve the linear system (1). Two classes of models will be considered. The first class include failure/repair models like those which can be specified by the SAVE modeling language [7]. Basically, these models correspond to fault-tolerant systems

made up of components which fail and are repaired with exponential distributions. There is an state in which all components are unfailed having only outgoing failure transitions. The remaining states have at least an outgoing repair transition. Note that in this class of models the detection of the failure of a component is assumed to be instantaneous, i.e. all failed components are immediately scheduled for repair. In the second class of models which we will consider, failures of "spare" (inactive) components will be detected only when they are tested. Test of spare components will be assumed to be performed periodically with deterministic intertests times. To be able to use CTMCs to represent such systems the deterministic intertests time will be approximated by a K-Erlang distribution with K large enough to obtain convergence in *SSRR* as K is incremented.

We will analyze and compare GS, SOR and block Gauss-Seidel (BGS). A more complete comparison including an efficient implementation of GMRES (see, for instance, [13]) and *MTTF* like measures can be found in [16]. We will show that GS is guaranteed to converge for (1) when the chain X is generated breadth-first and the first generated state is exchanged with the last generated state. Also, an algorithm to select dynamically ω in SOR will be briefly reviewed. The rest of the paper is organized as follows. Section 2 reviews the iterative methods. Section 3 analyzes convergence issues. Section 4 presents examples and numerical results and Sect. 5 presents the conclusions.

2 Numerical Methods

We are interested in solving a linear system of the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \ , \tag{2}$$

where $\boldsymbol{A} = \boldsymbol{Q}^T$ and $\boldsymbol{b} = \boldsymbol{0}$. In the following we will let n be the dimension of $\boldsymbol{A}$. We next review splitting-based iterative numerical methods which can be used to solve (2).

2.1 Gauss-Seidel, SOR and Block Gauss-Seidel

Splitting-based methods are based on the decomposition of matrix $\boldsymbol{A}$ in the form $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$, where $\boldsymbol{M}$ is nonsingular. The iterative method is then:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b} \ ,$$

where $\boldsymbol{x}^{(k)}$ is the k-th iterate for $\boldsymbol{x}$.

Both GS and SOR are easily derived by considering the decomposition $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{E} - \boldsymbol{F}$, where $\boldsymbol{D}$ is the diagonal of $\boldsymbol{A}$ and $-\boldsymbol{E}$ and $-\boldsymbol{F}$ are, respectively, the strict lower and upper part of $\boldsymbol{A}$. GS is obtained by taking $\boldsymbol{M} = \boldsymbol{D} - \boldsymbol{E}$ and $\boldsymbol{N} = \boldsymbol{F}$. The iterative step of GS can then be described as:

$$\boldsymbol{x}^{(k+1)} = (\boldsymbol{D} - \boldsymbol{E})^{-1}\boldsymbol{F}\boldsymbol{x}^{(k)} + (\boldsymbol{D} - \boldsymbol{E})^{-1}\boldsymbol{b} \ ,$$

or in terms of the components of $\boldsymbol{A}$ as:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}}\Big(-\sum_{j=1}^{i-1} a_{i,j}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k)} + b_i\Big),\ i = 1,\ldots,n\ . \qquad (3)$$

SOR is obtained by taking $\boldsymbol{M} = (\boldsymbol{D} - \omega\boldsymbol{E})/\omega$ and $\boldsymbol{N} = \big((1-\omega)\boldsymbol{D} + \omega\boldsymbol{F}\big)/\omega$. The iterative step of SOR can then be described as:

$$\boldsymbol{x}^{(k+1)} = (\boldsymbol{D} - \omega\boldsymbol{E})^{-1}\big((1-\omega)\boldsymbol{D} + \omega\boldsymbol{F}\big)\boldsymbol{x}^{(k)} + (\boldsymbol{D} - \omega\boldsymbol{E})^{-1}\omega\boldsymbol{b}\ ,$$

or in terms of the components of $\boldsymbol{A}$ as:

$$x_i^{(k+1)} = \omega\, x_i^{\mathrm{GS}} + (1-\omega)x_i^{(k)},\ i = 1,\ldots,n\ ,$$

where x_i^{GS} is the right-hand side of (3).

BGS is the straightforward generalization of GS when the coefficient matrix, the right-hand side and the solution vector of (2) are partitioned in p blocks as follows:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{1,1} & \ldots & \boldsymbol{A}_{1,p} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{p,1} & \ldots & \boldsymbol{A}_{p,p} \end{pmatrix}\ , \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_p \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_p \end{pmatrix}.$$

The iterative step of BGS is:

$$\boldsymbol{x}_i^{(k+1)} = \boldsymbol{A}_{i,i}^{-1}\Big(-\sum_{j=1}^{i-1} \boldsymbol{A}_{i,j}\boldsymbol{x}_j^{(k+1)} - \sum_{j=i+1}^{p} \boldsymbol{A}_{i,j}\boldsymbol{x}_j^{(k)} + \boldsymbol{b}_i\Big),\ i = 1,\ldots,p\ . \qquad (4)$$

Hence, each iteration of BGS requires to solve p systems of linear equations of the form $\boldsymbol{A}_{i,i}\boldsymbol{x}_i = \boldsymbol{z}_i$. Depending on the sizes of the matrices $\boldsymbol{A}_{i,i}$, such systems may be solved using direct or iterative methods.

2.2 An Algorithm for the Optimization of ω in SOR

In this section we briefly describe an algorithm for the optimization of the relaxation parameter ω of SOR. The algorithm does not assume any special property on the matrix of the linear system and searches the optimum ω in the interval $[0, 2]$.

The algorithm is based on estimations of the convergence factor (modulus of the sub-dominant eigenvalue of the iteration matrix). After each iteration k such that the last two iterations have been performed with the same value of ω, the convergence factor η is estimated as:

$$\tilde{\eta} = \frac{\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_\infty}{\|\boldsymbol{x}^{(k-1)} - \boldsymbol{x}^{(k-2)}\|_\infty}\ .$$

Stabilization of $\tilde{\eta}$ is monitored and it is assumed that a good estimate has been achieved when the relative difference in $\tilde{\eta}/(1-\tilde{\eta})$ is smaller than or equal to a given threshold parameter *TOLNI* three consecutive times. For a given ω, except $\omega = 1$, for which no limit is imposed, a maximum of $M = \max\{MAXITEST, est/RATIOETAST\}$ iterations are allocated for the stabilization of $\tilde{\eta}/(1-\tilde{\eta})$, where *est* is the number of iterations required for the stabilization of $\tilde{\eta}/(1-\tilde{\eta})$ for $\omega = 1$. If after M iterations $\tilde{\eta}/(1-\tilde{\eta})$ has not been stabilized, SOR is assumed not to converge for the current ω. Selection of appropriate values for *TOLNI* is a delicate matter. If *TOLNI* is chosen too large an erroneous estimate of the convergence factor may result and the optimization method may become confused. If *TOLNI* is chosen too small non-convergence may be assumed when the method converges but $\tilde{\eta}/(1-\tilde{\eta})$ takes a large number of iterations to stabilize. Selection of values for *MAXITEST* and *RATIOETAST* also involves a tradeoff. If the resulting M is too small, non-convergence may be assumed erroneously. If the resulting M is too large, iterations may be wasted for a bad ω. After some experimentation we found $TOLNI = 0.0001$, $MAXITEST = 150$, and $RATIOETAST = 5$ to be appropriate choices.

The algorithm starts with $\omega = 1$ and, while the estimate for η decreases, makes a scanning in the interval $[1, 2]$ taking increments for ω of 0.1. If a minimum for η is bracketed, a golden search (see, for instance, [12]) is initiated. If for an ω it is found that the method does not converge, the increment for ω is divided by 10 and the search continues to the right starting from the last ω for which the method converged. This process is repeated till the increment for ω is 0.001 (the minimum allowed). If the estimate for η for $\omega > 1$ is found to increase a similar scanning is made to the left in the interval $[0, 1]$. At any point of the algorithm, the best ω is recorded and used till convergence or the maximum number of allowed iterations is reached when the algorithm becomes "lost". The complete algorithm is implemented using an automaton with 13 different states corresponding to different states of the search. Due to lack of space we cannot give a precise description of the automaton, but only highlight the main ideas on which it is based.

3 Convergence

$-\boldsymbol{Q}^T$ is a singular M-matrix and it is well known that SOR converges for $-\boldsymbol{Q}^T\boldsymbol{\pi} = \boldsymbol{0}$ (and, therefore, for (1)) if $0 < \omega < 1$ [15, Theorem 3.17]. We prove next that if X is generated breadth-first and the first state and the last one are exchanged, convergence of GS when solving (1) is also guaranteed.

First we briefly describe breadth-first generation. The initial state is put in an empty FIFO queue. From that point, the generation process continues by taking a state from the queue, generating all its successors and putting in the queue the successors not previously generated. The generation process finishes when the queue becomes empty.

Given the $n \times n$ matrix $\boldsymbol{A}$, its associated directed graph $\Gamma(\boldsymbol{A}) = (V, E)$ is defined by a set of vertices $V = \{1, \ldots, n\}$ and a set of edges $E = \{(i, j) \in$

$V|\, a_{i,j} \neq 0\}$. A sequence of vertices $\alpha = (i_0, i_1, \ldots, i_l, i_0)$ is called a cycle of $\Gamma(\boldsymbol{A})$ if $i_j \neq i_k$, $j \neq k$, $0 \leq j, k \leq l$ and, $(i_k, i_{(k+1) \mod (l+1)}) \in E$, $0 \leq k \leq l$. The cycle is said to be monotone increasing if $i_0 < i_1 < \ldots < i_l$ and it is said to be monotone decreasing if $i_0 > i_1 > \ldots > i_l$ [1].

Theorem 1. *Let $\boldsymbol{Q}$ be the infinitesimal generator of a finite and irreducible CTMC obtained by generating the CTMC breadth-first and exchanging the first and last states. Forward*[1] *Gauss-Seidel converges for the linear system $\boldsymbol{Q}^T\boldsymbol{\pi} = \boldsymbol{0}$ for each initial guess $\boldsymbol{\pi}^{(0)}$.*

Proof. Let $\Gamma(\boldsymbol{Q}) = (V, E)$ be the directed graph associated to $\boldsymbol{Q}$, infinitesimal generator of X, and let $i_n = n$ be the index of the last state of X. Also, let i_{n-1} be any state with $(i_{n-1}, i_n) \in E$. Because of irreducibility of $\boldsymbol{Q}$, some state i_{n-1} exists. More generally, since X is generated breadth-first, each state i_l has a predecessor which was generated before it. Such a precedessor may be i_n (the state from which X is generated) or $i_{l-1} < i_l$. Then, it is clear that a cycle $(i_n, i_k, \ldots, i_{n-2}, i_{n-1}, i_n)$ with $i_{l-1} < i_l$, $1 \leq k < l \leq n$ can be formed in $\Gamma(\boldsymbol{Q})$. Such a cycle becomes $(i_n, i_{n-1}, i_{n-2}, \ldots, i_k, i_n)$ in $\Gamma(\boldsymbol{Q}^T)$, which is monotone decreasing. Then [1, Corollary 1], forward Gauss-Seidel converges for solving $\boldsymbol{Q}^T\boldsymbol{\pi} = \boldsymbol{0}$ for each $\boldsymbol{\pi}^{(0)}$. □

Of course, with $\boldsymbol{\pi}^{(0)} = \boldsymbol{0}$ the method will converge to $\boldsymbol{0}$, a trivial solution of the linear system which is not of interest. Thus, we should start with any $\boldsymbol{\pi}^{(0)} > \boldsymbol{0}$.

Regarding BGS when applied to solve (1), it is known that there always exists a convergent block splitting for $\boldsymbol{Q}^T$ provided that an appropriate ordering of the states is used [8].

As convergence test we require the relative variation on *SSRR* to be smaller than or equal to a specified tolerance ϵ three consecutive times. The rationale for this test is that it takes into account only "important" components of the solution vector.

4 Numerical Results

In this section we compare the performance of the numerical methods to solve (1) using two examples. The first example is a model with failure and repair transitions and immediate detection of component failures; the second example is a model with failure and repair transitions and K-Erlang intertests time of spare components.

For all methods, the relative tolerance for convergence is taken $\epsilon = 1 \times 10^{-8}$ and a maximum of 100,000 iterations is allowed. In all cases, the CTMC is generated breadth-first and, when the GS and SOR methods (SOR reverts to GS when it cannot find an appropriate $\omega \neq 1$) are used for the solution of (1),

[1] The method we have called Gauss-Seidel should be more properly called forward Gauss-Seidel.

the first state (state with all components unfailed) is exchanged with the last generated state so that convergence of GS is guaranteed (Theorem 1). CPU times have been all measured on a Ultra 1 SPARC workstation.

The first example is the distributed fault-tolerant database system depicted in Fig. 1. The system includes two processors, two controllers and three disk clusters, each with four disks. When both processors are unfailed, one of them is in the active state and the other in the spare state. Similarly, when both controllers are unfailed, one of them is active and the other spare. The system is operational if at least one processor, one controller and three disks of each cluster are unfailed. Processors, controllers and disks fail with constant rates 2×10^{-5}, 2×10^{-4} and 3×10^{-5}, respectively. Spare components fail with rate 0.2λ, where λ is the failure rate of the active component. There are two failure modes for processors: "soft" mode, which occurs with probability 0.8, and "hard" mode, which occurs with probability 0.2. Soft failures are recovered by an operator restart, while hard failures require hardware repair. Coverage is assumed perfect for all failures except those of the controllers, for which the coverage probability is C. Uncovered controller failures are propagated to two failure free disks of a randomly chosen cluster. Processor restarts are performed by an unlimited number of repairmen. There is only one repairman who gives preemptive priority first to disks, next to controllers, and last to processors in hard failure mode. Failed components with the same priority are taken at random for repair. Repair rates for processors in soft and hard failure mode are, respectively, 0.5 and 0.2. Controllers and disks are repaired with rates 0.5 and 1, respectively. Components continue to fail when the system is down. The measure of interest is the steady-state unavailability (UA), a particular case of the *SSRR* generic measure. The generated CTMC has 2,250 states and 19,290 transitions. Four values for the coverage probability are considered: $C = 0.9$, 0.99, 0.999, and 0.9999. For this example we only experimented with GS and SOR. The CPU time required for the generation of the model was 0.263 s.

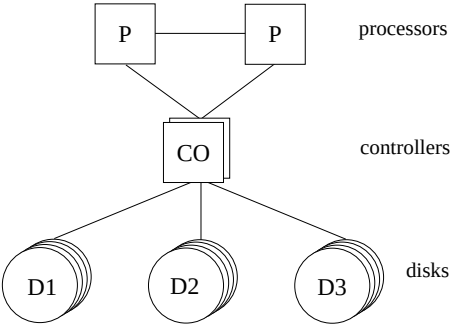

Fig. 1. Distributed fault-tolerant database system

In Table 1 we show the number of iterations and CPU times in seconds for the first example. The GS method is the faster. SOR requires the same number of iterations to achieve convergence as GS because the convergence is so fast that it is achieved before any adjustment on ω can be done. The time per iteration for SOR is slightly greater than for GS.

Table 1. Number of iterations under, respectively, GS and SOR, it_{GS}, it_{SOR}, CPU time in seconds under, respectively, GS and SOR, t_{GS}, t_{SOR}, and UA for the first example

C	it_{GS}	t_{GS}	it_{SOR}	t_{SOR}	UA
0.9	20	0.097	20	0.12	4.054×10^{-5}
0.99	19	0.094	19	0.12	4.461×10^{-6}
0.999	19	0.094	19	0.12	8.537×10^{-7}
0.9999	19	0.094	19	0.12	4.929×10^{-7}

The system considered in the second example is exactly the same as the system of the first example, with the only difference that spare processors and controllers are tested with deterministic intertests times T approximated by a K-Erlang distribution with expected value T with K large enough to make the approximation error small. The measure of interest is again the steady-state unavailability UA. It is clear that the greater T the greater UA. Intuitively, the fact that faulty spare units are not immediately scheduled for repair "increases" the repair times of such units and so increases UA. Then, only values for the intertests time not much greater than the average repair times of the components are reasonable choices. Since the minimum repair rate is 0.2, we consider the following five values for T: 100, 10, 1, 0.1, and 0.01. For the sake of brevity, we will give only results for a coverage probability C equal to 0.99. The value of K is chosen as the minimum value which makes the relative difference between UA for two consecutive K's smaller than or equal to 5×10^{-4}. In Table 2 we show, for each value of T, the number of Erlang stages K, the number of states and transitions of the CTMC X and its generation time.

Table 2. Number of Erlang stages K, number of states, number of transitions and generation time t_{g} in seconds for the second example and $C = 0.99$

T	K	states	transitions	t_{g}
0.01	3	12,000	125,766	1.98
0.1	3	12,000	125,766	1.98
1	6	24,000	251,532	4.14
10	9	36,000	377,298	6.36
100	25	100,000	1,048,050	18.7

The state descriptions of the second example have a component ϕ, $1 \leq \phi \leq K$ used to indicate the phase of the K-Erlang distribution. For BGS, the blocks are chosen to include all states which only differ in the value of the state variable ϕ. In addition, states within each block are sorted following increasing values of ϕ (from 1 to K). With that ordering, the diagonal matrices $\boldsymbol{A}_{ii}$ of BGS have the form:

$$\begin{pmatrix} q_{m,m} & 0 & \dots & 0 & q_{n,m} \\ q_{m,m+1} & q_{m+1,m+1} & 0 & \dots & 0 \\ & q_{m+1,m+2} & q_{m+2,m+2} & \dots & 0 \\ & & \dots & & \\ 0 & \dots & q_{n-2,n-1} & q_{n-1,n-1} & 0 \\ 0 & \dots & 0 & q_{n-1,n} & q_{n,n} \end{pmatrix} .$$

Taking advantage of this form, we solve efficiently the linear systems (4) of BGS using Gaussian elimination with fill-in only in the last column.

The iterative methods considered for the second example are GS, SOR and BGS. Table 3 shows the results obtained. Notice first that although UA tends fast to the value corresponding to instantaneous detection of failed spare components (4.461×10^{-6}), its dependence on T is significant, at least for moderate values of the intertests time. The performance of the numerical methods is also affected by T. For large values of T, the GS method performs very well, but its performance degrades quickly as T decreases. The same type of comments can be made for the SOR algorithm. Note, however, that as the number of iterations required by GS increases, the relative reduction in the number of iterations achieved by SOR is greater. This means that the algorithm used for selecting the relaxation parameter ω is efficient. BGS is the method which requires less iterations. For $T = 100$ it requires more CPU time than GS and SOR. This is due to the time required to sort the states as explained before. For $T = 10$ BGS is as fast as GS and SOR, and for smaller values of T it should be clearly considered as the method of choice. Overall, BGS seems to be the method of choice for the second example.

Table 3. Number of iterations under, respectively, GS, SOR and BGS, it_{GS}, it_{SOR}, it_{BGS}, CPU time in seconds under, respectively, GS, SOR and BGS, t_{GS}, t_{SOR}, t_{BGS}, and UA for the second example, $C = 0.99$ and several values of T

T	it_{GS}	t_{GS}	it_{SOR}	t_{SOR}	it_{BGS}	t_{BGS}	UA
100	21	10.5	21	11.9	10	15.6	6.129×10^{-6}
10	31	5.01	31	5.74	11	4.92	4.641×10^{-6}
1	162	14.8	162	17.5	12	3.15	4.480×10^{-6}
0.1	1,391	58.4	1,045	51.4	12	1.43	4.463×10^{-6}
0.01	12,632	527	5,953	284	12	1.45	4.461×10^{-6}

5 Conclusions

In this paper three splitting-based iterative numerical methods for solving linear systems have been analyzed in the context of two classes of dependability models and the measure *SSRR*. A new and robust algorithm to dynamically tune the relaxation parameter ω of SOR has been briefly described. It has been proved that Gauss-Seidel converges for the solution of the linear system which results when the steady-state probability vector of an irreducible CTMC has to be computed if the CTMC is generated breadth-first and the first and last state are exchanged. Experimental results have shown that the method of choice for each class of model is different (GS for the first class and BGS for the second class).

References

1. G. P. Barker and R. J. Plemmons, "Convergent Iterations for Computing Stationary Distributions of Markov Chains", SIAM J. Alg. Disc. Math., vol. 7, no. 3, July 1986, pp. 390–398.
2. C. Béounes, M. Aguéra, J. Arlat, S. Bachman, C. Bourdeau, J. E. Doucet, K. Kanoun, J. C. Laprie, S. Metge. J. Moreira de Souza, D. Powell and P. Spiesser, "SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems," *Proc. 23rd IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-23)*, Toulouse, 1993, pp. 668–673.
3. G. Ciardo, J. K. Muppala and K. S. Trivedi, "SPNP: Stochastic Petri Net Package", *Proc. 3rd IEEE Int. Workshop on Petri Nets and Performance Models (PNPM89)*, Kyoto, December 1989, pp. 142–150.
4. G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi, "Automated Generation and Analysis of Markov Reward Models Using Stochastic Reward Nets", in C. Meyer and R. Plemmons, editors, *Linear Algebra, Markov Chains and Queuing Models*, IMA Volumes in Mathematics and its Applications, Springer-Verlag 1983, pp. 145–191.
5. J. Couvillon, R. Freire, R. Johnson, W. O. II, A. Qureshi, M. Rai, W. Sanders, and J. Tvedt, "Performability modeling with UltraSAN", *IEEE Software*, September 1981, pp. 69–80.
6. R. W. Freund and M. Hochbruck, "On the Use of Two QMR Algorithms for Solving Singular Systems and Applications in Markov Chain Modeling", Numerical Linear Algebra with Applications, vol. 1, no. 4, 1994, pp. 403–420.
7. A. Goyal, W. C. Carter, E. de Souza e Silva, S. S. Lavenberg, and K. S. Trivedi, "The System Availability Estimator", *Proc. of the 16th Int. Symp. on Fault-Tolerant Computing (FTCS-16)*, 1986, pp. 84–89.
8. L. Kaufmann, B. Gopinath and nE. F. Wunderlich, "Analysis of Packet Network Congestion Control using Sparse Matrix Algortihms", IEEE Trans. on Communications, vol COM-29, no. 4, April 1981, pp. 453–465.
9. U. R. Krieger, B. Műller-Clostermann, M. Sczittnick, "Modeling and Analysis of Communication Systems Based on Computational Methods for Markov Chains", IEEE J. on Selected Areas in Comm., vol. 8, no. 9, December 1990, pp. 1630–1648.
10. P. Heidelberger, J. K. Muppala and K. Trivedi, "Accelerating Mean Time to Failure Computations", IBM Research Report RC–0415, 1996 (to appear in Performance Evaluation).

11. B. Philippe, Y. Saad and W. J. Stewart, "Numerical Methods in Markov Chain Modeling", Operations Research, vol. 40, no. 6, Nov.–Dec. 1992, pp. 1156–1179.
12. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes. The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986.
13. Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems", SIAM J. Sci. Stat. Comput., vol. 7, no. 3, July 1986, pp. 856–869.
14. W. J. Stewart and A. Goyal, "Matrix Methods in Large Dependability Models", IBM Thomas J. Watson Research Center, Technical Report RC-11485, November 1985.
15. W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, 1994.
16. V. S. Suñé and J. A. Carrasco, "A Comparison of Numerical Iterative Methods for Markovian Dependability and Performability Models", Technical Report, Universitat Politècnica de Catalunya, June 1998.
17. R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.