Numerical Iterative Methods for Markovian Dependability and Performability Models: New Results and a Comparison[*][†]

Víctor Suñé, José L. Domingo and Juan A. Carrasco
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya
Diagonal 647, plta. 9, 08028 Barcelona, Spain

Except for formatting details, this version matches exactly the version published with the same title and authors in *Performance Evaluation*, vol. 39, no. 1–4, 2000, pp. 99–125

Abstract

In this paper we deal with iterative numerical methods to solve linear systems arising in continuous-time Markov chain (CTMC) models. We develop an algorithm to dynamically tune the relaxation parameter of the successive over-relaxation method. We give a sufficient condition for the Gauss-Seidel method to converge when computing the steady-state probability vector of a finite irreducible CTMC, an a suffient condition for the Generalized Minimal Residual projection method not to converge to the trivial solution **0** when computing that vector. Finally, we compare several splitting-based iterative methods an a variant of the Generalized Minimal Residual projection method.

Keywords: Continuous-time Markov chains; Dependability; Performability; Iterative numerical methods

[*]Revised and extended version of V. Suñé and J. A. Carrasco, A Comparison of Numerical Iterative Methods for Markovian Dependability and Performability Models, in: R. Puigjaner, N. Savino and B. Serra, eds., *Lecture Notes in Computer Science* 1469 (Springer, 1998) 154–164.

[†]The work has been supported by the "Comisión Interministerial de Ciencia y Tecnología" (CICYT) under the research grant TIC95–0707–C02–02.

1 Introduction

Continuous-time Markov chains (CTMCs) are widely used for dependability and performability modeling. For these models, several measures of interest can be computed from the solution of a linear system of equations. Typically, such a system is sparse and may have many unknowns, making iterative methods attractive for its solution.

Several currently available tools allow the specification and solution of dependability and performability CTMC models. These are, among others, SAVE [12], SPNP [9], UltraSAN [10] and SURF-2 [4]. For the solution of linear systems of equations SPNP uses Successive over-relaxation (SOR) with dynamic tuning of the relaxation parameter ω [8]. SAVE uses SOR for the computation of the steady-state probability vector and SOR combined with an acceleration technique [15] for the 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 conjugate gradient method (see, for instance, [24]).

Several papers have compared numerical methods for solving the linear systems of equations which arise in CTMC models. In an early paper [13], performance models were considered and several iterative methods were compared for the computation of the steady-state probability vector of finite irreducible CTMCs. The methods included Gauss-Seidel (GS), SOR, block SOR, and Chebyshev acceleration with GS preconditioning. For SOR, an algorithm based on the theory of p-cyclic matrices [26] was used to select a value for ω. In [25], failure/repair models were considered and SOR with dynamic tuning of ω, also based on the theory of p-cyclic matrices, was compared with GS and the power methods, showing that SOR was considerably more efficient specially for the linear systems arising in MTTF computations. In [19] a number of direct and iterative methods were reviewed for CTMC performance models. Among others, two projection methods were considered: the Arnoldi's method and the Generalized Minimal Residual (GMRES) method. In [11] GMRES and two variants of the quasi-minimal residual algorithm were compared. In [14], direct and splitting-based iterative methods were considered for solving CTMC models arising in communication systems and the authors suggested using the extrapolated Jacobi method and SOR with suitable values for ω in combination with some aggregation/disaggregation steps.

We consider two measures defined over rewarded CTMC models: the steady-state reward rate (SSRR) and the mean cumulative reward to failure (MCRTF). We start by defining formally the measures and establishing the linear systems which have to be solved. Let $X = \{X(t); t \geq 0\}$ be a finite irreducible CTMC. X has state space Ω and infinitesimal generator $\mathbf{Q} = (q_{i,j})_{i,j\in\Omega}$ ($q_{i,j}$, $i \neq j$ is the transition rate from i to j and $-q_{i,i}$ is the output rate from i). Let r_i, $i \in \Omega$ be a reward rate structure defined over X. The steady-state reward rate is defined as

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

and can be computed as

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

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

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

where matrix $\mathbf{Q}^{\mathrm{T}}$ is singular, the superscript T indicates transpose and $\mathbf{0}$ is a null column vector of appropriate dimension. In order to avoid divisions it is convenient to transform (1) into the linear system

$$\mathbf{P}\boldsymbol{\phi} = \mathbf{0}\,, \tag{2}$$

where $\mathbf{P}$ is the singular matrix $\mathbf{Q}^{\mathrm{T}}[\mathrm{diag}(\mathbf{Q})]^{-1}$, being $[\mathrm{diag}(\mathbf{Q})]$ the matrix with diagonal entries equal to q_{ii} and null off-diagonal entries. The solution vector $\boldsymbol{\pi}$ of (1) is related to the positive normalized ($\|\boldsymbol{\phi}\|_1 = 1$) solution vector $\boldsymbol{\phi}$ of (2) by $\boldsymbol{\pi} = [\mathrm{diag}(\mathbf{Q})]^{-1}\boldsymbol{\phi}/\|[\mathrm{diag}(\mathbf{Q})]^{-1}\boldsymbol{\phi}\|_1$. We note that $\mathbf{P}$ is an M-matrix [24].

The steady-state unavailability (UA) 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.

To define the mean cumulative reward to failure, consider the CTMC X^U with state space $U \cup \{a\}$, where U includes all up states and a is an absorbing state, obtained by directing to state a the transitions to states in which the system is failed, and assume that all states of U are transient. Let $\boldsymbol{\alpha}^U$ be the initial probability distribution of X^U restricted to U and assume that X^U is initially in U with probability 1, i.e. $\sum_{i\in U}\alpha_i^U = 1$. Let r_i, $i \in U$ be a reward rate structure defined on the transient states of X^U. Then, the mean cumulative reward to failure is defined as

$$\mathrm{MCRTF} = E\left[\int_0^{\mathrm{T}} r_{X^U(t)}\mathrm{t}\right]\,, \quad T = \min\{t : X^U(t) = a\}\,,$$

and can be computed as

$$\mathrm{MCRTF} = \sum_{i\in U} r_i \tau_i^U\,,$$

where $\boldsymbol{\tau}^U = (\tau_i^U)_{i\in U}$ is the mean times to absorption vector of X^U. $\boldsymbol{\tau}^U$ can be obtained (see, for instance, [6]) by solving

$$\mathbf{Q}_{UU}^{\mathrm{T}}\,\boldsymbol{\tau}^U = -\boldsymbol{\alpha}^U\,, \tag{3}$$

where $\mathbf{Q}_{UU}$ is the restriction of the infinitesimal generator of X^U to the subset U and matrix $\mathbf{Q}_{UU}^{\mathrm{T}}$ is nonsingular. Again, it is convenient to transform (3) into the linear system

$$\mathbf{P}_{UU}\,\boldsymbol{\nu}^U = -\boldsymbol{\alpha}^U\,, \tag{4}$$

where $\mathbf{P}_{UU}$ is the nonsingular matrix $\mathbf{Q}_{UU}^{\mathrm{T}}[\mathrm{diag}(\mathbf{Q}_{UU})]^{-1}$. The solution vector $\boldsymbol{\tau}^U$ of (3) is related to the solution vector $\boldsymbol{\nu}^U$ of (4) by $\boldsymbol{\tau}^U = [\mathrm{diag}(\mathbf{Q}_{UU})]^{-1}\boldsymbol{\nu}^U$ and $\mathbf{P}_{UU}$ is an M-matrix. The MTTF is obtained as a particular case of MCRTF for the reward rate structure $r_i = 1$, $i \in U$.

The convergence of both GS and SOR may be extremely slow when they are used to solve (4). In [15] an efficient technique is described which improves the convergence of such methods. The technique consists in defining suitable subsets S, T, $U = S \bigcup T$ and then solving either $|T|$ or

$|T|+1$ linear systems depending on whether the initial probability distribution of X^U is concentrated in a single state of T or not, whose nonsingular matrix is $\mathbf{Q}_{SS}^{\mathrm{T}}$, where $\mathbf{Q}_{SS}$ is the restriction of the infinitesimal generator of X^U to S. Next, we briefly describe that technique in a pure algebraic manner for the case $S = U - \{1\}$, $T = \{1\}$, where without loss of generality we assume that state 1 is the state in which all components are unfailed. In that case the following two linear systems are solved:

$$\mathbf{Q}_{SS}^{\mathrm{T}}\,\widetilde{\boldsymbol{\tau}}' = -\beta\,, \tag{5}$$

$$\mathbf{Q}_{SS}^{\mathrm{T}}\,\widetilde{\boldsymbol{\tau}}'' = -\xi\,, \tag{6}$$

where $\beta_i = q_{1,i}/q_{1,1}$, $i \in S$ and $\xi_i = q_{1,i}\alpha_1^U/q_{1,1} + \alpha_i^S$, $i \in S$. Then, $\boldsymbol{\tau}$ is computed as $\boldsymbol{\tau} = h''\boldsymbol{\tau}'/(1-h') + \boldsymbol{\tau}''$, with $\boldsymbol{\tau}' = (q_{1,1}^{-1}, \widetilde{\boldsymbol{\tau}}'^T)$, $\boldsymbol{\tau}'' = (\alpha_1^U/q_{1,1}, \widetilde{\boldsymbol{\tau}}''^T)$, $h' = \sum_{i\in S}\widetilde{\tau}_i' q_{i,1}$, and $h'' = \sum_{i\in S}\widetilde{\tau}_i'' q_{i,1}$. Again, it is convenient to transform (5) and (6) into the linear systems

$$\mathbf{P}_{SS}\,\widetilde{\boldsymbol{\nu}}' = -\beta\,, \tag{7}$$

$$\mathbf{P}_{SS}\,\widetilde{\boldsymbol{\nu}}'' = -\xi\,, \tag{8}$$

where $\mathbf{P}_{SS}$ is the nonsingular matrix $\mathbf{Q}_{SS}^{\mathrm{T}}[\mathrm{diag}(\mathbf{Q}_{SS})]^{-1}$. The solution vector $\widetilde{\boldsymbol{\nu}}'$ of (7) is related to the solution vector $\widetilde{\boldsymbol{\tau}}'$ of (5) by $\boldsymbol{\tau}' = [\mathrm{diag}(\mathbf{Q}_{SS})]^{-1}\widetilde{\boldsymbol{\nu}}'$. Analogously, the solution vector $\widetilde{\boldsymbol{\nu}}''$ of (8) is related to the solution vector $\widetilde{\boldsymbol{\tau}}''$ of (6) by $\boldsymbol{\tau}'' = [\mathrm{diag}(\mathbf{Q}_{SS})]^{-1}\widetilde{\boldsymbol{\nu}}''$. Again, we note that $\mathbf{P}_{SS}$ is an M-matrix. In the particular case in which state 1 of X has initial probability equal to 1, only the linear system (7) needs to be solved and $\boldsymbol{\tau}$ is computed as $\boldsymbol{\tau} = \boldsymbol{\tau}'/(1-h')$ since in that case $\alpha_1^U = 1$, $\alpha_i^S = 0$, $i \in S$ and, therefore, $\boldsymbol{\tau}' = \boldsymbol{\tau}''$. As a final remark, note that since h' can be very close to 1, straight computation of $1-h'$ might result in severe cancellations. It can be shown, though, that

$$1 - h' = \sum_{i\in\Omega-U}\left(\frac{q_{1,i}}{q_{1,1}} + \sum_{j\in S}\tau_j' q_{j,i}\right),$$

so $1-h'$ can be computed safely using only additions of nonnegative numbers.

In [15] it is shown that GS usually converges much faster for both (7) and (8) than it does for (4). We will use the technique in combination with GS, SOR and block Gauss-Seidel (BGS). The resulting methods will be called AGS, ASOR and ABGS, where the prefix “A” stands for accelerated.

In this paper we are concerned with numerical iterative methods to solve the linear systems (2), (4), (7) and (8). Three classes of models will be considered. The first class include failure/repair models like those which can be specified by the SAVE modeling language [12]. Basically, these models correspond to fault-tolerant systems made up of components which fail and are repaired with exponential distributions. There is a 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 immediate, 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 intertest time. To be able to use CTMCs to represent such systems the deterministic intertest time will be approximated by a K-Erlang distribution, with K large enough to obtain convergence in the computed measure as K is incremented. The third class of models is quite wide and includes models with failure, repair and performance transitions.

We will describe an efficient and robust algorithm to dynamically tune ω in SOR with the objective of reducing the number of iterations required to achieve convergence. Moreover, we will give a sufficient condition for GS to converge when solving (2) and a sufficient condition for GMRES not to converge to the trivial solution $\mathbf{0}$ when solving (2). The condition for GS encompasses the very common situation in which the CTMC is generated from a given start state using a set of generation rules and states are numbered increasingly as they are generated. Finally, we will analyze and compare the splitting-based methods GS, SOR and BGS with its accelerated versions, and a variant [23] of GMRES [22] which we will call GMR. The rest of the paper is organized as follows. Section 2 describes the iterative methods and the algorithm to dynamically tune ω in SOR. Section 3 analyzes convergence issues. Section 4 presents examples and numerical results. Section 5 includes the conclusions. Appendix A gives a formal description of the proposed algorithm to dynamically tune ω in SOR.

2 Numerical methods

We are interested in solving a linear system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{9}$$

where $\mathbf{A} = \mathbf{P}$ and $\mathbf{b} = \mathbf{0}$ (2), $\mathbf{A} = \mathbf{P}_{UU}$ and $\mathbf{b} = -\boldsymbol{\alpha}^U$ (4), $\mathbf{A} = \mathbf{P}_{SS}$ and $\mathbf{b} = -\beta$ (7), or $\mathbf{A} = \mathbf{P}_{SS}$ and $\mathbf{b} = -\xi$ (8). In the following we will let n be the dimension of $\mathbf{A}$. We next describe iterative numerical methods which can be used to solve (9). We start by splitting-based methods and next will consider a variant of GMRES.

2.1 Gauss-Seidel, SOR and block Gauss-Seidel

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

$$\mathbf{x}^{(k+1)} = \mathbf{H}\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{b}\,, \tag{10}$$

where $\mathbf{x}^{(k)}$ is the k-th iterate for $\mathbf{x}$ and $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ is the iteration matrix.

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

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

or in terms of the elements of $\mathbf{A}$ as

$$x_i^{(k+1)} = \frac{1}{a_{i,i}}\Big(-\sum_{j=1}^{i-1} a_{i,j}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k)} + b_i\Big), \quad i = 1, 2, \ldots, n\,. \tag{11}$$

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

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

or in terms of the elements of $\mathbf{A}$ as

$$x_i^{(k+1)} = \omega\, x_i^{\mathrm{GS}} + (1-\omega)x_i^{(k)},\ i = 1, 2, \ldots, n,$$

where x_i^{GS} is the right-hand side of (11).

BGS is the straightforward generalization of GS when matrix $\mathbf{A}$, the right-hand side and the solution vectors of (9) are partitioned in p blocks as follows:

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

The iterative step of BGS is:

$$\mathbf{x}_i^{(k+1)} = \mathbf{A}_{i,i}^{-1}\Big(-\sum_{j=1}^{i-1} \mathbf{A}_{i,j}\mathbf{x}_j^{(k+1)} - \sum_{j=i+1}^{p} \mathbf{A}_{i,j}\mathbf{x}_j^{(k)} + \mathbf{b}_i\Big), \quad i = 1, 2, \ldots, p\,. \tag{12}$$

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

2.2 An algorithm for the tuning of ω in SOR

In this section we describe an algorithm for dynamically tuning the relaxation parameter ω of SOR with the objective of reducing the number of iterations required to achieve convergence.

The algorithm is based on estimations of the convergence factor η (spectral radius of the iteration matrix $\mathbf{H}$, $\rho(\mathbf{H})$, when $\mathbf{A}$ is nonsingular, largest modulus of the eigenvalues of $\mathbf{H}$ different from 1, $\gamma(\mathbf{H})$, when $\mathbf{A}$ is singular) as well as on detecting when SOR diverges. Recall that a necessary (and sufficient when $\mathbf{A}$ is nonsingular) condition for SOR to converge is that $\eta < 1$ [5].

After each iteration k for which the last two iterations have been performed with the same value of ω, η is estimated as

$$\widetilde{\eta} = \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_\infty}{\|\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)}\|_\infty}\,.$$

Stabilization of $\widetilde{\eta}$ is monitored and it is assumed to be stabilized when the relative difference in $1/|\log\widetilde{\eta}|$ between two consecutive iterations is smaller than or equal to a given tolerance TOL_ETA three consecutive times. The rationale for using $1/|\log\widetilde{\eta}|$ instead of $\widetilde{\eta}$ is that the number of iterations required to achieve convergence is proportional to $1/|\log\eta|$. The estimator of η may take many iterations to stabilize or simply not to stabilize at all (for instance, if η corresponds to complex conjugate eigenvalues of $\mathbf{H}$). In order not to waste iterations for values of ω for which SOR diverges or $\widetilde{\eta}$ does not stabilize because it corresponds to complex eigenvalues of $\mathbf{H}$, for any ω, except $\omega = 1$, for which no limit is imposed, a maximum of $M = \max\{\text{IT\_ETA}, \text{it}_{\text{gs}}/\text{FACT\_ETA}\}$ iterations are allocated for the stabilization of $\widetilde{\eta}$, where it_{gs} is the number of iterations required for the stabilization of $\widetilde{\eta}$ for $\omega = 1$, IT_ETAis an integer value > 1 which prevents M from being too small and FACT_ETA > 1.

We have found convenient to introduce a divergence test which is also applicable when $\widetilde{\eta}$ does not stabilize. Let $\textbf{tol}^{(j)} = (|(x_i^{(j)} - x_i^{(j-1)})/x_i^{(j)}|)_{1\leq i\leq n}$. The divergence test is based on monitoring the progress of $\|\textbf{tol}^{(j)}\|_\infty$. Let i be the iteration index associated with the current value of ω (i.e. i is set to 0 when ω is changed into a new value). Every IT_TEST iterations performed with the same ω the algorithm computes

$$\Delta_k = \sum_{i=(k-1)\times\text{IT\_TEST}+1}^{k\times\text{IT\_TEST}} \|\textbf{tol}^{(i)}\|_\infty \,.$$

SOR is assumed to diverge and the iterations to estimate η for the current ω are stopped as soon as it is found that $\sum_{l=k\times\text{IT\_TEST}+1}^{i}\|\textbf{tol}^{(l)}\|_\infty > \text{DIV\_FACT}\times\Delta_k$, $k\geq 1$, $i\leq (k+1)\times\text{IT\_TEST}$, where DIV_FACT > 1. The divergence test is not used when $\omega \leq 1$, since in that case SOR is guaranteed to converge (see Section 3.1).

The algorithm tries to find an ω minimizing η in the interval $(0,\,2)$ for the linear system (2) and in the interval $[1,\,2)$ for the linear systems (4), (7) and (8) (see Section 3.1 for a justification). For the linear system (2), the algorithm gives priority to scanning to the right the interval $[1,\,2)$ because after performing many numerical experiments we have found that typically the minimum of η is in that case on the right of $\omega = 1$. The algorithm only considers values of ω for which $\widetilde{\eta}$ stabilizes to a value < 1 and it assumes that η is either a monotone function of ω or has a single local minimum in the subset of values of ω considered. If evidence is found that η as a function of ω does not satisfy any of those conditions, the tuning process is stopped and the method continues using the best explored ω (see next for details).

The algorithm performs scans to the right and scans to the left in intervals of ω. Scans to the right are performed at steps δ_ω^+ and scans to the left are performed at steps δ_ω^-. Initially, both δ_ω^+ and δ_ω^- are set to a given constant INI_DELTA, but they are divided, if necessary, by factors FACT_DELTA so that the scanning can continue within the interval for ω under exploration. The algorithm is called with a limit number of iterations and exits with failure if such a limit is exhausted and the linear system did not converge. The algorithm starts iterating with $\omega = 1$ until the estimate for η, $\widetilde{\eta}_{\text{gs}}$, stabilizes to a value < 1. If $\widetilde{\eta}_{\text{gs}}$ stabilizes to a value ≥ 1, the algorithm reverts to Gauss-Seidel. Next, while the estimate for η does not increase, the interval $(1,\,2)$ is scanned to the right. If for a given ω it is found that $\widetilde{\eta}$ does not stabilize or the method does not converge, the right limit,

r, of the search interval (initially, $r = 2$) is set to that ω and the scan continues to the right starting from the last ω for which $\widetilde{\eta}$ stabilized to a value < 1. The scan to the right of the interval $(1,\, 2)$ is stopped as soon as one of the following conditions holds:

(a) it is not possible to further reduce δ_ω^+ without falling below the desired accuracy, ϵ_ω, for the location of the optimum ω,

(b) $\widetilde{\eta}$ is found to increase.

In case (a), two situations are possible:

(a.1) $\widetilde{\eta}$ has only stabilized to a value < 1 for $\omega = 1$,

(a.2) $\widetilde{\eta}$ has already stabilized to a value < 1 for more than one value of ω; the last two values of ω for which it has happened are ω_m and ω_r, $\omega_\mathrm{m} < \omega_\mathrm{r}$, and the corresponding estimates for η are $\widetilde{\eta}_\mathrm{m}$ and $\widetilde{\eta}_\mathrm{r}$, $\widetilde{\eta}_\mathrm{m} \geq \widetilde{\eta}_\mathrm{r}$.

In case (a.1), the algorithm reverts to Gauss-Seidel if the linear system being solved is (4), (7) or (8); otherwise, the search continues in the interval $(0,\, 1)$ as will be explained later. In case (a.2), if $\omega_\mathrm{r} - \omega_\mathrm{m} > \epsilon_\omega$ the algorithm scans to the left the interval $(\omega_\mathrm{m},\, \omega_\mathrm{r})$ beginning at ω_r; otherwise, the algorithm stops tuning ω and continues using the best explored ω (i.e. the one corresponding to the smallest $\widetilde{\eta}$). Next, we discuss case (b). Three cases are possible depending again on for how many values of the relaxation parameter $\widetilde{\eta}$ has stabilized to a value < 1:

(b.1) $\widetilde{\eta}$ has only stabilized to a value < 1 for two values of ω; the two values of ω for which it has happened are $\omega = 1$ and ω_r, $\omega_\mathrm{r} > 1$; the corresponding estimates for η are $\widetilde{\eta}_\mathrm{gs}$ and $\widetilde{\eta}_\mathrm{r}$, $\widetilde{\eta}_\mathrm{r} > \widetilde{\eta}_\mathrm{gs}$,

(b.2) $\widetilde{\eta}$ has already stabilized to a value < 1 for more than two values of ω; the last three values of ω for which it has happened are ω_l, ω_m and ω_r, $\omega_\mathrm{l} < \omega_\mathrm{m} < \omega_\mathrm{r}$, and the corresponding estimates for η are $\widetilde{\eta}_\mathrm{l}$, $\widetilde{\eta}_\mathrm{m}$ and $\widetilde{\eta}_\mathrm{r}$, $\widetilde{\eta}_\mathrm{l} > \widetilde{\eta}_\mathrm{m} < \widetilde{\eta}_\mathrm{r}$,

(b.3) as (b.2) but with $\widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{m}$.

In case (b.1), the search continues in the interval $(0,\, 1)$ if the linear system being solved is (2). When the linear system being solved is (4), (7) or (8), if $\omega_\mathrm{r} - 1 > \epsilon_\omega$ the algorithm scans to the right the interval $(1,\, \omega_\mathrm{r})$ beginning at 1; otherwise, the algorithm reverts to Gauss-Seidel. In case (b.2), a minimum of η has been bracketed. If $\omega_\mathrm{m} - \omega_\mathrm{l} > \epsilon_\omega$ or $\omega_\mathrm{r} - \omega_\mathrm{m} > \epsilon_\omega$, the golden section search method (see, for instance, [20]) is used in the interval $(\omega_\mathrm{l},\, \omega_\mathrm{r})$ to find such a minimum; otherwise, the algorithm stops tuning ω and continues with the best explored ω. In case (b.3), if $\omega_\mathrm{m} - \omega_\mathrm{l} > \epsilon_\omega$ the interval $(\omega_\mathrm{l},\, \omega_\mathrm{m})$ is scanned to the left beginning at ω_m; otherwise, the algorithm stops tuning ω and continues with the best explored ω.

To solve (2), the search in the interval $(0,\,1)$ is performed in a similar way. While $\widetilde{\eta}$ decreases, the algorithm makes a scan to the left. If for a given ω it is found that $\widetilde{\eta}$ does not stabilize or the method does not converge, the left limit, l, of the search interval (initially, $l = 0$) is set to that ω and the scan continues to the left starting from the last ω for which $\widetilde{\eta}$ stabilized to a value < 1. The scan to the left of the interval $(0,\,1)$ is stopped as soon as one of the following conditions holds:

(c) δ_ω^- cannot be reduced without falling below ϵ_ω,

(d) $\widetilde{\eta}$ is found not to decrease.

In case (c), two cases are possible:

(c.1) $\widetilde{\eta}$ has only stabilized to a value < 1 for $\omega = 1$,

(c.2) $\widetilde{\eta}$ has already stabilized to a value < 1 for more than one value of ω; the last two values of ω for which it has happened are ω_l and ω_m, $\omega_\mathrm{l} < \omega_\mathrm{m}$, and the corresponding estimates for η are $\widetilde{\eta}_\mathrm{l}$ and $\widetilde{\eta}_\mathrm{m}$, $\widetilde{\eta}_\mathrm{l} < \widetilde{\eta}_\mathrm{m}$.

In case (c.1), the algorithm reverts to Gauss-Seidel. In case (c.2), if $\omega_\mathrm{m} - \omega_\mathrm{l} > \epsilon_\omega$ the algorithm scans to the right the interval $(\omega_\mathrm{l},\,\omega_\mathrm{m})$ beginning at ω_l; otherwise, the algorithm stops tuning ω and continues with the best explored ω. Finally, in case (d) three cases are possible:

(d.1) $\widetilde{\eta}$ has only stabilized to a value < 1 for $\omega = 1$ and $\omega_\mathrm{l} < 1$; the corresponding estimates for η are $\widetilde{\eta}_\mathrm{gs}$ and $\widetilde{\eta}_\mathrm{l}$, $\widetilde{\eta}_\mathrm{l} \geq \widetilde{\eta}_\mathrm{gs}$,

(d.2) the same as (b.2),

(d.3) the same as (b.3).

In case (d.1), if $1 - \omega_\mathrm{l} > \epsilon_\omega$ the algorithm scans to the left the interval $(\omega_\mathrm{l},\,1)$ beginning at 1; otherwise, the algorithm reverts to Gauss-Seidel. Cases (d.2) and (d.3) are dealt with as cases (b.2) and (b.3), respectively. The previous description of the algorithm for tuning ω is rather informal. An automaton-based formal description is given in Appendix A.

Selection of appropriate values for the parameters on which the algorithm to dynamically tune ω depends is not trivial, being from our experience TOL_ETA, IT_ETA and FACT_ETA the most delicate ones. If TOL_ETA is chosen too large an erroneous estimate of the convergence factor may result and the tuning process may become confused. If TOL_ETA is chosen too small and $\widetilde{\eta}$ takes a large number of iterations to stabilize the algorithm may not explore all values of ω of interest. Selection of values for IT_ETA and FACT_ETA also involves a tradeoff. If the resulting M is too small, the algorithm may not explore all values of ω of interest. If the resulting M is too large, iterations may be wasted for an ω for which $\widetilde{\eta}$ does not stabilize (if, for instance, η corresponds to complex conjugate eigenvalues of the iteration matrix). After performing many experiments, we have found IT_ETA $= 150$, FACT_ETA $= 2$, TOL_ETA $= 0.001$, $\epsilon_\omega = 0.001$, INI_DELTA $= 0.1$, FACT_DELTA $= 10$, IT_TEST $= 30$, and DIV_FACT $= 1.5$ to be appropriate choices.

2.3 Generalized Minimal Residual

The GMRES method begins with an initial approximate solution $\mathbf{x}^{(0)}$ and an initial residue $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$, and generates an approximate solution at step j as $\mathbf{x}^{(j)} = \mathbf{x}^{(0)} + \mathbf{z}^{(j)}$. The vector $\mathbf{z}^{(j)}$ is the vector in $\mathcal{K}_j$ which minimizes $\|\mathbf{b} - \mathbf{A}\mathbf{x}^{(j)}\|_2 = \|\mathbf{r}^{(0)} - \mathbf{A}\mathbf{z}^{(j)}\|_2$, where $\mathcal{K}_j$ is the j-dimensional Krylov subspace generated by $\mathbf{A}$ and $\mathbf{r}^{(0)}$

$$\mathcal{K}_j = \text{span}\{\mathbf{r}^{(0)},\ \mathbf{A}\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{j-1}\mathbf{r}^{(0)}\}\,.$$

The least-squares problem is solved in such a way that for each j the norm of the residue $\|\mathbf{r}^{(j)}\|_2$ is available and convergence is achieved when $\|\mathbf{r}^{(j)}\|_2 \leq \delta_r$, where δ_r is a predefined small enough value.

The memory and time requirements of GMRES grow as j increases because the method needs the j vectors $\mathbf{r}^{(0)},\ \mathbf{A}\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{j-1}\mathbf{r}^{(0)}$ to construct an orthonormal basis of $\mathcal{K}_j$. Thus, in practice a *restarted* version, GMRES(k), is used: after every k iterations (assuming convergence has not been achieved) the algorithm is restarted, taking $\mathbf{x}^{(k)}$ as the next initial solution guess for the next cycle of k iterations. For details about GMRES and GMRES(k), see [22].

Convergence of GMRES is monotonic, i.e. $\|\mathbf{r}^{(j+1)}\|_2 \leq \|\mathbf{r}^{(j)}\|_2$. Furthermore, in exact arithmetic it reaches the exact solution in at most n steps if $\mathbf{A}$ is nonsingular [22]. However, if k is not large enough, GMRES(k) can converge very slowly or even *stagnate*, i.e. the reduction in the residual norm after each step tends to zero and the algorithm does not reach the solution. The convergence rate of the method typically increases with k but so does the memory and time requirements per iteration. Thus, the issue of selecting an appropriate value for k arises. In [23] an adaptive variant of GMRES(k) is proposed in which k is enlarged or maintained depending on how fast the residual norm decreases. The algorithm starts with $k = k_0$. After each cycle, if the 2-norm of the current residue $\mathbf{r}^{(j)}$ is larger than δ_r, an estimate of the number of iterations still needed to reach convergence is computed using the residue at the beginning of the recently completed cycle, $\mathbf{r}^{(j-k)}$, as $\xi = k\ \log(\delta_r/\|\mathbf{r}^{(j)}\|_2)/\log(\|\mathbf{r}^{(j)}\|_2/\|\mathbf{r}^{(j-k)}\|_2)$. Being $j_{\max}$ the iterations limit and sv a small number, the algorithm is assumed to be *near-stagnated* if $\xi \geq sv \times (j_{\max} - j)$. In that case, if k has not reached yet its maximum value $k_{\max}$, k is incremented by some value m and the cycle is continued till complete the new number k of iterations or achieve convergence. If k is not enlarged, first the 2-norm of $\mathbf{r}^{(j)}$ is checked to be non greater than that of $\mathbf{r}^{(j-k)}$ (it could be greater due to numerical instability of the method), aborting the procedure otherwise. Next, if $\xi \geq bv \times (j_{\max} - j)$, where bv is typically much larger than sv, stagnation is assumed and the whole algorithm aborted because it is unlikely that the algorithm will achieve convergence within the remaining $j_{\max} - j$ iterations. Notice that this last test subsumes the case in which the maximum number of iterations $j_{\max}$ is reached. Our algorithm (GMR), which is described in Fig. 1, closely follows that of [23]. The main difference is that the Krylov subspace basis is orthonormalized using the modified Gram-Schmidt method with double orthogonalization (MGO) [21] instead of the Householder reflection procedure (HO). MGO performs as the modified Gram-Schmidt procedure (see, for instance, [24]) but the norm of the new basis vector, $\mathbf{w}_l$, which is being computed is monitored to reduce the impact of cancellations: at each step, $h_{i,l} = \mathbf{w}_l^{\mathrm{T}}\mathbf{v}_i$ and $\mathbf{w}_l = \mathbf{w}_l - h_{i,l}\mathbf{v}_i$ are computed; since $\|\mathbf{v}_i\|_2 = 1$,

$s = \|\mathbf{w}_l\|_2^2$ will be reduced by $h_{i,l}^2$. If $h_{i,l}^2$ is greater than, say $0.99 \times s$, cancellations might be important and a second orthogonalization step is performed. HO is numerically more stable than MGO but it is about twice as expensive as MGO when the number of reorthogonalizations is small [21, 23]. Also, MGO is known to be appropriate for most applications [21].

Preconditioning techniques can significantly speed up GMRES(k) [21, 23, 24]. We use right-preconditioning with preconditioner $\mathbf{G}$, i.e. we transform the original problem (9) into the new one

$$\mathbf{A}\mathbf{G}^{-1}\mathbf{u} = \mathbf{b}\,, \qquad \mathbf{u} = \mathbf{G}\mathbf{x}\,.$$

This implies that the vector $\mathbf{w}_l$ which the MGO procedure starts with in Fig. 1 is now $\mathbf{w}_l = \mathbf{A}\mathbf{G}^{-1}\mathbf{v}_l$, and that the solution after each restart cycle, $\mathbf{x}$, is computed as $\mathbf{x} = \mathbf{x} + \mathbf{G}^{-1}\mathbf{V}_l\mathbf{y}$. We use the symmetric Gauss-Seidel preconditioner: $\mathbf{G} = (\mathbf{D}-\mathbf{E})\mathbf{D}^{-1}(\mathbf{D}-\mathbf{F})$, where $\mathbf{D} = [\mathrm{diag}(\mathbf{A})]$ and $-\mathbf{E}$ and $-\mathbf{F}$ are, respectively, the strict lower and upper part of $\mathbf{A}$. In practice, the matrix $\mathbf{G}^{-1}$ is not actually computed (this would be too expensive due to fill-in) and the required $\mathbf{z}_2 = \mathbf{G}^{-1}\mathbf{z}_0$ computations are carried out by solving first the lower triangular linear system $(\mathbf{I} - \mathbf{E}\mathbf{D}^{-1})\mathbf{z}_1 = \mathbf{z}_0$ and then the upper triangular linear system $(\mathbf{D} - \mathbf{F})\mathbf{z}_2 = \mathbf{z}_1$. Although there are more elaborate preconditioners which usually work better [19], we have found that the improvement achieved by symmetric Gauss-Seidel is enough. Moreover, it does not need any extra storage since $\mathbf{G}$ is "contained" into the coefficient matrix $\mathbf{A}$ of (9).

The most critical parameters of our GMR algorithm are k_0 and k_{max}. The larger they are, the more likely the algorithm will achieve convergence within the maximum number of allowed iterations. On the other hand, GMR requires as many extra arrays of size n as the dimension of the Krylov subspace $k \in \{k_0, \ldots, k_{\text{max}}\}$ the algorithm chooses, so neither k_0 nor k_{max} can be too large if memory is a concern. After some experimentation we have found $k_0 = 20$, $k_{\text{max}} = 30$, $m = 2$, $sv = 0.005$, and $bv = 1$ to be appropriate choices.

3 Convergence

3.1 Convergence results

First we analyze convergence for the splitting-based methods. A matrix $\mathbf{C}$ is said to be semiconvergent if $\lim_{k\to\infty} \mathbf{C}^k$ exists. In particular, $\mathbf{C}$ is called convergent if that limit equals $\mathbf{0}$. The splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$, $\mathbf{M}$ nonsingular, on which the iterative method defined by (10) is based is semiconvergent if $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ is semiconvergent. If the linear system (9) is consistent, the iterative method defined by (10) converges to a solution of (9) for each $\mathbf{x}^{(0)}$ if and only if the splitting from which (10) has been derived is semiconvergent [5, Chapter 7, Lemma 6.13]. Obviously, when $\mathbf{A}$ is singular the solution obtained by the iterative method will depend on $\mathbf{x}^{(0)}$. A necessary and sufficient condition for the nonsingular matrix $\mathbf{C}$ to be convergent is $\rho(\mathbf{C}) < 1$, while in the singular case $\mathbf{C}$ is semiconvergent (see, for instance, [18]) if and only if: (1) $\rho(\mathbf{C}) \leq 1$, and, if $\rho(\mathbf{C}) = 1$, then (2) 1 is an eigenvalue of $\mathbf{C}$ and $\gamma(\mathbf{C}) < 1$, and (3) $\mathbf{C}$ has only linear elementary divisors corresponding

```
    set $x$, $j_{\max}$, $k_0$, $k_{\max}$, $m$, $\delta_r$, $sv$, $bv$;
    $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$; $k = k_0$; $j = 0$;
    **while** $\|\mathbf{r}\|_2 > \delta_r$ **do**
        $\mathbf{r}^{\text{old}} = \mathbf{r}$;
        $\mathbf{v}_1 = \mathbf{r}/\|\mathbf{r}\|_2$;
        **for** $l = 1$ **until** $k$ **do**
A:          $j = j + 1$;
            $\mathbf{w}_l = \mathbf{A}\mathbf{v}_l$;
            $s = \|\mathbf{w}_l\|_2^2$;
            **for** $i = 1$ **until** $l$ **do**      /* MGO algorithm */
                $h_{i,l} = \mathbf{w}_l^{\mathrm{T}}\mathbf{v}_i$;
                $\mathbf{w}_l = \mathbf{w}_l - h_{i,l}\mathbf{v}_i$;
                **if** $h_{i,l}^2 > 0.99s$ **then**      /* second orthogonalization */
                    $h' = \mathbf{w}_l^{\mathrm{T}}\mathbf{v}_i$; $h_{i,l} = h_{i,l} + h'$; $\mathbf{w}_l = \mathbf{w}_l - h'\mathbf{v}_i$;
                **end if**
                $s = s - h_{i,l}^2$;
            **end for**
            $h_{l+1,l} = \|\mathbf{w}_l\|_2$;
            **if** $h_{l+1,l} = 0$ **then** go to B **else** $\mathbf{v}_{l+1} = \mathbf{w}_l/h_{l+1,l}$ **end if**
            update $\|\mathbf{r}\|_2$ as in [21];
            **if** $\|\mathbf{r}\|_2 \leq \delta_r$ **then** go to B **end if**
        **end for**
        $\xi = k\log\big(\delta_r/\|\mathbf{r}\|_2\big)/\log\big(\|\mathbf{r}\|_2/\|\mathbf{r}^{\text{old}}\|_2\big)$;
        **if** $\xi \geq sv(j_{\max} - j)$ **and** $k \leq k_{\max} - m$ **then**
            $k = k + m$;
            go to A;
        **end if**
B:      solve $\min_y \big\| \|\mathbf{r}\|_2\, \mathbf{e}_1 - \overline{\mathbf{H}}\mathbf{y} \big\|_2$, where $\mathbf{e}_1 = (1, 0, \ldots, 0)^{\mathrm{T}}$ and $\overline{\mathbf{H}}$ is
        an $(l+1) \times l$ matrix described in [21];
        $\mathbf{x} = \mathbf{x} + \mathbf{V}_l\mathbf{y}$, where $\mathbf{V}_l = [\mathbf{v}_1 \ldots \mathbf{v}_l]$; $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$;
        **if** $\|\mathbf{r}\|_2 \leq \delta_r$ **then** exit **else if** $\|\mathbf{r}^{\text{old}}\|_2 < \|\mathbf{r}\|_2$ **then** abort **end if**
        $\xi = k\log\big(\delta_r/\|\mathbf{r}\|_2\big)/\log\big(\|\mathbf{r}\|_2/\|\mathbf{r}^{\text{old}}\|_2\big)$;
        **if** $\xi \geq bv(j_{\max} - j)$ **then** abort **end if**
    **end while**
```

Figure 1: Algorithm GMR.

to the eigenvalue 1 (i.e. all the Jordan blocks associated with 1 in the Jordan canonical form of $\mathbf{C}$ have dimension 1). $\mathbf{P}$ is a singular M-matrix and both $\mathbf{P}_{UU}$ and $\mathbf{P}_{SS}$ are nonsingular M-matrices [24] having all them nonvanishing diagonal elements. We have the following well-known results:

1. The SOR method can only converge for the linear system (2) for $0 < \omega < 2$ [27, Ch. 4, Theorem 1.2]. Moreover, the splitting on which SOR is based is semiconvergent for $0 < \omega < 1$ [2, Corollary 3].

2. For the linear systems systems (4), (7) and (8) SOR can only converge for $0 < \omega < 2$ [5, Ch. 7, Theorem 4.5]. The splittings from which GS and SOR with $0 < \omega \leq 1$ are derived are convergent [24, Theorems 3.6, 3.7]. In addition, η is a nonincreasing function of ω in the range $0 < \omega \leq 1$ [5, Ch. 7, Theorem 5.23]

Using the fact that $\det(\mathbf{H}_{\mathrm{SOR}}) = (1-\omega)^n$, where $\mathbf{H}_{\mathrm{SOR}}$ is the iteration matrix for the SOR method and n is the dimension of the linear system being solved, it is justified in [27, Ch. 4, Theorem 1.2] that SOR diverges for the singular system (2) if $\omega < 0$ or $\omega > 2$. Here, we discuss the divergence of the method for the cases $\omega = 0$ and $\omega = 2$. The case $\omega = 0$ is immediate since the splitting which gives place to the SOR method is not defined. In the case $\omega = 2$, $|\det(\mathbf{H}_{\mathrm{SOR}})| = 1$, which does not prevent 1 from being the only eigenvalue of $\mathbf{H}_{\mathrm{SOR}}$, thus fulfilling convergence conditions (1) and (2). Note that if $\mathbf{H}_{\mathrm{SOR}}$ has an unique eigenvalue, condition (2) states that the Jordan canonical form of $\mathbf{H}_{\mathrm{SOR}}$ must have n Jordan blocks. The number of Jordan blocks having the eigenvalue λ in the Jordan canonical form of an $n \times n$ matrix $\mathbf{C}$ is equal to $\dim(\mathrm{Ker}(\mathbf{C} - \lambda\mathbf{I}))$ (see, for instance, [1]), where $\mathrm{Ker}(\mathbf{C} - \lambda\mathbf{I})$ is the null space of $\mathbf{C} - \lambda\mathbf{I}$, and, trivially, $\dim(\mathrm{Ker}(\mathbf{C} - \lambda\mathbf{I})) < n$ unless $\mathbf{C} - \lambda\mathbf{I} = \mathbf{0}$. It is easy to verify that the (1, 1)-element of $\mathbf{H}_{\mathrm{SOR}} - \mathbf{I}$ is equal to -2 when $\omega = 2$. Therefore, $\mathbf{H}_{\mathrm{SOR}} - \mathbf{I} \neq \mathbf{0}$ and, then, $\dim(\mathrm{Ker}(\mathbf{H}_{\mathrm{SOR}} - \mathbf{I})) < n$, implying that the canonical Jordan form of $\mathbf{H}_{\mathrm{SOR}}$ has at most $n-1$ Jordan blocks. Thus, condition (3) above is violated and the SOR method diverges for $\omega = 2$.

Let $\mathbf{H}_{\mathrm{GS}}$ be the iteration matrix of GS. Since $\mathbf{P}$ has "property c" [24, Theorem 3.16], it follows [18, Theorem 5] that $\mathbf{H}_{\mathrm{GS}}$ is semiconvergent if and only if 1 is the only eigenvalue of $\mathbf{H}_{\mathrm{GS}}$ on the unit circle, i.e. $\gamma(\mathbf{H}_{\mathrm{GS}}) < 1$. The directed graph $\Gamma(\mathbf{C}) = (V, E)$ associated with the $n \times n$ matrix $\mathbf{C} = (c_{i,j})_{1 \leq i,j \leq n}$ is defined by the set of vertices $V = \{1, \ldots, n\}$ and the set of edges $E = \{(i,j) \in V | \, c_{i,j} \neq 0\}$. A sequence of vertices $(i_0, \ldots, i_{l-1}, i_l)$ such that $(i_k, i_{k+1}) \in E$, $0 \leq k < l$ is called a path. If $i_l = i_0$ and $i_0, \ldots, i_{l-1}$ are distinct, the sequence is called a cycle. A path $(i_0, \ldots, i_l)$ is monotone increasing if $i_0 < \cdots < i_l$ and monotone decreasing if $i_0 > \cdots > i_l$. Similarly, a cycle $(i_0, \ldots, i_{l-1}, i_0)$ is monotone increasing if the path $(i_0, \ldots, i_{l-1})$ is monotone increasing and monotone decreasing if the path $(i_0, \ldots, i_{l-1})$ is monotone decreasing [2]. A necessary and sufficient condition on $\Gamma(\mathbf{P}) = \Gamma(\mathbf{Q}^{\mathrm{T}})$ for $\gamma(\mathbf{H}_{\mathrm{GS}})$ to be < 1 is given in [2, Theorem 1]. Unfortunately, the result requires the knowledge of all the cycles in $\Gamma(\mathbf{P})$. Several, more practical sufficient conditions on $\Gamma(\mathbf{P})$ or $\Gamma(\mathbf{Q})$ for $\gamma(\mathbf{H}_{\mathrm{GS}})$ to be < 1 have been derived [2, 17]. The result derived in [2, Corollary 1] states that if $\Gamma(\mathbf{P})$ has a monotone decreasing cycle, then forward Gauss-Seidel[1] converges for each initial guess. Let i_f be the highest index in the

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

ordering of the states of the CTMC. The result proven in [17, Theorem 5.2] states that if for each $i_0 \in V$ there exists a monotone increasing path $(i_0, \ldots, i_f)$, then forward Gauss-Seidel converges for each initial guess. Next, we give another sufficient condition for Gauss-Seidel to converge for each initial guess.

Theorem 1. *Let* $\mathbf{Q}$ *be the infinitesimal generator of a finite and irreducible CTMC and let* $\Gamma(\mathbf{Q}) = (V, E)$ *be the directed graph associated with* $\mathbf{Q}$*. If the ordering of the states of the CTMC is such that for any state with index* $i > 1$ *there exists another state with index* $j < i$ *such that* $(j,\, i) \in E$*, then forward Gauss-Seidel converges for the linear system* $\mathbf{P}\boldsymbol{\phi} = \mathbf{0}$ *for each initial guess* $\boldsymbol{\phi}^{(0)}$*.*

Proof. Let $i_0 \neq 1$ be the index of any state such that $(i_0,\, 1) \in E$. Because of irreducibility of $\mathbf{Q}$, some state with index i_0 exists. Let $i_1 < i_0$ be the index of some state such that $(i_1,\, i_0) \in E$. By assumption, some state with index i_1 satisfying the condition exists. i_1 may be 1 or have a value > 1. If $i_1 = 1$, we have finished. If $i_1 > 1$, we can consider a state with index $i_2 < i_1$ such that $(i_2,\, i_1) \in E$, which by assumption must exist. Iterating the reasoning it is clear that a cycle $(i_0,\, 1,\, i_k,\, \ldots,\, i_2,\, i_1,\, i_0)$, $k \geq 0$ with $i_l < i_{l-1}$, $1 \leq l \leq k$ can be formed in $\Gamma(\mathbf{Q})$. That cycle becomes $(i_0,\, i_1,\, i_2,\, \ldots,\, i_k,\, 1,\, i_0)$ in $\Gamma(\mathbf{Q}^{\mathrm{T}})$ and, since $\Gamma(\mathbf{P}) = \Gamma(\mathbf{Q}^{\mathrm{T}})$, in $\Gamma(\mathbf{P})$, which is monotone decreasing. Then [2, Corollary 1], forward Gauss-Seidel converges to solve $\mathbf{P}\boldsymbol{\phi} = \mathbf{0}$ for each $\boldsymbol{\phi}^{(0)}$. $\square$

The assumption on the ordering of the states of Theorem 1 is not very restrictive in practice, since it encompasses the very common situation in which the CTMC is generated from a given start state using a set of generation rules and states are numbered increasingly as they are generated.

Two issues must be considered for the algorithm GMR: convergence and breakdown. As it has been stated in Section 2.3, convergence of GMRES(k) can be very slow or even the algorithm can stagnate and never reach the solution, and the same applies to our variant, GMR. Breakdown is related to the least-squares problem which has to be solved at the end of each cycle. That problem can be formulated as the minimization of the functional

$$J(\mathbf{y}) = \|\mathbf{r}^{(0)} - \mathbf{A}\mathbf{V}_l\, \mathbf{y}\|_2\,, \tag{13}$$

where $\mathbf{V}_l = [\mathbf{v}_1 \ldots \mathbf{v}_l]$ (see Fig. 1) contains an orthonormal basis of $\mathcal{K}_l$. Breakdown occurs when $\mathrm{rank}(\mathbf{A}\mathbf{V}_l) < l$ and, consequently, (13) has not an unique solution. At some step l, GMRES (and GMR) can either (a) break down through rank deficiency of the least-squares problem ($\mathrm{rank}(\mathbf{A}\mathbf{V}_l) < \mathrm{rank}(\mathbf{V}_l)$) without determining a solution, or (b) determine a solution without breakdown and then break down at the next step through degeneracy of $\mathcal{K}_{l+1}$ ($\mathrm{rank}(\mathbf{V}_{l+1}) < l$) [7, Theorem 2.2]. Matrices $\mathbf{P}_{UU}$ and $\mathbf{P}_{SS}$ of the linear systems (4), (7) and (8) are nonsingular. $\mathbf{A} = \mathbf{P}$ of the linear system (2) is singular, the system is consistent because the CTMC X is finite and irreducible, and, since $\mathrm{index}(\mathbf{P}) = 1$[2] [24], $\mathrm{Ker}(\mathbf{P}) \bigcap \mathrm{Im}(\mathbf{P}) = \mathbf{0}$ (see, for instance, [11]). Therefore, for the linear systems (2), (4), (7), and (8) only case (b) above is possible. Note that

[2]The index of a square matrix $\mathbf{A}$ is defined as the lowest nonnegative integer k such that $\mathbf{A}^k$ and $\mathbf{A}^{k+1}$ have the same rank [5].

$\mathrm{rank}(\mathbf{V}_{l+1}) < l$ is equivalent to $\mathbf{v}_{l+1}$ to be linearly dependent of $\mathbf{v}_i$, $i = 1, \ldots, l$, i.e. to have $h_{l,l+1} = 0$ at step l of GMR. Therefore, for the linear systems being considered the algorithm GMR is safe in the sense that it can only break down if $h_{l,l+1} = 0$ and in that case the solution reached is exact.

We conclude this section with two remarks regarding the GMR algorithm. The first one has to do with the initial approximation $\mathbf{x}^{(0)}$ used. If the coefficient matrix $\mathbf{A}$ is nonsingular (4), (7), (8), it does not matter which initial iteration vector is taken. However, if $\mathrm{index}(\mathbf{A}) = 1$ and $\mathbf{b} = \mathbf{0}$ (2), $\mathbf{x}^{(0)}$ must not belong to $\mathrm{Im}(\mathbf{A})$, since otherwise the iterates will converge to the trivial solution $\mathbf{x} = \mathbf{0}$ [11, Corollary 3.2]. The following theorem gives a criterium to choose an appropriate $\mathbf{x}^{(0)}$.

Theorem 2. *Let* $\mathbf{Q}$ *be the infinitesimal generator of a finite and irreducible CTMC and let* $\mathbf{P} = \mathbf{Q}^{\mathrm{T}}[\mathrm{diag}(\mathbf{Q})]^{-1}$. *A sufficient condition for the solution of the linear system (2) using GMRES or GMR not to converge to the trivial solution* $\mathbf{x} = \mathbf{0}$ *is to take an initial iteration vector* $\mathbf{x}^{(0)}$ *with* $\sum_{i=1}^{n} x_i^{(0)} \neq 0$.

Proof. Assume a vector $\mathbf{x} \in \mathrm{Im}(\mathbf{P})$. This implies the existence of a vector $\mathbf{y}$ such that $\mathbf{P}\mathbf{y} = \mathbf{x}$. Since the rows of $\mathbf{Q}$ add up 0, we have

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} y_j = \sum_{j=1}^{n} y_j \sum_{i=1}^{n} p_{ij} = \sum_{j=1}^{n} y_j \sum_{i=1}^{n} \frac{q_{ji}}{q_{jj}} = \sum_{j=1}^{n} y_j \frac{1}{q_{jj}} \sum_{i=1}^{n} q_{ji} = 0\,.$$

Therefore, $\sum_{1 \le i \le n} x_i^{(0)} \neq 0$ is a sufficient condition for $\mathbf{x}^{(0)} \notin \mathrm{Im}(\mathbf{P})$. Since (2) is consistent, this implies [11, Corollary 3.2] that neither GMRES nor GMR converge to the trivial solution $\mathbf{0}$. $\square$

Our last remark has to do with the preconditioner. If the system (9) is transformed into the new one

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}\mathbf{u} = \mathbf{L}^{-1}\mathbf{b}\,, \qquad \mathbf{x} = \mathbf{U}^{-1}\mathbf{u}\,,$$

by means of a nonsingular preconditioner matrix $\mathbf{G} = \mathbf{L}\mathbf{U}$, $\mathrm{index}(\mathbf{A}) = 1$ implies $\mathrm{index}(\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}) = 1$, so the above results still apply.

3.2 Test of convergence

As convergence test we require the relative variation on the computed measure (SSRR or MCRTF) 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 and avoids false convergence if the iteration vectors oscillate.

The convergence test described above is easily implemented for the Gauss-Seidel, SOR and block Gauss-Seidel methods. Implementation of the convergence test for GMR requires some discussion. The natural criterion for checking convergence in GMR is to use the 2-norm of the residue. If $\widetilde{\mathbf{x}}$ is the computed solution of the linear system (9), when $\mathbf{A}$ is nonsingular and $\mathbf{b} \neq \mathbf{0}$ the 2-norm of the residue $\mathbf{r} = \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}$ is related to the 2-norm of the error vector $\mathbf{\Delta x} = \widetilde{\mathbf{x}} - \mathbf{x}$ through the condition number of $\mathbf{A}$, $\kappa(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2$, by

$$\frac{\|\mathbf{\Delta x}\|_2}{\|\mathbf{x}\|_2} \leq \kappa(\mathbf{A}) \frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \, .$$

When $\mathbf{A}$ is a singular $n \times n$ M-matrix of rank $n-1$, as it is $\mathbf{P}$ [24], one possible bound for $\|\mathbf{\Delta x}\|_2$ is [3, Theorem 2.1]

$$\|\mathbf{\Delta x}\|_2 \leq |\epsilon| + \frac{\left(1+\sqrt{n}\right)\|\mathbf{r}\|_2}{\sigma_{n-1}} \, ,$$

where $\epsilon = \|\mathbf{\Delta x}\|_1 - 1$ and σ_{n-1} is the smallest positive singular value of $\mathbf{A}$. However, neither $\kappa(\mathbf{A})$ nor σ_{n-1} are known, so, in practice, from the knowledge of $\|\mathbf{r}\|_2$ we cannot estimate the accuracy of the solution. Moreover, if $\kappa(\mathbf{A})$ or $1/\sigma_{n-1}$ are large, the actual error can be large even though $\|\mathbf{r}\|_2$ is small. Therefore, we proceed as follows. Given ϵ and a reduction factor ϵ_r for the residual norm, we run GMR with $\delta_r = \epsilon_r \|\mathbf{r}^{(0)}\|_2$ until it reaches convergence. Then, three more iterations are performed computing the explicit solution for each of them (i.e. with $k_0 = k_{\max} = 1$) and we check the relative variation on the computed measure. If the convergence test is satisfied, the algorithm finishes. Otherwise, ϵ_r is divided by a given factor $f > 1$ and adaptive GMR starts again taking the last computed solution vector as initial iteration vector. We have found $\epsilon_r = \epsilon$ and $f = 10$ to be appropriate choices.

4 Numerical results

In this section we compare the performance of the numerical methods described in Section 2 using examples representing five scenarios:

1. solution of (2) for a model with failure and repair transitions and immediate detection of component failures,

2. solution of (4) and (7) for a model with failure and repair transitions and immediate detection of component failures,

3. solution of (2) for a model with failure and repair transitions and K-Erlang intertest time of spare components,

4. solution of (4) and (7) for a model with failure and repair transitions and K-Erlang intertest time of spare components,

5. solution of (2) for a model with failure, repair and performance transitions and immediate detection of component failures.

For all methods, the relative tolerance for convergence is taken $\epsilon = 10^{-8}$ and a maximum of 100 000 iterations is allowed. As initial guess $\mathbf{x}^{(0)}$ we take $\mathbf{x}^{(0)} = (1/n, \ldots, 1/n)$ to solve (2) and $\mathbf{x}^{(0)} = (1, \ldots, 1)$ to solve both (4) and (7). CPU times have been all measured on a 128 MB, 167

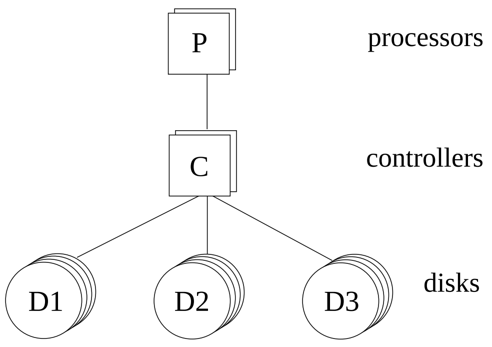


Figure 2: Distributed fault-tolerant database system.

MHz ULTRA 1 SPARC workstation. Main memory usage was in all cases smaller than the available one.

The first example (corresponding to scenario 1) is the distributed fault-tolerant database system depicted in Fig. 2. 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. The dormancy factor for the spare units is 0.2 (i.e. spare components fail with rate 0.2 times the failure rate of active components). 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. Repairs of processors in hard failure mode, controllers and disks are performed by 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 $25\,250$ 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, SOR and GMR. The CPU time required for the generation of the model was 0.242 s. We give in Table 1 the number of iterations, CPU time in seconds and UA for the first example. The GS method is the fastest one. SOR requires the same number of iterations to achieve convergence as GS because the convergence is so fast that it is achieved before any tuning on ω can be done. The time per iteration for SOR is slightly greater than it is for GS. GMR requires the smallest number of iterations but is the most expensive in time. This is because the number of floating-point operations per iteration of GMR is substantially greater than the number of floating-

Table 1: Number of iterations (top), CPU time in s (bottom) and UA for the first example and several values of the coverage probability C.

	method			
C	GS	SOR	GMR	UA
0.9	20	20	10	4.054×10^{-5}
	(0.104)	(0.116)	(0.193)	
0.99	19	19	12	4.461×10^{-6}
	(0.103)	(0.121)	(0.236)	
0.999	19	19	14	8.537×10^{-7}
	(9.66×10^{-2})	(0.122)	(0.281)	
0.9999	20	20	14	4.929×10^{-7}
	(0.109)	(0.133)	(0.287)	

point operations per iteration of both GS and SOR.

The second example, corresponding to scenario 2, is identical to the first one except that the number of disk clusters is increased to six. The measure of interest is the MTTF, a particular case of the MCRTF measure. The state in which all components are unfailed has initial probability equal to 1. The number of transient states of the CTMC X^U is 384 and the number of transitions among the states of U is 2884. The generation time of X^U was 5.82×10^{-2} s. We consider the numerical methods GS, SOR, GMR, AGS, and ASOR. We give in Table 2 the number of iterations, CPU time in seconds and MTTF for the second example. We consider the same values for the coverage probability C as we did for the first example. If the acceleration technique described in Section 1 is not used, the GMR method requires by far the smaller CPU time. Also, its performance is almost independent of the coverage probability. The GS and SOR methods do not perform satisfactorily, as it was expected [15]. Notice that the larger C, the greater the number of iterations required by GS and SOR. This is because the rate associated to the uncovered failures of the controllers approaches zero and system failure becomes a rarer event. The behavior is in accordance with the theory given in [15]. SOR performs substantially better than GS, and its performance does not degrade so sharply as C increases. When the acceleration technique is used, the AGS method is the fastest. Both AGS and ASOR appear to be insensitive to the value of the coverage parameter C. ASOR requires the same number of iterations than AGS because the convergence is so fast that ASOR has not left the value $\omega = 1$. However, ASOR is slightly slower than AGS because each iteration step of SOR is slightly more expensive than each iteration step of GS.

The system considered in the third example, corresponding to scenario 3, is exactly the same as the system of the first example, with the only difference that failures in spare processors and controllers are not immediately detected. These components are tested with deterministic intertest time T approximated by a K-Erlang distribution with expected value T and K large enough to make the approximation error small. The measure of interest is UA, a particular case of SSRR. It

Table 2: Number of iterations (top) and CPU time in s (bottom) for the second example and several values of the coverage probability C.

	method					
C	GS	SOR	GMR	AGS	ASOR	4.964×10^4
0.9	804	153	8	13	13	
	(0.488)	(0.132)	(1.95×10^{-2})	(8.26×10^{-3})	(1.16×10^{-2})	
0.99	4743	385	8	13	13	4.629×10^5
	(2.88)	(0.330)	(2.02×10^{-2})	(8.41×10^{-3})	(1.16×10^{-2})	
0.999	28 163	1340	8	13	13	2.763×10^6
	(17.1)	(1.13)	(2.04×10^{-2})	(8.21×10^{-3})	(1.16×10^{-2})	
0.9999	52 694	3905	9	13	13	5.492×10^6
	(32.1)	(3.18)	(2.22×10^{-2})	(8.24×10^{-3})	(1.16×10^{-2})	

Table 3: Number of required Erlang stages K, number of states, number of transitions and CPU generation time (s) for the third example, $C = 0.99$ and several values of the intertest time T.

T	100	10	1	0.1	0.01
K	25	9	6	3	3
states	100 000	36 000	24 000	12 000	12 000
transitions	1 048 050	377 298	251 532	125 766	125,766
generation time	16.9	5.70	3.73	1.80	1.79

is clear that the greater T is, the greater UA is. Intuitively, the fact that failed spare components are not immediately scheduled for repair "increases" the repair time of these components and so increases UA. We consider the following five values for T: 100, 10, 1, 0.1, and 0.01. For the sake of conciseness, we only give 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 K and $K - 1$ smaller than or equal to 5×10^{-4}. We show in Table 3 the number of required Erlang stages K, the number of states and transitions of the CTMC X, and the CPU generation time in seconds for each value of T.

The state descriptions of the third example have a component μ, $1 \leq \mu \leq K$ 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 μ. In addition, states within each block are sorted following increasing values of μ (from 1 to K). With that ordering, the diagonal matrices $\mathbf{A}_{ii}$ 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 (12) of BGS using Gaussian elimination with fill-in only in the last column.

The iterative methods considered for the third example are GS, SOR, BGS and GMR. We show in Table 4 the number of iterations, CPU time in seconds and UA for the third example. Notice first that as T becomes smaller UA tends to the value corresponding to instantaneous detection of failed spare components (4.461×10^{-6}). The performance of the numerical methods is 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 indicates that the algorithm used for selecting the relaxation parameter ω is efficient. BGS is the method which requires fewer iterations. For $T = 100$ it requires significantly 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 slightly slower than GS and SOR, and for lower values of T it should be clearly considered as the method of choice. Overall, BGS seems to be the method of choice for scenario 3. GMR performs significantly worse in terms of CPU time than BGS and, for $T \geq 0.1$, than GS and SOR. However, it is clearly faster than SOR or GS for $T = 0.01$. It can be observed that for GMR the number of required iterations decreases from $T = 0.1$ to $T = 0.01$, which is in contrast with the behavior observed for greater values of T. We analyzed the behavior of GMR in these two cases and found the following explanation. For $T = 0.1$ the problem is less harder than for $T = 0.01$ and GMR is happy with a smaller value of k ($k = 22$ for $T = 0.1$; $k = 28$ for $T = 0.01$). This smaller value of k makes the convergence slower in the long term.

The fourth example, corresponding to scenario 4, is identical to the second one but now failures of spare components are detected only where they are tested. The test of spare components is performed periodically with deterministic intertest time T, approximated by a K-Erlang distribution with K large enough. The measure of interest is the MTTF, a particular case of MCRTF. The state in which all components are unfailed has initial probability equal to 1. We consider the same values for C and T as we did for the third example. Once again, K is chosen as the minimum value for which the relative variation in the MTTF computed with K and $K-1$ is $\leq 5 \times 10^{-4}$. We show in Table 5 the required K, the number of transient states of X^U, the number of transitions of the chain X^U within the subset U, and the CPU generation time in seconds for several values of T.

For this fourth example we consider the methods GS, SOR, BGS, AGS, ASOR, ABGS, and GMR. Diagonal blocks for BGS and ABGS are chosen and the states within each block sorted as in the previous example. We show in Table 6 the number of iterations, CPU time in seconds required

Table 4: Number of iterations (top), CPU time (s) (bottom) and UA for the third example, $C = 0.99$ and several values of the intertest time T.

	method				
T	GS	SOR	BGS	GMR	UA
100	21	21	10	25	6.129×10^{-6}
	(9.42)	(9.46)	(15.6)	(39.0)	
10	31	31	11	36	4.641×10^{-6}
	(4.81)	(4.83)	(4.92)	(18.8)	
1	162	114	12	100	4.480×10^{-6}
	(15.6)	(11.5)	(3.15)	(30.8)	
0.1	1391	651	12	237	4.463×10^{-6}
	(64.0)	(32.4)	(1.43)	(33.7)	
0.01	12 632	2254	12	207	4.461×10^{-6}
	(584)	(109)	(1.45)	(30.2)	

Table 5: Number of required Erlang stages K, number of transient states of X^U, number of transitions of X^U within U, and CPU generation time (s) for the fourth example with $C = 0.99$ and several values of the intertest time T.

T	100	10	1	0.1	0.01
K	20	7	3	3	3
states	19 200	6720	2880	2880	2880
transitions	166 540	58 289	24 981	24 981	24 981
generation time	3.75	1.27	0.528	0.528	0.541

Table 6: Number of iterations (top), CPU time (s) (bottom) and MTTF for the fourth example, $C = 0.99$ and several values of the intertest time T (an asterisk denotes that the method was unable to converge within 100,000 iterations).

	method							
T	GS	SOR	BGS	AGS	ASOR	ABGS	GMR	MTTF
100	47 267	1623	4896	21	22	12	16	3.882×10^5
	(2.60×10^3)	(106)	(733)	(1.26)	(1.43)	(2.93)	(3.31)	
10	*	10 976	5614	25	28	10	19	4.533×10^5
	*	(243)	(179)	(0.504)	(0.633)	(0.653)	(1.27)	
1	*	*	5629	133	84	10	53	4.618×10^5
	*	*	(58.2)	(0.879)	(0.679)	(0.273)	(1.15)	
0.1	*	*	5572	1092	490	11	87	4.628×10^5
	*	*	(57.7)	(7.11)	(3.90)	(0.248)	(1.93)	
0.01	*	*	5563	9,41	2307	12	127	4.629×10^5
	*	*	(57.9)	(62.7)	(17.9)	(0.267)	(3.45)	

by each method and MTTF. GS performs badly. It is able to reach convergence within 100 000 iterations only for $T = 100$. SOR also fails for values of T smaller than 10, but significantly outperforms GS. The BGS algorithm does not perform very well. In all cases it reaches convergence but the number of iterations and CPU time are large. AGS and ASOR perform well for large and moderate values of T, but they degrade sharply when T decreases. Note that while ASOR is faster than AGS for medium and small values of T, it requires a few more iterations for $T \geq 10$. We analyzed the behavior of ASOR in these two cases and found the following explanation. The ω tuning algorithm changes ω to 1.1 when the algorithm has almost reached convergence with $\omega = 1$. This is a sensible decision since $\widetilde{\eta}(1.1) < \widetilde{\eta}(1)$ but the change in ω produces a "transient" perturbation in the progress of the measure which finally results in a number of iterations slightly greater. ABGS is the best of the seven methods in number of iterations. It is only slightly outperformed in terms of CPU time by AGS and ASOR for $T \geq 10$. As for example 3, the reason for the behavior for these values of T is the time consumed in sorting the states. If a single method were to be chosen, ABGS would be the reasonable choice. GMR, though being slower than ABGS, performs reasonably well and its performance degrades less as T becomes smaller than any other method except ABGS.

The fifth example, corresponding to scenario 5, is the queuing system depicted in Fig. 3. The system, which has been adapted from [16], consists of 3 identical servers with associated finite queues of length C being fed by tasks which arrive following a Poisson process with arrival rate θ. Service time is exponential with average value ψ^{-1}. Servers (but not queues) are subjected to exponentially distributed failures and repairs with rates λ and μ, respectively. A scheduler routes arriving tasks to servers following the join-the-shortest-queue routing algorithm. To simplify the model, we assume that if a server breaks down, the customer being served is kept in the queue and

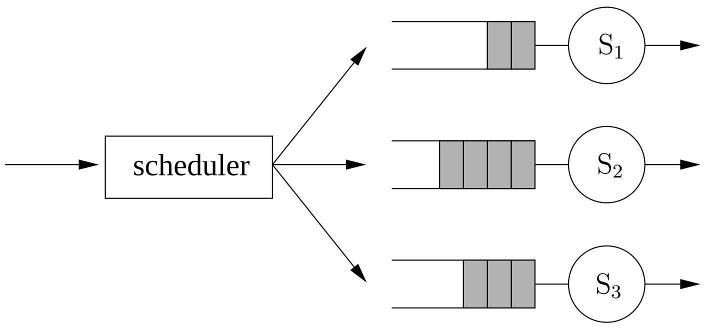


Figure 3: Example queuing system.

Table 7: Sets of model parameter values considered for the fifth example.

Set	a	b
C	15	15
θ	1.60	160
ψ	0.60	60
λ	1×10^{-4}	1×10^{-4}
μ	60	12

its service is resumed after the server has been repaired. The measure of interest is the probability of a customer being rejected, p_{loss}, which can be computed as the probability of the system being in the states in which all queues are full and, therefore, is a particular case of the generic SSRR measure. We use the two sets of model parameter values given in Table 7. Since the number of states and transitions depends only on the queue length C, the generated CTMC is the same for both sets and has $32\ 768$ states and $177\ 144$ transitions. The CPU generation time was 4.26 s. In this example we only experimented with GS, SOR and GMR. We give in Table 8 the number of iterations, CPU time in seconds and p_{loss}. The GMR method requires the least number of iterations. For model parameters set a, SOR works much better than GS does, reducing the number of iterations by a factor approximately equal to 5. For these parameters SOR outperforms GMR as well in terms of CPU time. For set b, however, the improvement of SOR compared to GS is small and GMR outperforms SOR both in number of iterations and CPU time. Note however that GMR has a memory consumption significantly larger than that of SOR and the latter could be preferred when the model is very large.

In order to assess the efficiency of the proposed algorithm for tuning ω in SOR, we will compare the performance of the proposed SOR algorithm with dynamic tuning of ω with that of SOR with ω set to its optimum value, ω_{opt}. The comparison will be made for example 2 with $C = 0.999$ when the linear system to be solved is (4) (i.e. without the acceleration technique) and example 5 for both model parameter sets. The optimum value of the relaxation parameter was found scanning the interval $[1,\ 2)$ for example 2 and $(0,\ 2)$ for example 5. We show in Figs. 4 and 5 the number of iterations required by SOR with fixed ω as a function of ω. The optimum value of the relaxation

Table 8: Number of iterations (top), CPU time (s) (bottom) and loss probability p_{loss} for the fifth example and fro model parameter sets a and b.

	method			
Set	GS	SOR	GMR	p_{loss}
a	1545	308	115	6.929×10^{-4}
	(104)	(24.9)	(33.1)	
b	1614	719	95	6.932×10^{-4}
	(109)	(57.9)	(27.7)	

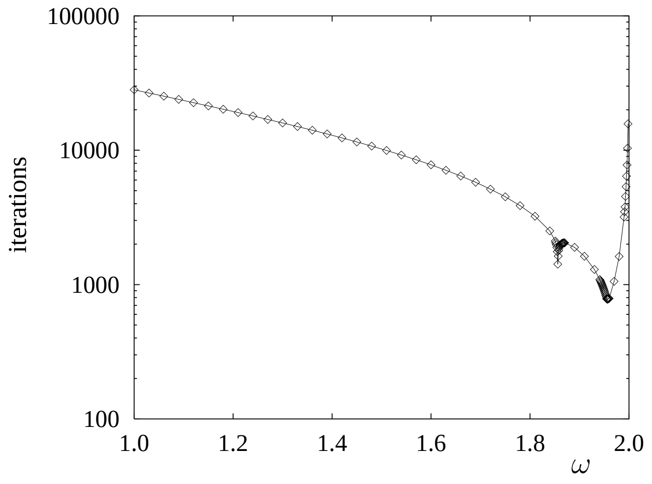


Figure 4: Number of iterations required by SOR with fixed ω to achieve convergence solving the linear system (4) for the second example with $C = 0.999$ as a function of ω.

parameter for example 2 with $C = 0.999$ is 1.956. SOR with ω fixed at this optimum value takes 779 iterations and 0.487 s while the proposed algorithm stopped at $\omega = 1.934$ and took (see Table 2) 1340 iterations and 1.13 s. Therefore, the proposed algorithm performs satisfactorily for this example. Note that η has a second minimum at $\omega = 1.856$. Hence, this example also illustrates how the proposed algorithm can perform well even if η has more than one local minimum in the interval where η is estimated. The optimum values of the relaxation parameter for example 5 are 1.610 and 1.461 for sets a and b, respectively. SOR with ω fixed at those optimum values requires 96 iterations and 6.94 s for set a and 418 iterations and 29.7 s for set b. The proposed algorithm stopped at $\omega = 1.600$ and required (see Table 8) 308 iterations and 24.9 s for set a and stopped at $\omega = 1.496$ and required 719 iterations and 57.9 s for set b. Therefore, the proposed algorithm also performs reasonably well for example 5.

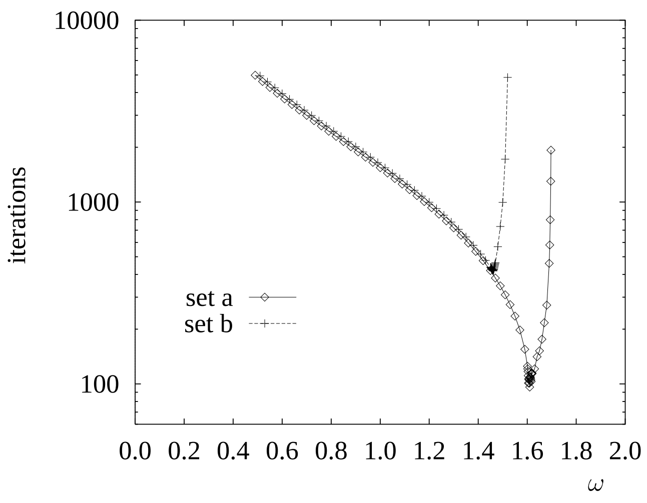


Figure 5: Number of iterations required by SOR with fixed ω to achieve convergence to compute p_{loss} for the fifth example for model parameter sets a and b as a function of ω.

5 Conclusions

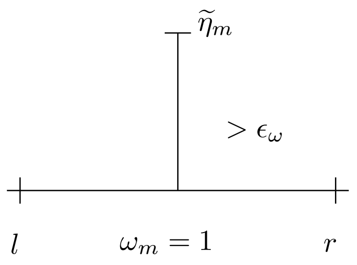
In this paper we have proposed an efficient and robust algorithm to dynamically tune the relaxation parameter ω of SOR. We also have given a sufficient condition for the Gauss-Seidel method to converge for the solution of the linear system which results when the steady-state probability vector of an irreducible CTMC has to be computed. The condition encompasses the very common situation in which the CTMC is generated from a given start state using a set of generation rules and states are numbered increasingly as they are generated. We have developed a variant, called GMR, of the GMRES algorithm in which convergence is monitored based on the relative difference of the computed measure between successive iterates. Also, we have given a sufficient condition on the initial iteration vector for GMRES and GMR not to converge to the trivial solution $\mathbf{0}$ when solving the linear system which arises when the steady-state probability of an irreducible CTMC is computed. We have analyzed and compared several iterative numerical methods in the context of CTMC dependability and performability modeling. We have considered three classes of models: failure/repair models with immediate detection of failed components, failure/repair models with deterministic preventive test of spare components approximated by Erlang distributions, and models without special structure including failure, repair and performance transitions. Two measures have been considered: the steady-state reward rate (SSRR) and the mean cumulative reward to failure (MCRTF). The measure SSRR has been considered for all model classes and the measure MCRTF for the first two classes, giving five scenarios. Experimental results have shown that the method of choice is a splitting-based method for four scenarios and either tuned SOR or GMR for one scenario. In all five scenarios the SOR method with the proposed algorithm to dynamically tune ω clearly outperforms the Gauss-Seidel method when this method does not work well, and performs slightly worse than GS in terms of CPU time when GS is so fast that no tuning on ω can be done before achieving convergence. The performance of GMR is reasonable for all scenarios. However, it takes significantly more memory than splitting-based methods. Both GMR and SOR methods do not require any special structure on the CTMC being solved and both should be offered in a general purpose dependability/performability modeling tool.

Appendix A

Here we give a formal description of the algorithm to dynamically tune the relaxation parameter ω in SOR. The algorithm assumes that $\widetilde{\eta}$ has stabilized to a value < 1 for $\omega = 1$. The algorithm has been implemented as an automaton with seven states. The automaton keeps and updates: (1) the left, l, and right, r, limits of the interval where the optimum ω is searched for, (2) the step to the right, δ_ω^+, and the step to the left, δ_ω^-, for ω, (3) a set of three relaxation parameters, ω_l, ω_m and ω_r, $\omega_\mathrm{l} < \omega_\mathrm{m} < \omega_\mathrm{r}$ for which $\widetilde{\eta}$ has stabilized to a value < 1, as well as the corresponding estimates of η, $\widetilde{\eta}_\mathrm{l}$, $\widetilde{\eta}_\mathrm{m}$ and $\widetilde{\eta}_\mathrm{r}$, (4) the last iteration vector, $\mathbf{x}_\mathrm{last}$, for the last ω for which SOR did not diverge, and (5) a list of values of ω, $\mathcal{W}$, for which $\widetilde{\eta}$ stabilized to a value < 1, sorted from smaller to larger $\widetilde{\eta}$. Initially, $l = 1$ to solve (4), (7) or (8) and $l = 0$ to solve (2), $r = 2$, $\delta_\omega^+ = \delta_\omega^- =$ INI_DELTA, $\omega_\mathrm{m} = 1$, $\widetilde{\eta}_\mathrm{m}$ is set to the estimate of η for $\omega = 1$, $\mathbf{x}_\mathrm{last}$ is the iteration vector which resulted from estimating η for $\omega = 1$, and $\mathcal{W} = \{1\}$. The initial state is 1.

In each state except state 7 which implements the golden section search method, the automaton selects a new value for the relaxation parameter, ω_new, using the procedures $\mathrm{inc\_ome}(\omega,\, a)$, and $\mathrm{dec\_ome}(\omega,\, a)$. The procedure $\mathrm{inc\_ome}(\omega,\, a)$ is called only if $a - \omega > \epsilon_\omega$. The procedure returns $\omega + \delta_\omega^+$ if $\omega + \delta_\omega^+ < a$; otherwise, it computes the minimum integer, m, for which $\omega + \delta_\omega^+/\text{FACT\_DELTA}^m < a$, sets $\delta_\omega^+ = \max\{\epsilon_\omega, \delta_\omega^+/\text{FACT\_DELTA}^m\}$ and returns $\omega + \delta_\omega^+$. The procedure $\mathrm{dec\_ome}(\omega,\, a)$ is called only if $\omega - a > \epsilon_\omega$. The procedure returns $\omega - \delta_\omega^-$ if $\omega - \delta_\omega^- > a$; otherwise, it computes the minimum integer, m, for which $\omega - \delta_\omega^-/\text{FACT\_DELTA}^m > a$, sets $\delta_\omega^- = \max\{\epsilon_\omega, \delta_\omega^-/\text{FACT\_DELTA}^m\}$ and returns $\omega - \delta_\omega^-$. Next, iterations are performed to find an estimate, $\widetilde{\eta}_\mathrm{new}$, of η for ω_new. Let $\mathbf{x}^{(i)}$ be the last iteration vector resulting from these iterations. If $\widetilde{\eta}_\mathrm{new}$ stabilized to a value < 1, ω_new is added to $\mathcal{W}$ and $\mathbf{x}_\mathrm{last}$ is set to $\mathbf{x}^{(i)}$. If $\widetilde{\eta}_\mathrm{new}$ did not stabilize, but $\omega_\mathrm{new} \leq 1$ (for $\omega \leq 1$ SOR is guaranteed to converge) or SOR was not found to diverge, $\mathbf{x}_\mathrm{last}$ is also set to $\mathbf{x}^{(i)}$. If η as a function of ω is found to violate the assumptions on monotonicity or existence of a single local minimum, or the room to tune ω is exhausted, or $\widetilde{\eta}_\mathrm{new}$ does not stabilize but ω_new lies between two already explored values of ω for which $\widetilde{\eta}$ stabilized to a value < 1, or, finally, $\widetilde{\eta}_\mathrm{new} \geq 1$ being $\omega_\mathrm{new} < 1$, the algorithm stops tuning ω and iterates until convergence or the limit number of iterations is exhausted using the procedure *finish* described below. Otherwise, the automaton updates the 8-tuple $\{l,\, r,\, \omega_\mathrm{l},\, \widetilde{\eta}_\mathrm{l},\, \omega_\mathrm{m},\, \widetilde{\eta}_\mathrm{m},\, \omega_\mathrm{r},\, \widetilde{\eta}_\mathrm{r}\}$ and enters a new state. The procedure *finish* simply performs SOR iterations using the values of the relaxation parameter kept in $\mathcal{W}$ beginning with the first value (that with smallest $\widetilde{\eta}$). During these iterations the progress of $\|\mathbf{tol}^{(j)}\|_\infty$ is monitored as it was done to estimate η and if SOR is found to diverge for the current ω, the relaxation parameter is changed into the next value kept in $\mathcal{W}$ and the process continues. The fact that the values of ω for which $\widetilde{\eta}$ stabilized to a value < 1 are kept makes the algorithm more reliable since we have observed that $\widetilde{\eta}$ may stabilize to a value < 1 when indeed SOR diverges for that ω.

Table 9: State 1 of the automaton.

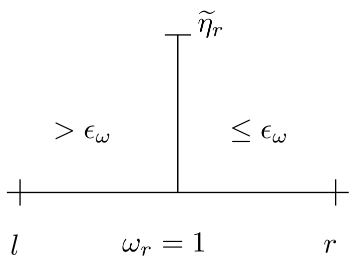


Description: $\widetilde{\eta}$ has only stabilized to a value $\widetilde{\eta}_{\rm m} < 1$ for $\omega_{\rm m} = 1$. There is room on the right of $\omega_{\rm m}$, i.e. $r - \omega_{\rm m} > \epsilon_\omega$. Compute the new value of the relaxation parameter, $\omega_{\rm new}$, as $\omega_{\rm new} = \mathrm{inc\_ome}(\omega_{\rm m}, r)$.

conditions					next
$\widetilde{\eta}_{\rm new}$	$\omega_{\rm new} - \omega_{\rm m}$	$\omega_{\rm m} - l$	$r - \omega_{\rm new}$	actions	state
SNC or NS	$> \epsilon_\omega$			$r = \omega_{\rm new}$	1
SNC or NS	$\leq \epsilon_\omega$	$> \epsilon_\omega$		$r = \omega_{\rm new}$, $\omega_{\rm r} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm m}$	2
SNC or NS	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		*finish*	
$> \widetilde{\eta}_{\rm m}$		$> \epsilon_\omega$		$\omega_{\rm r} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm new}$	4
$> \widetilde{\eta}_{\rm m}$	$> \epsilon_\omega$	$\leq \epsilon_\omega$		$\omega_{\rm l} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	6
$> \widetilde{\eta}_{\rm m}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		*finish*	
$\leq \widetilde{\eta}_{\rm m}$			$> \epsilon_\omega$	$\omega_{\rm r} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm new}$	3
$\leq \widetilde{\eta}_{\rm m}$	$> \epsilon_\omega$		$\leq \epsilon_\omega$	$\omega_{\rm r} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm new}$	5
$\leq \widetilde{\eta}_{\rm m}$	$\leq \epsilon_\omega$		$\leq \epsilon_\omega$	*finish*	

We give in Tables 9–15 a graphical description of each state of the automaton, the actions taken (updates performed on the variables l, r, $\omega_{\rm l}$, $\widetilde{\eta}_{\rm l}$, $\omega_{\rm m}$, $\widetilde{\eta}_{\rm m}$, $\omega_{\rm r}$, and $\widetilde{\eta}_{\rm r}$, or invocation of the procedure *finish*) and, in the case *finish* is not invoked, the next state chosen. For the purposes of the description, SNC (stabilized and nonconvergent) stands for $\widetilde{\eta}_{\rm new}$ stabilized to a value ≥ 1 and NS (nonstabilized) stands for $\widetilde{\eta}_{\rm new}$ not stabilized within $M = \max\{\text{IT\_ETA}, it_{\rm gs}/\text{FACT\_ETA}\}$ iterations ($it_{\rm gs}$ is the number of iterations required for the stabilization of $\widetilde{\eta}$ for $\omega = 1$) or SOR was detected to diverge for $\omega_{\rm new}$. For conciseness, the updating of $\mathbf{x}_{\rm last}$ and $\mathcal{W}$ is not shown.

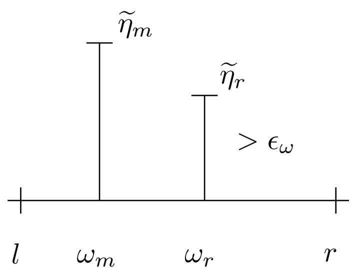
Table 10: State 2 of the automaton.



Description: $\widetilde{\eta}$ has only stabilized to a value $\widetilde{\eta}_{\mathrm{r}} < 1$ for $\omega_{\mathrm{r}} = 1$. There is no room on the right of ω_{r} and there is room on the left, i.e. $r - \omega_{\mathrm{r}} \leq \epsilon_\omega$ and $\omega_{\mathrm{r}} - l > \epsilon_\omega$. Compute the new value of the relaxation parameter, ω_{new}, as $\omega_{\mathrm{new}} = \mathrm{dec\_ome}(\omega_{\mathrm{r}},\, l)$.

conditions				next
$\widetilde{\eta}_{\mathrm{new}}$	$\omega_{\mathrm{r}} - \omega_{\mathrm{new}}$	$\omega_{\mathrm{new}} - l$	actions	state
SNC			*finish*	
NS	$> \epsilon_\omega$		$l = \omega_{\mathrm{new}}$	2
NS	$\leq \epsilon_\omega$		*finish*	
$\geq \widetilde{\eta}_{\mathrm{r}}$	$> \epsilon_\omega$		$\omega_{\mathrm{m}} = \omega_{\mathrm{new}},\, \widetilde{\eta}_{\mathrm{m}} = \widetilde{\eta}_{\mathrm{new}}$	5
$\geq \widetilde{\eta}_{\mathrm{r}}$	$\leq \epsilon_\omega$		*finish*	
$< \widetilde{\eta}_{\mathrm{r}}$		$> \epsilon_\omega$	$\omega_{\mathrm{m}} = \omega_{\mathrm{new}},\, \widetilde{\eta}_{\mathrm{m}} = \widetilde{\eta}_{\mathrm{new}}$	4
$< \widetilde{\eta}_{\mathrm{r}}$	$> \epsilon_\omega$	$\leq \epsilon_\omega$	$\omega_{\mathrm{l}} = \omega_{\mathrm{new}},\, \widetilde{\eta}_{\mathrm{l}} = \widetilde{\eta}_{\mathrm{new}},$ $\omega_{\mathrm{m}} = \omega_{\mathrm{r}},\, \widetilde{\eta}_{\mathrm{m}} = \widetilde{\eta}_{\mathrm{r}}$	6
$< \widetilde{\eta}_{\mathrm{r}}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$	*finish*	

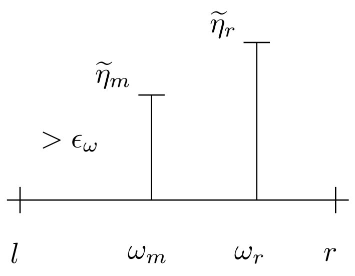
Table 11: State 3 of the automaton.



Description: $\widetilde{\eta}$ has stabilized to a value < 1 for two values of $\omega \geq 1$, ω_m and ω_r, $\omega_\mathrm{m} < \omega_\mathrm{r}$. The corresponding estimates of η, $\widetilde{\eta}_\mathrm{r}$ and $\widetilde{\eta}_\mathrm{m}$, satisfy $\widetilde{\eta}_\mathrm{r} \leq \widetilde{\eta}_\mathrm{m}$. There is room on the right of ω_r, i.e. $r - \omega_\mathrm{r} > \epsilon_\omega$. Compute the new value of the relaxation parameter, ω_new, as $\omega_\mathrm{new} = \mathrm{inc\_ome}(\omega_\mathrm{r}, r)$.

conditions						next
$\widetilde{\eta}_\mathrm{new}$	$\omega_\mathrm{new} - \omega_\mathrm{r}$	$\omega_\mathrm{r} - \omega_\mathrm{m}$	$r - \omega_\mathrm{new}$	$\widetilde{\eta}_\mathrm{r}$	actions	state
SNC or NS	$> \epsilon_\omega$				$r = \omega_\mathrm{new}$	3
SNC or NS	$\leq \epsilon_\omega$	$> \epsilon_\omega$			$r = \omega_\mathrm{new}$	5
SNC or NS	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$			*finish*	
$> \widetilde{\eta}_\mathrm{r}$	$> \epsilon_\omega$			$< \widetilde{\eta}_\mathrm{m}$	$\omega_\mathrm{l} = \omega_\mathrm{m},\ \widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{m},$ $\omega_\mathrm{m} = \omega_\mathrm{r},\ \widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{r},$ $\omega_\mathrm{r} = \omega_\mathrm{new},\ \widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{new}$	7
$> \widetilde{\eta}_\mathrm{r}$	$\leq \epsilon_\omega$	$> \epsilon_\omega$		$< \widetilde{\eta}_\mathrm{m}$	$\omega_\mathrm{l} = \omega_\mathrm{m},\ \widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{m},$ $\omega_\mathrm{m} = \omega_\mathrm{r},\ \widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{r},$ $\omega_\mathrm{r} = \omega_\mathrm{new},\ \widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{new}$	7
$> \widetilde{\eta}_\mathrm{r}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		$< \widetilde{\eta}_\mathrm{m}$	*finish*	
$> \widetilde{\eta}_\mathrm{r}$		$> \epsilon_\omega$		$= \widetilde{\eta}_\mathrm{m}$		5
$> \widetilde{\eta}_\mathrm{r}$		$\leq \epsilon_\omega$		$= \widetilde{\eta}_\mathrm{m}$	*finish*	
$\leq \widetilde{\eta}_\mathrm{r}$			$> \epsilon_\omega$		$\omega_\mathrm{m} = \omega_\mathrm{r},\ \widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{r},$ $\omega_\mathrm{r} = \omega_\mathrm{new},\ \widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{new}$	3
$\leq \widetilde{\eta}_\mathrm{r}$	$> \epsilon_\omega$		$\leq \epsilon_\omega$		$\omega_\mathrm{m} = \omega_\mathrm{r},\ \widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{r},$ $\omega_\mathrm{r} = \omega_\mathrm{new},\ \widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{new}$	5
$\leq \widetilde{\eta}_\mathrm{r}$	$\leq \epsilon_\omega$		$\leq \epsilon_\omega$		*finish*	

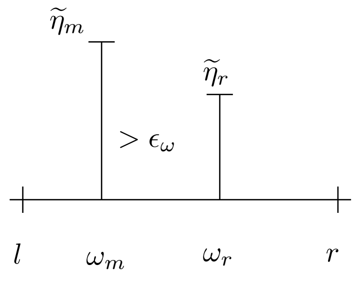
Table 12: State 4 of the automaton.



Description: $\widetilde{\eta}$ has stabilized to a value < 1 for two values of ω, $\omega_{\rm m}$ and $\omega_{\rm r}$, $\omega_{\rm m} < \omega_{\rm r}$, $\omega_{\rm m} \leq 1$. The corresponding estimates of η, $\widetilde{\eta}_{\rm r}$ and $\widetilde{\eta}_{\rm m}$, satisfy $\widetilde{\eta}_{\rm m} < \widetilde{\eta}_{\rm r}$. There is room on the left of $\omega_{\rm m}$, i.e. $\omega_{\rm m} - l > \epsilon_\omega$. Compute the new value of the relaxation parameter, $\omega_{\rm new}$, as $\omega_{\rm new} = \text{dec\_ome}(\omega_{\rm m},\, l)$.

conditions					next
$\widetilde{\eta}_{\rm new}$	$\omega_{\rm m} - \omega_{\rm new}$	$\omega_{\rm r} - \omega_{\rm m}$	$\omega_{\rm new} - l$	actions	state
SNC				*finish*	
NS	$> \epsilon_\omega$			$l = \omega_{\rm new}$	4
NS	$\leq \epsilon_\omega$	$> \epsilon_\omega$		$l = \omega_{\rm new}$, $\omega_{\rm l} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm r}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm r}$	6
NS	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		*finish*	
$> \widetilde{\eta}_{\rm m}$	$> \epsilon_\omega$			$\omega_{\rm l} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm new}$	7
$> \widetilde{\eta}_{\rm m}$	$\leq \epsilon_\omega$	$> \epsilon_\omega$		$\omega_{\rm l} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm new}$	7
$> \widetilde{\eta}_{\rm m}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		*finish*	
$= \widetilde{\eta}_{\rm m}$	$> \epsilon_\omega$			$\omega_{\rm r} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	5
$= \widetilde{\eta}_{\rm m}$	$\leq \epsilon_\omega$			*finish*	
$< \widetilde{\eta}_{\rm m}$			$> \epsilon_\omega$	$\omega_{\rm r} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	4
$< \widetilde{\eta}_{\rm m}$	$> \epsilon_\omega$		$\leq \epsilon_\omega$	$\omega_{\rm l} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm new}$	6
$< \widetilde{\eta}_{\rm m}$	$\leq \epsilon_\omega$		$\leq \epsilon_\omega$	*finish*	

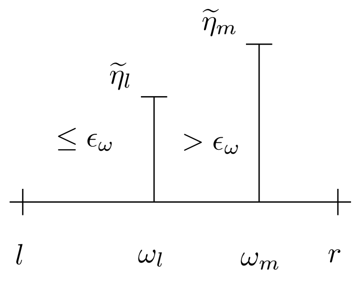
Table 13: State 5 of the automaton.



Description. $\widetilde{\eta}$ has stabilized to a value < 1 for two values of ω, $\omega_{\rm m}$ and $\omega_{\rm r}$, $\omega_{\rm r} > \omega_{\rm m}$. The corresponding estimates of η, $\widetilde{\eta}_{\rm m}$ and $\widetilde{\eta}_{\rm r}$, satisfy $\widetilde{\eta}_{\rm r} \le \widetilde{\eta}_{\rm m}$. There is room between $\omega_{\rm m}$ and $\omega_{\rm r}$, i.e. $\omega_{\rm r} - \omega_{\rm m} > \epsilon_\omega$. There is either no room on the right of $\omega_{\rm r}$, i.e. $r - \omega_{\rm r} \le \epsilon_\omega$, or $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm r}$ and η has been found to increase on the right of $\omega_{\rm r}$. Compute the new value of the relaxation parameter, $\omega_{\rm new}$, as $\omega_{\rm new} = {\rm dec\_ome}(\omega_{\rm r}, \omega_{\rm m})$.

conditions				next
$\widetilde{\eta}_{\rm new}$	$\omega_{\rm r} - \omega_{\rm new}$	$\omega_{\rm new} - \omega_{\rm m}$	actions	state
NS or $> \widetilde{\eta}_{\rm m}$			*finish*	
$\le \widetilde{\eta}_{\rm m}$ and $\ge \widetilde{\eta}_{\rm r}$	$> \epsilon_\omega$		$\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	5
$\le \widetilde{\eta}_{\rm m}$ and $\ge \widetilde{\eta}_{\rm r}$	$\le \epsilon_\omega$		*finish*	
$< \widetilde{\eta}_{\rm r}$	$> \epsilon_\omega$		$\omega_{\rm l} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	7
$< \widetilde{\eta}_{\rm r}$	$\le \epsilon_\omega$	$> \epsilon_\omega$	$\omega_{\rm l} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm l} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	7
$< \widetilde{\eta}_{\rm r}$	$\le \epsilon_\omega$	$\le \epsilon_\omega$	*finish*	

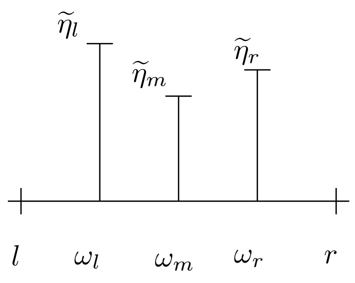
Table 14: State 6 of the automaton.



Description: $\widetilde{\eta}$ has stabilized to a value < 1 for two values of ω, $\omega_{\rm l}$ and $\omega_{\rm m}$, $\omega_{\rm l} < \omega_{\rm m}$. The corresponding estimates of η, $\widetilde{\eta}_{\rm l}$ and $\widetilde{\eta}_{\rm m}$, satisfy $\widetilde{\eta}_{\rm l} < \widetilde{\eta}_{\rm m}$. There is room between $\omega_{\rm l}$ and $\omega_{\rm m}$ but there is not room on the left of $\omega_{\rm l}$, i.e. $\omega_{\rm m} - \omega_{\rm l} > \epsilon_\omega$ and $\omega_{\rm l} - l \leq \epsilon_\omega$. Compute the new value of the relaxation parameter, $\omega_{\rm new}$, as $\omega_{\rm new} = \text{inc\_ome}(\omega_{\rm l}, \omega_{\rm m})$.

conditions				next
$\widetilde{\eta}_{\rm new}$	$\omega_{\rm new} - \omega_{\rm l}$	$\omega_{\rm m} - \omega_{\rm new}$	actions	state
NS or $> \widetilde{\eta}_{\rm m}$			*finish*	
$\leq \widetilde{\eta}_{\rm m}$ and $> \widetilde{\eta}_{\rm l}$	$> \epsilon_\omega$		$\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	6
$\leq \widetilde{\eta}_{\rm m}$ and $> \widetilde{\eta}_{\rm l}$	$\leq \epsilon_\omega$		*finish*	
$= \widetilde{\eta}_{\rm l}$	$> \epsilon_\omega$		$\omega_{\rm m} = \omega_{\rm l}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm l}$, $\omega_{\rm r} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm new}$	5
$= \widetilde{\eta}_{\rm l}$	$\leq \epsilon_\omega$		*finish*	
$< \widetilde{\eta}_{\rm l}$	$> \epsilon_\omega$		$\omega_{\rm r} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	7
$< \widetilde{\eta}_{\rm l}$	$\leq \epsilon_\omega$	$> \epsilon_\omega$	$\omega_{\rm r} = \omega_{\rm m}$, $\widetilde{\eta}_{\rm r} = \widetilde{\eta}_{\rm m}$, $\omega_{\rm m} = \omega_{\rm new}$, $\widetilde{\eta}_{\rm m} = \widetilde{\eta}_{\rm new}$	7
$< \widetilde{\eta}_{\rm l}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$	*finish*	

Table 15: State 7 of the automaton.



Description: A minimum of η has been bracketed, i.e. $\widetilde{\eta}$ has stabilized to a value < 1 for three values of ω, ω_l, ω_m and ω_r, $\omega_\mathrm{l} < \omega_\mathrm{m} < \omega_\mathrm{r}$, and the corresponding estimates of η, $\widetilde{\eta}_\mathrm{l}$, $\widetilde{\eta}_\mathrm{m}$ and $\widetilde{\eta}_\mathrm{r}$, satisfy $\widetilde{\eta}_\mathrm{l} > \widetilde{\eta}_\mathrm{m} < \widetilde{\eta}_\mathrm{r}$. There is room between ω_l and ω_m or between ω_m and ω_r, i.e. $\omega_\mathrm{m} - \omega_\mathrm{l} > \epsilon_\omega$ or $\omega_\mathrm{r} - \omega_\mathrm{m} > \epsilon_\omega$. Compute the new value of the relaxation parameter, ω_new, as $\omega_\mathrm{new} = \omega_\mathrm{m} + (1 - R) \times (\omega_\mathrm{r} - \omega_\mathrm{m})$ if $\omega_\mathrm{r} - \omega_\mathrm{m} > \omega_\mathrm{m} - \omega_\mathrm{l}$ and $\omega_\mathrm{new} = \omega_\mathrm{l} + R \times (\omega_\mathrm{m} - \omega_\mathrm{l})$ otherwise, where $R = (\sqrt{5} - 1)/2$ is the golden section.

Case: $\omega_\mathrm{r} - \omega_\mathrm{m} > \omega_\mathrm{m} - \omega_\mathrm{l}$					
conditions					next
$\widetilde{\eta}_\mathrm{new}$	$\omega_\mathrm{new} - \omega_\mathrm{m}$	$\omega_\mathrm{m} - \omega_\mathrm{l}$	$\omega_\mathrm{r} - \omega_\mathrm{new}$	actions	state
NS or $> \widetilde{\eta}_\mathrm{r}$				*finish*	
$\leq \widetilde{\eta}_\mathrm{r}$ and $\geq \widetilde{\eta}_\mathrm{m}$	$> \epsilon_\omega$			$\omega_\mathrm{r} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{new}$	7
$\leq \widetilde{\eta}_\mathrm{r}$ and $\geq \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$	$> \epsilon_\omega$		$\omega_\mathrm{r} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{new}$	7
$\leq \widetilde{\eta}_\mathrm{r}$ and $\geq \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		*finish*	
$< \widetilde{\eta}_\mathrm{m}$	$> \epsilon_\omega$			$\omega_\mathrm{l} = \omega_\mathrm{m}$, $\widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{m}$, $\omega_\mathrm{m} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{new}$	7
$< \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$		$> \epsilon_\omega$	$\omega_\mathrm{l} = \omega_\mathrm{m}$, $\widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{m}$, $\omega_\mathrm{m} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{new}$	7
$< \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$		$\leq \epsilon_\omega$	*finish*	
Case: $\omega_\mathrm{r} - \omega_\mathrm{m} \leq \omega_\mathrm{m} - \omega_\mathrm{l}$					
conditions					next
$\widetilde{\eta}_\mathrm{new}$	$\omega_\mathrm{m} - \omega_\mathrm{new}$	$\omega_\mathrm{r} - \omega_\mathrm{m}$	$\omega_\mathrm{new} - \omega_\mathrm{l}$	actions	state
NS or $> \widetilde{\eta}_\mathrm{l}$				*finish*	
$\leq \widetilde{\eta}_\mathrm{l}$ and $\geq \widetilde{\eta}_\mathrm{m}$	$> \epsilon_\omega$			$\omega_\mathrm{l} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{new}$	7
$\leq \widetilde{\eta}_\mathrm{l}$ and $\geq \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$	$> \epsilon_\omega$		$\omega_\mathrm{l} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{l} = \widetilde{\eta}_\mathrm{new}$	7
$\leq \widetilde{\eta}_\mathrm{l}$ and $\geq \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$	$\leq \epsilon_\omega$		*finish*	
$< \widetilde{\eta}_\mathrm{m}$	$> \epsilon_\omega$			$\omega_\mathrm{r} = \omega_\mathrm{m}$, $\widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{m}$, $\omega_\mathrm{m} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{new}$	7
$< \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$		$> \epsilon_\omega$	$\omega_\mathrm{r} = \omega_\mathrm{m}$, $\widetilde{\eta}_\mathrm{r} = \widetilde{\eta}_\mathrm{m}$, $\omega_\mathrm{m} = \omega_\mathrm{new}$, $\widetilde{\eta}_\mathrm{m} = \widetilde{\eta}_\mathrm{new}$	7
$< \widetilde{\eta}_\mathrm{m}$	$\leq \epsilon_\omega$		$\leq \epsilon_\omega$	*finish*	

References

[1] W. A. Adkins and S. H. Weintraub, *Algebra. An Approach via Module Theory* (Springer, New York, 1992).

[2] G. P. Barker and R. J. Plemmons, Convergent Iterations for Computing Stationary Distributions of Markov Chains, *SIAM J. Alg. Disc. Meth.* **7(3)** (1986) 390–398.

[3] J. L. Barlow, On the Smallest Positive Singular Value of a Singular M-Matrix with Applications to Ergodic Markov Chains, *SIAM J. Alg. Disc. Meth.* **7(3)** (1986) 414–424.

[4] 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, in: *Proceedings 23rd IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-23)* (1993) 668–673.

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

[6] U. N. Bhat, *Elements of Applied Stochastic Processes* (2nd ed., John Wiley and Sons, 1984).

[7] P. N. Brown and H. F. Walker, GMRES on (Nearly) Singular Systems, *SIAM J. Matrix Anal. Appl.* **18(3)** (1997) 37–51.

[8] 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, eds., *Linear Algebra, Markov Chains and Queuing Models*, (IMA Volumes in Mathematics and its Applications, Springer, 1983) 145–191.

[9] G. Ciardo, J. K. Muppala and K. S. Trivedi, SPNP: Stochastic Petri Net Package, in: *Proceedings 3rd IEEE Int. Workshop on Petri Nets and Performance Models (PNPM89)* (Kyoto, Japan, 1989) 142–150.

[10] 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), 69–80.

[11] 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*, **1(4)** (1994) 403–420.

[12] A. Goyal, W. C. Carter, E. de Souza e Silva, S. S. Lavenberg, and K. S. Trivedi, The System Availability Estimator, in: *Proceedings of the 16th Int. Symp. on Fault-Tolerant Computing (FTCS-16)* (1986) 84–89.

[13] L. Kaufman, B. Gopinath and E. F. Wunderlich, Analysis of Packet Network Congestion Control Using Sparse Matrix Algortihms, *IEEE Trans. on Communications* **COM-29(4)** (1981) 453–465.

[14] U. R. Krieger, B. Müller-Clostermann and M. Sczittnick, Modeling and Analysis of Communication Systems Based on Computational Methods for Markov Chains, *IEEE J. on Selected Areas in Communications* **8(3)** (1990) 1630–1648.

[15] P. Heidelberger, J. K. Muppala and K. S. Trivedi, Accelerating mean time to failure computations, *Performance Evaluation* **27-28** (1996) 627–645.

[16] J. C. S. Lui, R. R. Muntz and D. Towsley, Bounding the Mean Response Time of the Minimum Expected Delay Routing Policy: An Algorithmic Approach, *IEEE Trans. on Computers* **44(12)** (1995) 1371–1382.

[17] D. Mitra and P. Tsoucas, Relaxations for the Numerical Solutions of some Stochastic Problems, *Communications in Statistics. Stochastic Models* **4(3)** (1988) 387–419.

[18] M. Neumann and R. J. Plemmons, Convergent Nonnegative Matrices and Iterative Methods for Consistent Linear Systems, *Numerische Mathematik* **31** (1978) 265–279.

[19] B. Philippe, Y. Saad and W. J. Stewart, Numerical Methods in Markov Chain Modeling, *Operations Research* **40(6)** (1992) 1156–1179.

[20] 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).

[21] Y. Saad, *Iterative Methods for Sparse Linear Systems* (PWS Publishing Company, Boston, MA, 1996).

[22] Y. Saad and M. H. Schultz, GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM J. Sci. Stat. Comput.* **7(3)** (1986) 856–869.

[23] M. Sosonkina, L. T. Watson, R. K. Kapania, and H. F. Walker, A New Adaptive GMRES Algorithm for Achieving High Accuracy, *Numerical Linear Algebra with Applications* **5** (1998) 275–297.

[24] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains* (Princeton University Press, Princeton, NJ, 1994).

[25] W. J. Stewart and A. Goyal, *Matrix Methods in Large Dependability Models* (IBM Thomas J. Watson Research Center, Technical Report RC-11485, 1985).

[26] R. S. Varga, *Matrix Iterative Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1962).

[27] D. M. Young, *Iterative Solution of Large Linear Systems* (Academic Press, Orlando, Florida, 1971).