An efficient block-circulant preconditioner for simulating fracture using large fuse networks

INSTITUTE OF PHYSICS PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND GENERAL

J. Phys. A: Math. Gen. **37** (2004) 2093–2103 PII: S0305-4470(04)68393-4

An efficient block-circulant preconditioner for simulating fracture using large fuse networks*

Phani Kumar V V Nukala and Srdan Simunovic

Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6359, USA

E-mail: nukalapk@ornl.gov

Received 29 August 2003, in final form 20 November 2003
Published 28 January 2004
Online at stacks.iop.org/JPhysA/37/2093 (DOI: 10.1088/0305-4470/37/6/009)

Abstract
Critical slowing down associated with the iterative solvers close to the critical point often hinders large-scale numerical simulation of fracture using discrete lattice networks. This paper presents a block-circulant preconditioner for iterative solvers for the simulation of progressive fracture in disordered, quasi-brittle materials using large discrete lattice networks. The average computational cost of the present algorithm per iteration is $O(rs \log s) + delops$, where the stiffness matrix $\mathbf{A}$ is partitioned into $r \times r$ blocks such that each block is an $s \times s$ matrix, and *delops* represents the operational count associated with solving a block-diagonal matrix with $r \times r$ dense matrix blocks. This algorithm using the block-circulant preconditioner is faster than the Fourier accelerated preconditioned conjugate gradient algorithm, and alleviates the *critical slowing down* that is especially severe close to the critical point. Numerical results using random resistor networks substantiate the efficiency of the present algorithm.

PACS numbers: 62.20.Mk, 46.50.+a

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Progressive damage evolution leading to the failure of disordered quasi-brittle materials has been studied extensively using various types of discrete lattice models [1–8]. Numerical simulation of large lattice networks has often been hampered due to *critical slowing down* associated with the iterative solvers as the lattice system approaches macroscopic fracture. The authors have developed a multiple-rank sparse Cholesky update algorithm based on direct

* The submitted manuscript has been authored by a contractor of the US Government under contract no DE-AC05-00OR22725. Accordingly, the US Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for US Government purposes.

solvers for simulating fracture using discrete lattice systems [9]. Using the algorithm presented in [9], the authors have reported numerical simulation results for large 2D lattice systems (e.g., $L = 512$), which to the authors knowledge, was so far the largest lattice system used in studying damage evolution using discrete lattice systems. Although the sparse direct solvers presented in [9] are superior to iterative solvers in two-dimensional lattice systems, for 3D lattice systems, the memory demands brought about by the amount of *fill-in* during sparse *Cholesky* factorization favour iterative solvers. Hence, iterative solvers are in common use for large-scale 3D lattice simulations. As the lattice system gets closer to macroscopic fracture, the condition number of the system of linear equations increases, thereby increasing the number of iterations required to attain a fixed accuracy. This becomes particularly significant for large lattices. Fourier accelerated preconditioned conjugate gradient (PCG) iterative solvers [10–12] have been used in the past to alleviate the critical slowing down. However, the Fourier acceleration technique based on ensemble averaged circulant preconditioner is not effective when fracture simulation is performed using central-force and bond-bending lattice models [11]. The main focus of the current paper is on developing an efficient algorithm based on iterative solvers for large-scale 3D lattice simulations, and the block-circulant preconditioner presented in the current paper is an effort towards this goal.

Since the Laplacian operator on a discrete lattice network results in the block structure of the stiffness matrix, we propose to use block-circulant matrices [13, 14] as preconditioners to the stiffness matrix for solving this class of problems. The proposed algorithm is benchmarked against the commonly used incomplete LU and Cholesky preconditioners [15], and the *optimal* [14, 16–18] and *superoptimal* [14, 19] circulant preconditioners to the Laplacian operator (Kirchhoff equations). The advantage of using the circulant preconditioners is that they can be diagonalized by discrete Fourier matrices, and hence the inversion of $n_{dof} \times n_{dof}$ circulant matrix can be done in $O(n_{dof} \log n_{dof})$ operations by using FFTs of size n_{dof}. In addition, since the convergence rate of the PCG method depends on the condition number of the preconditioned system, it is possible to choose a circulant preconditioner that minimizes the condition number of the preconditioned system [14, 19]. Furthermore, these circulant preconditioned systems exhibit favourable clustering of eigenvalues. In general, the more clustered the eigenvalues are, the faster the convergence rate is. Another important property of these circulant preconditioners proposed in this study is that they are positive definite if the stiffness matrix itself is positive definite. In this regard, we note that the Fourier accelerated PCG presented in [10–12] is not optimal in the sense described in [14, 16–18], and hence is expected to take more number of CG iterations compared with the *optimal* and *superoptimal* circulant preconditioners.

In this paper, we analyse a *random threshold* model problem, where a lattice consists of fuses having the same conductance, but the bond breaking thresholds, i_c, are based on a broad (uniform) probability distribution, which is constant between 0 and 1. This relatively simple model has been extensively used in the literature for simulating the fracture and progressive damage evolution in brittle materials, and provides a meaningful benchmark for comparing different algorithms. A broad threshold distribution represents large disorder and exhibits diffusive damage leading to progressive localization, whereas a very narrow threshold distribution exhibits brittle failure in which a single crack propagation causes material failure. Periodic boundary conditions are imposed in the horizontal direction to simulate an infinite system and a constant voltage difference (displacement) is applied between the top and the bottom of lattice system. The simulation is initiated with a triangular lattice of intact fuses of size $L \times L$, in which disorder is introduced through random breaking thresholds. The voltage V across the lattice system is increased until a fuse (bond breaking) burns out. The burning of a fuse occurs whenever the electrical current (stress) in the fuse (bond) exceeds the breaking threshold current (stress) value of the fuse. The current is redistributed instantaneously after a fuse is burnt. The voltage is then gradually increased until a second fuse is burnt, and the process is repeated. Each time a fuse is removed, the electrical current is redistributed and hence it is necessary to re-solve Kirchhoff equations to determine the current flowing in the remaining bonds of the lattice. This step is essential for determining the fuse that is going to burn up under the redistributed currents. Therefore, numerical simulations leading to final breaking of lattice system network are very time consuming especially with increasing lattice system size. Consequently, an efficient preconditioner to the Laplacian operator on fractal networks that mitigates the effect of critical slowing down as the lattice system approaches macroscopic fracture is of utmost importance in the numerical simulation of material breakdown.

In the following, we present point-circulant and block-circulant preconditioners for solving the linear system of equations that arise during the numerical simulation of progressive fracture in brittle materials using the random threshold model.

2. Circulant preconditioners for CG iterative solvers

Consider the $n_{dof} \times n_{dof}$ stiffness matrix $\mathbf{A}$. The *optimal* circulant preconditioner $c(\mathbf{A})$ [16] is defined as the minimizer of $\|\mathbf{C} - \mathbf{A}\|_F$ over all $n_{dof} \times n_{dof}$ circulant matrices $\mathbf{C}$. In the above description, $\|\cdot\|_F$ denotes the Frobenius norm [15], and the matrix $c(\mathbf{A})$ is called an *optimal* circulant preconditioner because it minimizes the norm $\|\mathbf{C} - \mathbf{A}\|_F$. The *optimal* circulant preconditioner $c(\mathbf{A})$ is uniquely determined by $\mathbf{A}$, and is given by

$$c(\mathbf{A}) = \mathbf{F}^*\delta(\mathbf{F}\mathbf{A}\mathbf{F}^*)\mathbf{F} \tag{1}$$

where $\mathbf{F}$ denotes the discrete Fourier matrix, $\delta(\mathbf{A})$ denotes the diagonal matrix whose diagonal is equal to the diagonal of the matrix $\mathbf{A}$ and $*$ denotes the adjoint (i.e. conjugate transpose). It should be noted that the diagonal elements of the matrix $\delta(\mathbf{F}\mathbf{A}\mathbf{F}^*)$ represent the eigenvalues of the matrix $c(\mathbf{A})$ and can be obtained in $O(n_{dof} \log n_{dof})$ operations by taking the FFT of the first column of $c(\mathbf{A})$. The first column vector of Chan's *optimal* circulant preconditioner matrix that minimizes the norm $\|\mathbf{C} - \mathbf{A}\|_F$ is given by

$$c_i = \frac{1}{n_{dof}} \sum_{j=1}^{n_{dof}} a_{j,(j-i+1)} \operatorname{mod}_{n_{dof}}. \tag{2}$$

The above formula can be interpreted simply as follows: the element c_i is simply the arithmetic average of those diagonal elements of $\mathbf{A}$ extended to length n_{dof} by wrapping around and containing the element $a_{i,1}$. If the matrix $\mathbf{A}$ is a Hermitian matrix, then the eigenvalues of $c(\mathbf{A})$ are bounded from below and above by

$$\lambda_{\min}(\mathbf{A}) \leqslant \lambda_{\min}(c(\mathbf{A})) \leqslant \lambda_{\max}(c(\mathbf{A})) \leqslant \lambda_{\max}(\mathbf{A}) \tag{3}$$

where $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the minimum and maximum eigenvalues, respectively. Based on the above result, if the matrix $\mathbf{A}$ is positive definite, then the circulant preconditioner $c(\mathbf{A})$ is also positive definite. In particular, if the circulant preconditioner is such that the spectra of the preconditioned system are clustered around 1, then the convergence of the solution will be fast. The *superoptimal* circulant preconditioner $t(\mathbf{A})$ [19] is based on the idea of minimizing the norm $\|\mathbf{I} - \mathbf{C}^{-1}\mathbf{A}\|_F$ over all nonsingular circulant matrices $\mathbf{C}$. In the above description, $t(\mathbf{A})$ is *superoptimal* in the sense that it minimizes $\|\mathbf{I} - \mathbf{C}^{-1}\mathbf{A}\|_F$, and is equal to

$$t(\mathbf{A}) = c(\mathbf{A}\mathbf{A}^*)c(\mathbf{A})^{-1}. \tag{4}$$

The preconditioner obtained by equation (4) is also positive definite if the matrix $\mathbf{A}$ itself is positive definite. Although the preconditioner $t(\mathbf{A})$ is obtained by minimizing the norm

$\|\mathbf{I}-\mathbf{C}^{-1}\mathbf{A}\|_F$, the asymptotic convergence of the preconditioned system is the same as $c(\mathbf{A})$ for large n_{dof} system. Hence, in this study, we limit ourselves to the investigation of preconditioned systems using $c(\mathbf{A})$ given by equation (2). The computational cost associated with the solution of preconditioned system $c(\mathbf{A})\mathbf{z} = \mathbf{r}$ is the initialization cost of $nnz(\mathbf{A})$ for setting the first column of $c(\mathbf{A})$ using equation (2) during the first iteration, and $O(n_{dof} \log n_{dof})$ during every iteration step during every iteration step, where $nnz(\mathbf{A})$.

In order to distinguish the block-circulant preconditioners that follow from the above described circulant preconditioners, we refer henceforth to the above preconditioners as point-circulant preconditioners.

2.1. Block-circulant preconditioners

Let the matrix $\mathbf{A}$ be partitioned into $r \times r$ blocks such that each block is an $s \times s$ matrix. That is, $n_{dof} = rs$, and

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \cdots & \mathbf{A}_{1,r} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \cdots & \mathbf{A}_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{r,1} & \mathbf{A}_{r,2} & \cdots & \mathbf{A}_{r,r} \end{bmatrix}. \tag{5}$$

Although the point-circulant preconditioner $c(\mathbf{A})$ defined by equation (2) can be used as a preconditioner, in general, the block structure is not restored by using $c(\mathbf{A})$ as a preconditioner. In contrast, the circulant-block preconditioners obtained by using circulant approximations for each of the blocks restore the block structure of $\mathbf{A}$. The circulant-block preconditioner of $\mathbf{A}$ can be expressed as

$$c_B(\mathbf{A}) = \begin{bmatrix} c(\mathbf{A}_{1,1}) & c(\mathbf{A}_{1,2}) & \cdots & c(\mathbf{A}_{1,r}) \\ c(\mathbf{A}_{2,1}) & c(\mathbf{A}_{2,2}) & \cdots & c(\mathbf{A}_{2,r}) \\ \vdots & \vdots & \ddots & \vdots \\ c(\mathbf{A}_{r,1}) & c(\mathbf{A}_{r,2}) & \cdots & c(\mathbf{A}_{r,r}) \end{bmatrix}. \tag{6}$$

The circulant-block preconditioner defined by equation (6) is the minimizer of $\|\mathbf{C}-\mathbf{A}\|_F$ over all matrices $\mathbf{C}$ that are $r \times r$ block matrices with $s \times s$ circulant blocks. The spectral properties as given by equation (3) for point-circulant preconditioners also extend to the circulant-block preconditioners [13, 14]. That is,

$$\lambda_{\min}(\mathbf{A}) \leqslant \lambda_{\min}(c_B(\mathbf{A})) \leqslant \lambda_{\max}(c_B(\mathbf{A})) \leqslant \lambda_{\max}(\mathbf{A}). \tag{7}$$

In particular, if the matrix $\mathbf{A}$ is positive definite, then the block preconditioner $c_B(\mathbf{A})$ is also positive definite.

The computational cost associated with the circulant-block preconditioners can be estimated as follows. Since the stiffness matrix $\mathbf{A}$ is real symmetric for the type of problems considered in this study, in the following, we assume block symmetric structure for $\mathbf{A}$, i.e., $\mathbf{A}_{j,i} = \mathbf{A}^t_{i,j}$. In forming the circulant-block preconditioner given by equation (6), it is necessary to obtain point-circulant preconditioners for each of the $r \times r$ block matrices of order s. Point-circulant approximation for each of the $s \times s$ blocks requires $O(s \log s)$ operations. This cost is in addition to the cost associated in forming the first column vectors (equation (2)) for each of the $c(\mathbf{A}_{i,j})$ blocks, which is given by $nnz(\mathbf{A})$ operations. Since there are $(r(r+1))/2$ blocks, we need $O(r^2 s \log s)$ operations to form

$$\boldsymbol{\Delta} = (\mathbf{I} \otimes \mathbf{F}) c_B(\mathbf{A}) (\mathbf{I} \otimes \mathbf{F}^*) = \begin{bmatrix} \delta(\mathbf{F}\mathbf{A}_{1,1}\mathbf{F}^*) & \delta(\mathbf{F}\mathbf{A}_{1,2}\mathbf{F}^*) & \cdots & \delta(\mathbf{F}\mathbf{A}_{1,r}\mathbf{F}^*) \\ \delta(\mathbf{F}\mathbf{A}_{2,1}\mathbf{F}^*) & \delta(\mathbf{F}\mathbf{A}_{2,2}\mathbf{F}^*) & \cdots & \delta(\mathbf{F}\mathbf{A}_{2,r}\mathbf{F}^*) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(\mathbf{F}\mathbf{A}_{r,1}\mathbf{F}^*) & \delta(\mathbf{F}\mathbf{A}_{r,2}\mathbf{F}^*) & \cdots & \delta(\mathbf{F}\mathbf{A}_{r,r}\mathbf{F}^*) \end{bmatrix}. \quad (8)$$

In the above equation, $\otimes$ refers to the Kronecker tensor product and $\mathbf{I}$ is an $r \times r$ identity matrix. In order to solve the preconditioned equation $c_B(\mathbf{A})\mathbf{z} = \mathbf{r}$, equation (8) is permuted to obtain a block-diagonal matrix of the form

$$\tilde{\boldsymbol{\Delta}} = \mathbf{P}^* \boldsymbol{\Delta} \mathbf{P} = \begin{bmatrix} \tilde{\boldsymbol{\Delta}}_{1,1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{\Delta}}_{2,2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \tilde{\boldsymbol{\Delta}}_{s,s} \end{bmatrix} \quad (9)$$

where $\mathbf{P}$ is the permutation matrix such that

$$[\tilde{\boldsymbol{\Delta}}_{k,k}]_{ij} = [\delta(\mathbf{F}\mathbf{A}_{i,j}\mathbf{F}^*)]_{kk} \qquad \forall\, 1 \leqslant i, j \leqslant r \qquad 1 \leqslant k \leqslant s. \quad (10)$$

During each iteration step, in order to solve the preconditioned system $c_B(\mathbf{A})\mathbf{z} = \mathbf{r}$, it is necessary to invert the block-diagonal matrix $\tilde{\boldsymbol{\Delta}}$. This task can be performed by first factorizing each of the $\tilde{\boldsymbol{\Delta}}_{k,k}$ blocks during the first iteration step, and then subsequently using these factored matrices to do the backsolve operations. Hence, without considering the first factorizing cost of each of the block diagonals, during each iteration step, the number of operations involving the inversion of $\tilde{\boldsymbol{\Delta}}$ is

$$delops = O\left(\sum_{k=1}^{s} \left|\mathcal{L}_{\tilde{\boldsymbol{\Delta}}_{k,k}}\right|\right) \quad (11)$$

where $\mathcal{L}_{\tilde{\boldsymbol{\Delta}}_{k,k}}$ denotes the number of non-zeros in the Cholesky factorization of the matrix $\tilde{\boldsymbol{\Delta}}_{k,k}$. Therefore, the system $c_B(\mathbf{A})\mathbf{z} = \mathbf{r}$ can be solved in $O(rs \log s) + delops$ operations per iteration step. Thus, we conclude that for the circulant-block preconditioner, the initialization cost is $nnz(\mathbf{A}) + O(r^2 s \log s)$ plus the cost associated with the factorization of each of the diagonal blocks $\tilde{\boldsymbol{\Delta}}_{k,k}$ during the first iteration, and $O(rs \log s) + delops$ during every iteration step.

Although from operational cost per iteration point of view, the point-circulant preconditioner may prove advantageous for some problems, it is not clear whether point-circulant or circulant-block is closest to the matrix $\mathbf{A}$ in terms of the number of CG iterations necessary for convergence. Hence, we investigate both point-circulant and circulant-block preconditioners in obtaining the solution of the linear system $\mathbf{Ax} = \mathbf{b}$ using iterative techniques. In addition, we also employ the commonly used point and block versions of the *incomplete* LU preconditioners to solve the linear system $\mathbf{Ax} = \mathbf{b}$.

Remark 1. In the case of 2D discrete lattice network with periodic boundary conditions in the horizontal direction and a constant voltage difference between the top and the bottom of the lattice network, the matrix $\mathbf{A}$ is a block tri-diagonal real symmetric matrix. Under these circumstances, the initialization cost is $nnz(\mathbf{A}) + O(rs \log s)$. Since each of the diagonal blocks $\tilde{\boldsymbol{\Delta}}_{k,k}$ is a tri-diagonal matrix, during each iteration step, the solution involving the inversion of $\tilde{\boldsymbol{\Delta}}$ can be obtained in $O(rs)$ operations. Thus, the cost per iteration is $O(rs \log s) + O(sr) = O(rs \log s)$ operations. The total computational cost involved in using

Table 1. Block-circulant PCG: 2D triangular lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
32	10.00	10.68	11 597	20 000
64	135.9	139.8	41 207	1 600
128	2 818	2 846	147 510	192
256	94 717	96 500		32

Table 2. *Optimal* circulant PCG: 2D triangular lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
32	11.66	12.26	25 469	20 000
64	173.6	178.8	120 570	1600
128	7 473	7 725	622 140	128

Table 3. Un-preconditioned CG: 2D triangular lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
32	7.667	8.016	66 254	20 000
64	203.5	205.7	405 510	1 600

the circulant-block preconditioner for a symmetric block tri-diagonal matrix is the initialization cost of $nnz(\mathbf{A}) + O(rs\log s)$, and $O(rs\log s)$ operations per iteration step. This is significantly less than the computational cost involved in using a generic circulant-block preconditioner. It should be noted that the block tri-diagonal structure of $\mathbf{A}$ does not change the computational cost associated with using a point-circulant preconditioner to solve the linear system $\mathbf{Ax} = \mathbf{b}$.

3. Numerical simulation results

In the following, we benchmark the proposed block-circulant preconditioner against the *optimal* [14, 16–18] circulant preconditioner used for the Laplacian operator (Kirchhoff equations). The main purpose behind the 2D lattice simulations presented below is to demonstrate the efficiency of block-circulant preconditioner over the *optimal* circulant preconditioner for the iterative solvers. Once again, we note that the type of ensemble-averaged circulant preconditioner presented in [10–12] is not optimal in the sense described in [14, 16–18], and hence is expected to take more number of CG iterations compared with the *optimal* circulant preconditioners. In the case of 2D lattice systems, we also present the simulation results using *solver type A* of [9] based on sparse direct solvers. As noted earlier, the sparse direct solvers presented in [9] are superior to the iterative solvers for 2D lattice systems, even with the block-circulant preconditioner presented in the current paper. However, the main advantage of the block-circulant preconditioner using iterative solvers is in the case of simulation of 3D lattice systems, where the usage of sparse direct solvers is limited by the (random access) memory constraints.

The numerical results presented in tables 1–5 (for 2D lattices) and 7–10 (for 3D lattices) are performed on a single processor of *Cheetah* (27 Regatta nodes with thirty-two 1.3 GHz Power4 processors each, http://www.ccs.ornl.gov). However, the numerical simulation results presented in tables 6 (for 2D lattices) and 11 (for 3D lattices) are performed on a single processor of *Eagle* (184 nodes with four 375 MHz Power3-II processors) supercomputer at

Table 4. Incomplete cholesky PCG: 2D triangular lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
32	2.831	3.008	5 857	20 000
64	62.15	65.61	29 496	4 000
128	1 391	1 430	148 170	320

Table 5. Computational cost associated with solver type A of [9].

Size	CPU (s)	Wall (s)	N_{config}
32	0.592	0.687	20 000
64	10.72	11.26	4 000
128	212.2	214.9	800
256	5 647	5 662	96
512	93 779	96515	16

Table 6. Number of broken bonds at peak and at failure.

		Triangular			
L	N_{config}	n_p (mean)	n_p (std)	n_f (mean)	n_f (std)
4	50 000	13	3	19	3
8	50 000	41	8	54	7
16	50 000	134	19	168	18
24	50 000	276	32	335	31
32	50 000	465	48	554	46
64	50 000	1 662	130	1 911	121
128	12 000	6 068	386	6 766	349
256	1 200	22 572	1 151	24 474	1 046

the Oak Ridge National Laboratory to run simulations simultaneously on more number of processors. In all the iterative schemes presented below, we employ a residual tolerance of $\epsilon = 10^{-12}$ for convergence of the iterations. Tables 1 and 2 present the cpu and wall-clock times taken on a single processor of *Cheetah* for one configuration (simulation) using the block-circulant and the *optimal* circulant precondioned CG iterative solvers, respectively. In the case of two-dimensional block-circulant PCG, we partition the matrix $\mathbf{A}$ into $L \times L$ blocks such that each block is a $(L + 1) \times (L + 1)$ matrix. For comparison purposes, we also present in tables 3 and 4, the cpu and wall-clock times taken by un-preconditioned and incomplete Cholesky preconditioned CG solvers. Table 5 presents the performance of the sparse direct solver (*Solver Type A*) reported in [9]. As discussed earlier, for 2D lattice systems, the sparse direct solvers and the incomplete Cholesky preconditioner are clearly superior to the block-circulant preconditioned CG iterative solver. However, this advantage of direct solvers (or the preconditioners such as incomplete Cholesky based on direct solvers) vanishes for large 3D lattice systems due to the amount of *fill-in* during Cholesky factorization. In tables 1–5, N_{config} indicates the number of configurations over which ensemble averaging of the numerical results is performed, and the number of iterations denote the average number of total iterations taken to break one intact lattice configuration until it falls apart. For some iterative solvers, the simulations for larger lattice systems were not performed either because they were expected to take larger cpu times or the numerical results do not influence the conclusions drawn in

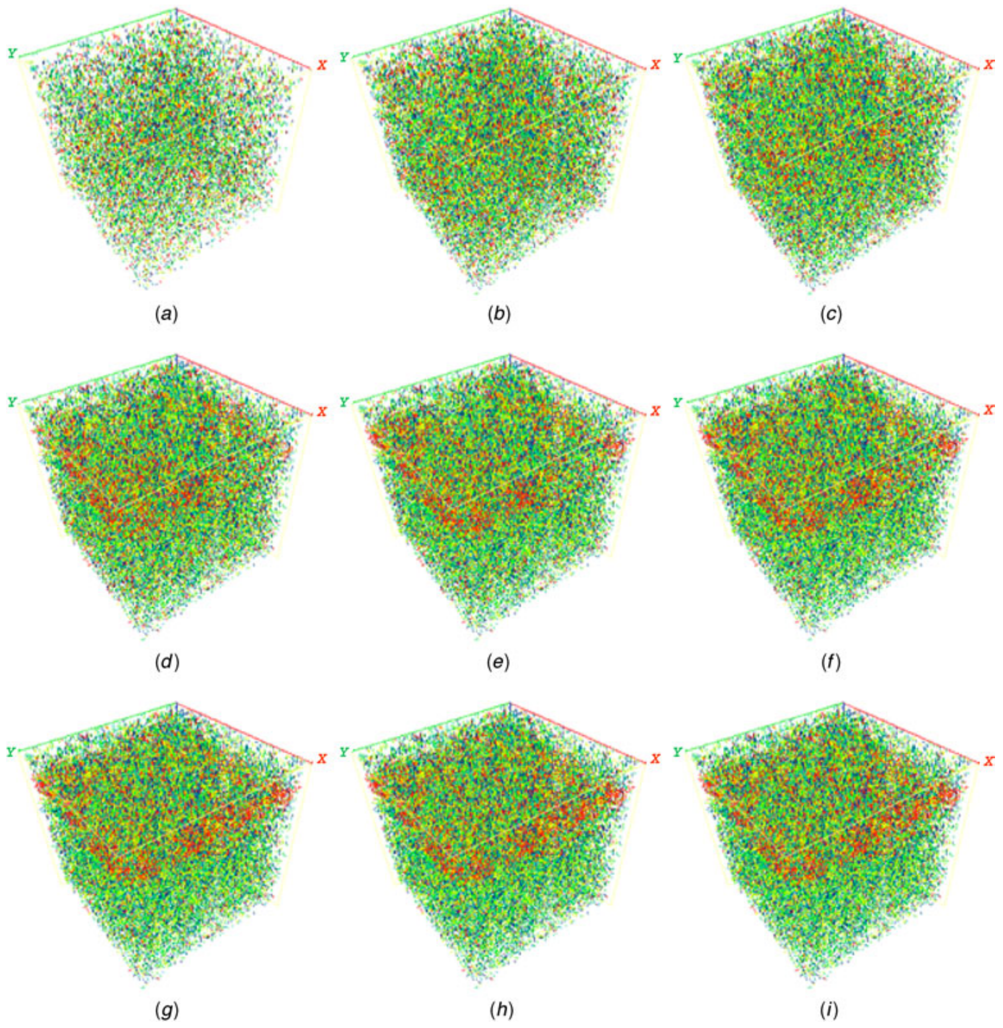

Figure 1. Snapshots of damage in a typical cubic lattice system of size $L = 48$. Number of broken bonds at the peak load and at failure are 48904 and 54744, respectively. (*a*)–(*i*) represent the snapshots of damage after breaking n_b number of bonds. The colouring scheme is such that in each snapshot, the bonds broken in the early stages are coloured blue, then green, followed by yellow, and finally the last stage of broken bonds are coloured red. (*a*) $n_b = 20\,000$ (*b*) $n_b = 40\,000$ (*c*) $n_b = 48\,904$ (peak load) (*d*) $n_b = 51\,000$ (*e*) $n_b = 52\,500$ (*f*) $n_b = 53\,500$ (*g*) $n_b = 54\,000$ (*h*) $n_b = 54\,500$ (*i*) $n_b = 54\,744$ (failure).

this study. In table 6, we present the average number of bonds broken at the peak load and at failure per lattice (triangular) configuration. It should also be noted that in table 6, we were able to perform ensemble averaging over many number of configurations because we were able to run these simulations simultaneously on many number of *Eagle* 375 MHz Power3-II processors.

In addition to the above presented simulations on two-dimensional (2D) triangular lattices, we have also carried out simulations on three-dimensional (3D) cubic lattice networks to investigate the efficiency of block-circulant PCG solvers in 3D simulations. Figure 1 presents the snapshots of progressive damage evolution for the case of a broadly distributed random

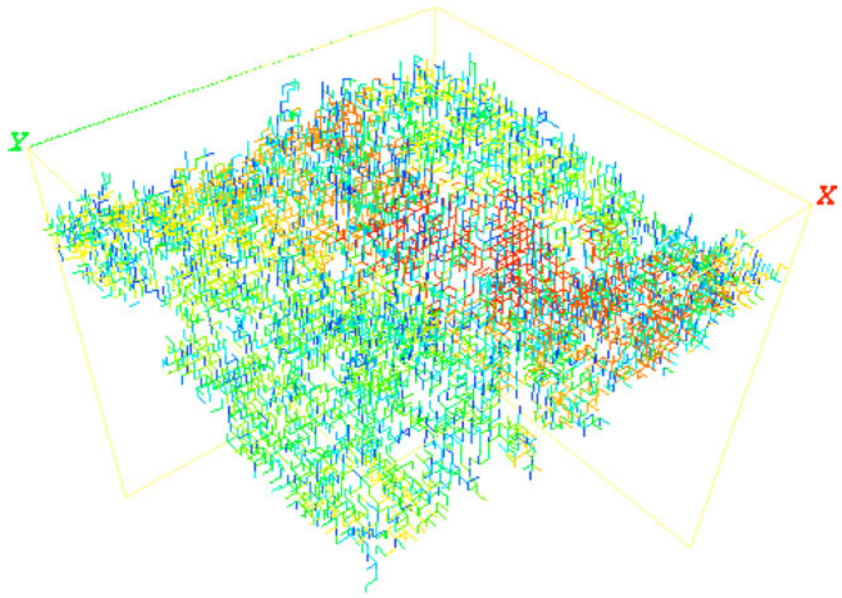

Figure 2. Spanning cluster in a typical cubic lattice system of size $L = 48$. The colouring scheme is such that the bonds broken in the early stages are coloured blue, then green, followed by yellow, and finally the last stage of broken bonds are coloured red.

Table 7. Block-circulant PCG: 3D cubic lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
10	16.54	16.99	16 168	40 000
16	304.6	308.5	58 756	1 920
24	2 154	2 216	180 204	256
32	12 716	12 937	403 459	128
48	130 522	133 063	1253 331	32

Table 8. *Optimal* circulant PCG: 3D cubic lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
10	15.71	16.10	27 799	40 000
16	386.6	391.1	121 431	1 920
24	2 488	2 548	446 831	256
32	20 127	20 380	1142 861	32
48	233 887	237 571	4335 720	32

threshold model problem in a cubic lattice system of size $L = 48$. The spanning cluster is shown in figure 2. Tables 7–10 present the cpu and wall-clock times taken on a single processor of *Cheetah* for simulating one configuration using the block-circulant, *optimal* circulant, un-preconditioned and the incomplete Cholesky iterative solvers, respectively. It should be noted that for large 3D lattice systems (e.g., $L = 32$), the performance of incomplete Cholesky preconditioner (see table 10) is similar to that of the block-circulant preconditioner (see table 7), even though the performance of incomplete Cholesky preconditioner is far more

Table 9. Un-preconditioned CG: 3D cubic lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
10	5.962	6.250	48 417	40 000
16	119.4	123.0	246 072	3 840
24	1 923	1 982	1030 158	256
32	16 008	16 206	2868 193	64

Table 10. Incomplete cholesky PCG: 3D cubic lattice.

Size	CPU (s)	Wall (s)	Iterations	N_{config}
10	5.027	5.262	8 236	40 000
16	118.1	122.3	42 517	3 840
24	1 659	1 705	152 800	512
32	12 091	12 366	422 113	64

Table 11. Number of broken bonds at peak and at failure.

		Cubic			
L	N_{config}	n_p (mean)	n_p (std)	n_f (mean)	n_f (std)
10	40 000	563	57	726	59
16	3 840	2 108	147	2 572	152
24	512	6 692	354	7 882	337
32	128	15 329	705	17 691	649
48	32	49 495	1 582	55 768	1 523

superior in the case of 2D lattice simulations. The memory limitations severely restricted the use of sparse direct solvers for simulating large 3D lattice systems, and hence the results corresponding to the direct solver for 3D lattice systems are not presented. In the case of block circulant PCG, we once again partition the matrix **A** into $L \times L$ blocks of size $(L+1)^2 \times (L+1)^2$ matrices. It should be noted that in order to get maximum efficiency using the block-circulant PCG solver, it is possible to further partition each of the $(L+1)^2 \times (L+1)^2$ matrix blocks into $(L+1) \times (L+1)$ blocks of matrices of size $(L+1) \times (L+1)$. However, the results presented in this study do not perform such nested block-circulant preconditioning. Table 11 presents the average number of bonds broken at the peak load and at failure per lattice configuration.

4. Conclusions

The main focus of the current paper is on developing an efficient algorithm based on iterative solvers for simulating large 3D fuse networks. Although the sparse direct solvers presented in [9] achieve superior performance over iterative solvers in 2D lattice systems, the available random access memory poses a severe constraint over the usage of sparse direct solvers for large 3D lattice systems due to the amount of *fill-in* during sparse Cholesky factorization. In this regard, the block-circulant preconditioner presented in the current paper is an effort towards efficiently solving large 3D fuse networks.

Based on the numerical simulation results presented in tables 1–5 (2D) and tables 7–10 (3D) for random threshold fuse model networks, it is clear that the block-circulant preconditioned CG is superior to the *optimal* circulant preconditioned PCG solver, which in turn is superior to the Fourier accelerated PCG solvers. Furthermore, in the case of large 3D lattice systems, the block-circulant preconditioner exhibits superior performance (for system sizes $L > 32$) over the sparse direct solvers and the related incomplete Cholesky preconditioned CG solvers.

In addition, during the CG iterative solution, the preconditioned system using the block-circulant preconditioner is trivially parallel, and hence a parallel implementation of the block-circulant precondioner can be employed to further speed up the solution of large 3D lattice systems. This allowed us to consider larger 3D lattice simulations, which will be a subject of future publication.

Acknowledgments

This research is sponsored by the Mathematical, Information and Computational Sciences Division, Office of Advanced Scientific Computing Research, US Department of Energy under contract number DE-AC05-00OR22725 with UT-Battelle, LLC. The first author wishes to thank Ed F D'Azevedo for many helpful discussions and excellent suggestions.

References

[1] de Arcangelis L, Redner S and Herrmann H J 1985 A random fuse model for breaking processes *J. Phys. (Paris) Lett.* **46** 585–90
[2] Sahimi M and Goddard J D 1986 Elastic percolation models for cohesive mechanical failure in heterogeneous systems *Phys. Rev.* B **33** 7848–51
[3] Duxbury P M, Beale P D and Leath P L 1986 Size effects of electrical breakdown in quenched random media *Phys. Rev. Lett.* **57** 1052–5
[4] Duxbury P M, Leath P L and Beale P D 1987 Breakdown properties of quenched random systems: the random-fuse network *Phys. Rev.* B **36** 367–80
[5] Hansen A and Roux S 2000 *Statistical Toolbox for Damage and Fracture* (New York: Springer) pp 17–101
[6] Herrmann H J and Roux S (ed) 1990 *Statistical Models for the Fracture of Disordered Media* (Amsterdam: North-Holland)
[7] Sahimi M 1998 Non-linear and non-local transport processes in heterogeneous media from long-range correlation percolation to fracture and materials breakdown *Phys. Rep.* **306** 213–395
[8] Chakrabarti B K and Benguigui L G 1997 *Statistical Physics of Fracture and Breakdown in Disordered Systems* (Oxford: Oxford Science Publications)
[9] Nukala P K V V and Simunovic Srdan 2003 An efficient algorithm for simulating fracture using large fuse networks *J. Phys. A: Math. Gen.* **36** 11403–12
[10] Batrouni G G, Hansen A and Nelkin M 1986 Fourier acceleration of relaxation processes in disordered systems *Phys. Rev. Lett.* **57** 1336–9
[11] Batrouni G G and Hansen A 1988 Fourier acceleration of iterative processes in disordered-systems *J. Stat. Phys.* **52** 747–73
[12] Batrouni G G and Hansen A 1998 Fracture in three-dimensional fuse networks *Phys. Rev. Lett.* **80** 325–8
[13] Chan R H and Jin X-Q 1992 A family of block preconditioners for block systems *SIAM J. Sci. Stat. Comput.* **13** 1218–35
[14] Chan R H and Ng M K 1996 Conjugate gradient methods for toeplitz systems *SIAM Rev.* **38** 427–82
[15] Golub G H and van Loan C F 1996 *Matrix Computations* (Baltimore, MD: Johns Hopkins University Press)
[16] Chan T 1988 An optimal circulant preconditioner for Toeplitz systems *SIAM J. Sci. Stat. Comput.* **9** 766–71
[17] Chan R H 1989 Circulant preconditioners for Hermitian Toeplitz systems *SIAM J. Matrix Anal. Appl.* **10** 542–50
[18] Chan R and Chan T 1992 Circulant preconditioners for elliptic problems *Numer. Lin. Algebra Appl.* **1** 77–101
[19] Tyrtyshnikov E 1992 Optimal and superoptimal circulant preconditioners *SIAM J. Matrix Anal. Appl.* **13** 459–73