PROCEEDINGS OF SCIENCE

Computational challenges in the large-scale simulations of fracture in disordered media

Phani Kumar V.V. Nukala[*†]
Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA
E-mail: nukalapk@ornl.gov

Srđan Šimunović
Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA
E-mail: simunovics@ornl.gov

Computational modeling of fracture in disordered materials using discrete lattice models is often limited to small system sizes due to high computational cost associated with re-solving the governing system of equations every time a new lattice bond is broken. Previously, we proposed an efficient algorithm based on multiple-rank sparse Cholesky downdating scheme for 2D simulations, and an iterative scheme using block-circulant preconditioners for 3D simulations. Based on these algorithms, we were able to simulate large 2D lattice systems (e.g., $L = 1024$). However, despite these algorithmic advances, the largest 3D lattice system that we were able to solve was limited to a size of $L = 64$. In this paper, we present three alternate approaches, namely, the efficient preconditioners, *krylov subspace recycling*, and massive parallelization of the algorithms, the combination of which promise to significantly reduce the computational cost associated with simulating large 3D lattice systems of sizes $L = 200$. The main idea associated with krylov subspace recycling is to retain a subspace determined while solving the current system and reuse it to reduce the cost of solving the subsequent system obtained after removing the new broken bond. Preliminary numerical simulation of fracture using 3D random fuse networks of sizes $L = 64$ substantiates the efficiency of the present algorithms.

International conference on Statistical Mechanics of Plasticity and Related Instabilities
August 29-September 2, 2005
Indian Institute of Science, Bangalore, India

[*]Speaker.

[†]The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. 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 U.S. Government purposes.

PoS(SMPRI2005)037

1. INTRODUCTION

The statistical properties of fracture in disordered media are interesting not only in view of practical applications such as scaling and size effects on the fracture strength, but also for purely theoretical reasons(1). Despite considerable progress, there exist many controversial issues between the theoretically estimated results and the experimentally measured values, and also among various theoretical and numerical models used for studying fracture of disordered media. Among these partly still controversial issues, is the scaling of crack geometries; in particular, the origin of both the scaling and the universality of the fracture surface roughness exponent is at the heart of the controversy.

Experiments have shown that in several materials under different loading conditions the fracture surface is self-affine (2) and the out of plane roughness exponent displays a universal value of $\simeq 0.8$ irrespective of the material studied (3). In particular, experiments have been done in metals (4), glass (5), rocks (6) and ceramics (7), covering both ductile and brittle materials. However, the current understanding that has emerged is that crack roughness displays a universal value of $\simeq 0.8$ only at larger scales and at higher crack speeds, whereas another roughness exponent in the range of $0.4-0.6$ is observed at smaller length scales under quasi-static or slow crack propagation (3).

The situation that exists today is that there is a large discrepancy between theoretical estimates and experimentally measured values of roughness exponents. This is true in both the cases of three-dimensional fracture surface roughness and the interfacial crack front roughness measurements. In both these cases, the numerical simulations have been limited to very small system sizes because of high computational cost involved in performing the numerical simulations. It is believed that roughness measurements based on numerical results obtained using large system sizes is necessary to bridge the gap that exists between numerically estimated roughness exponents and the experimentally measured roughness exponents. Understanding the scaling properties of fracture in disordered media including that of universality of crack surface roughness exponents at small and large length scales represents an intriguing theoretical problem with many technological implications.

Progressive damage evolution leading to failure of disordered quasi-brittle materials has been studied extensively using various types of discrete lattice models (1; 8; 9; 10; 11; 12). In these discrete lattice models, damage is accumulated progressively by breaking one bond at a time until the lattice system falls apart. Each time a bond is broken, it is necessary to redistribute the stresses to determine the subsequent bond failure. Large scale numerical simulation of these discrete lattice networks has often been hampered for two reasons.

- First, a new large set of equations has to be solved everytime a new lattice bond is broken. This becomes especially severe with increasing lattice system size, L, since the number of broken bonds at failure, n_f, increases with system size L as $n_f \sim O(L^{1.8})$ in 2D and $n_f \sim O(L^{2.7})$ in 3D.

- Second, *critical slowing down* associated with the iterative solvers close to the macroscopic fracture. That is, 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.

In addition, since the response of the lattice system corresponds to a specific realization of the random breaking thresholds, an ensemble averaging of numerical results over N_{config} configurations is necessary to obtain a realistic representation of the lattice system response. This further increases the computational time required to perform simulations on large lattice systems.

2. State-of-the-art Algorithms

Algebraically, the process of simulating fracture using discrete lattice systems is equivalent to solving a new set of linear equations

$$\mathbf{A}_n\mathbf{x}_n = \mathbf{b}_n, \quad n = 0,1,2,\ldots, \tag{2.1}$$

every time a new lattice bond is broken. An important feature of fracture simulations using the discrete lattice systems is that, for each $n = 0,1,2,\ldots$, the new matrix $\mathbf{A}_{n+1}$ of the lattice system after the $(n+1)^{th}$ broken bond is equivalent to a rank-p downdate of the matrix $\mathbf{A}_n$ (13). The matrix $\mathbf{A}_n$ refers to the lattice conductance matrix in the case of fuse models and the lattice stiffness matrix in the case of spring and beam models, $\mathbf{b}_n$ refers to the applied nodal current or force vector, and $\mathbf{x}_n$ nodal potential or displacement vector.

2.1 Fourier Acceleration Method

Traditionally, iterative techniques based on preconditioned conjugate gradient (PCG) method have been used to simulate fracture using fuse networks (see Ref. (14) for a excellent review of iterative methods; see Ref. (15) for a review of multigrid method). However, large-scale numerical simulations using iterative solvers have often been hindered due to the *critical slowing down* associated with the iterative solvers as the lattice system approaches macroscopic fracture. As a remedy, Fourier accelerated PCG iterative solvers (16; 17; 18) have been suggested to alleviate the critical slowing down. The Fourier acceleration algorithm proposed in Refs. (16; 17) for solving the system of equations $\mathbf{Ax} = \mathbf{b}$ can be expressed as shown in Algorithm 1. In this subsection, we do not explicitly write the subscript n to refer to the system of equations $\mathbf{A}_n\mathbf{x}_n = \mathbf{b}_n$ obtained after the failure of n bonds. Instead, the subscript n is implicitly is understood.

Algorithm 1 *Algorithm 1* Fourier Acceleration Method (16; 17)

1: Compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{Ax}^{(0)}$ for some initial guess $\mathbf{x}^{(0)}$
2: **for** $k = 1,2,\ldots$ **do**
3: Solve: $\mathbf{Mz}^{(k-1)} = \mathbf{r}^{(k-1)}$
4: $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \mathbf{z}^{(k-1)}$
5: $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \mathbf{Az}^{(k-1)}$
6: Check convergence; continue if necessary
7: **end for**

In Algorithm 1, the matrix $\mathbf{M}$ is referred to as the preconditioner, and the superscript in the paranthesis refers to the iteration number. The steps in the Algorithm 1 can be combined into an iterative scheme as shown below

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{r}^{(k)} \tag{2.2}$$

where $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$. Denoting the error $\mathbf{e}$ (or *algebraic error*) as $\mathbf{e} = \mathbf{x} - \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is the exact solution such that $\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$, the errors in the $(k)^{th}$ and $(k+1)^{th}$ iterations may be related as

$$\mathbf{e}^{(k+1)} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})\mathbf{e}^{(k)} \tag{2.3}$$

Consequently, after (k) number of relaxation sweeps, the error in the k^{th} approximation is given by $\mathbf{e}^{(k)} = \mathbf{P}^k\mathbf{e}^{(0)}$, where $\mathbf{P} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})$. Choosing a particular vector norm and its associated matrix norm, it is possible to bound the error after (k) iterations by $\|\mathbf{e}^{(k)}\| \leq \|\mathbf{P}\|^k\|\mathbf{e}^{(0)}\|$. From this relation, it can be shown that the iteration associated with the matrix $\mathbf{P}$ converges for all initial guesses if and only if $\rho(\mathbf{P}) < 1$, where $\rho(\mathbf{P})$ is the spectral radius defined as

$$\rho(\mathbf{P}) = \max_j |\lambda_j(\mathbf{P})| \tag{2.4}$$

and $\lambda_j(\mathbf{P})$ denotes any of the eigenvalues of $\mathbf{P}$ given by $\lambda_j(\mathbf{P}) = 1 - \lambda_j(\mathbf{M}^{-1}\mathbf{A})$. Hence, convergence of Algorithm 1 is fast whenever the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ are clustered around one, i.e., when $\mathbf{M}^{-1}$ is an approximation of the inverse of $\mathbf{A}$. For computational purposes, $\mathbf{M}$ is chosen such that its inverse can be computed cheaply so that the solution of $\mathbf{M}\mathbf{z} = \mathbf{r}$ can be computed with least computational effort.

The Fourier acceleration algorithm proposed in Refs. (16; 17) chooses an ensemble averaged matrix $\bar{\mathbf{A}}$ (19; 20) as the preconditioner in Algorithm 1, where $\bar{\mathbf{A}}(i,j) = r^{(d_w - d_f)}$, $r = |i-j|$, and d_f and d_w refer to the fractal dimension of the current-carrying backbone and the random-walk dimension respectively. This ensemble averaged matrix $\bar{\mathbf{A}}$ is a symmetric Toeplitz matrix and hence the preconditioned system $\mathbf{M}\mathbf{z} = \mathbf{r}$ in Algorithm 1 can be solved in $O(NlogN)$ operations by using FFTs of size N. Compared with the unconditioned CG methods, the Fourier accelerated technique based on ensemble averaged circulant preconditioner significantly reduced the computational time required to simulate fracture using random fuse networks. However, the ensemble averaged matrix $\bar{\mathbf{A}}$ is not the optimal circulant preconditioner since it does not minimize the norm $\|\mathbf{I} - \mathbf{C}^{-1}\mathbf{A}\|_F$ over all non-singular circulant matrices $\mathbf{C}$. Consequently, the Fourier accelerated scheme proposed in Refs. (16; 17) is expected to take more CG iterations than the *optimal* (21; 22; 23) and *superoptimal* (24; 25) circulant preconditioners.

2.2 Naive Application of Sparse Direct Solvers

Alternatively, since each of the matrices $\mathbf{A}_n$ for $n = 0, 1, 2, \ldots$, is sparse, it is possible to use any of the existing sparse direct solvers (26; 27; 28) to solve each of the system of equations formed by $\mathbf{A}_n$. It should be noted that the Cholesky factorization of $\mathbf{A}_n$ for any particular n can be done efficiently. However, factorization of each of the $\mathbf{A}_n$ matrices for $n_f \sim O(L^{1.8})$ (or $n_f \sim O(L^{2.7})$ in 3D) number of times is still a computationally daunting task. For example, in the case of a 2D triangular lattice system of size $L = 128$, the algorithm based on factorizing each of the $\mathbf{A}_n$ matrices using supernodal sparse direct solver (29) took on an average 8300 seconds compared to an average of 7473 seconds taken by the *optimal* circulant precondioner based CG algorithm. It should however be noted that, in general, the *optimal* circulant precondioner based CG algorithm used in the above comparison is much faster than the Fourier accelerated CG algorithm described in Ref. (17). However, as the following description elaborates, the computational advantages of sparse direct solvers are not fully realized when the algorithm is based on factorizing each of the $\mathbf{A}_n$ matrices.

2.3 Multiple-rank sparse Cholesky downdating algorithm

An important feature of fracture simulations using discrete lattice models is that, for each $n = 0, 1, 2, \ldots$, the new matrix $\mathbf{A}_{n+1}$ of the lattice system after the $(n+1)^{th}$ broken bond is equivalent to a rank-p downdate of the matrix $\mathbf{A}_n$ (13; 30). Mathematically, in the case of the fuse and spring models, the breaking of a bond is equivalent to a rank-one downdate of the matrix $\mathbf{A}_n$, whereas in the case of beam models, it is equivalent to multiple-rank (rank-3 for 2D, and rank-6 for 3D) downdate of the matrix $\mathbf{A}_n$. Thus, an updating scheme of some kind is therefore likely to be more efficient than solving the new set of equations formed by Eq. (2.1) for each n.

Consider the Cholesky factorizations

$$\mathbf{P}\mathbf{A}_n\mathbf{P}^t = \mathbf{L}_n\mathbf{L}_n^t \tag{2.5}$$

for each $n = 0, 1, 2, \cdots$, where $\mathbf{P}$ is a permutation matrix chosen to preserve the sparsity of $\mathbf{L}_n$. Since the breaking of bonds is equivalent to removing the edges in the underlying graph structure of the matrix $\mathbf{A}_n$, for each n, the sparsity pattern of the Cholesky factorization $\mathbf{L}_{n+1}$ of the matrix $\mathbf{A}_{n+1}$ must be a subset of the sparsity pattern of the Cholesky factorization $\mathbf{L}_n$ of the matrix $\mathbf{A}_n$. Hence, for all n, the sparsity pattern of $\mathbf{L}_n$ is contained in that of $\mathbf{L}_0$. That is, denoting the sparsity pattern of $\mathbf{L}$ by $\mathscr{L}$, we have

$$\mathscr{L}_m \supseteq \mathscr{L}_n \quad \forall\, m < n \tag{2.6}$$

For two-dimensional lattice simulations, in Ref. (13), we proposed an efficient algorithm based on multiple-rank sparse Cholesky downdating scheme of Davis and Hager (31; 32) to successively downdate the Cholesky factorizations $\mathbf{L}_n$ of $\mathbf{A}_n$ to $\mathbf{L}_{n+1}$ of $\mathbf{A}_{n+1}$, i.e., $\mathbf{L}_n \rightarrow \mathbf{L}_{n+1}$ for $n = 0, 1, 2, \cdots$. Since $\mathscr{L}_n \supseteq \mathscr{L}_{n+1}$, it is necessary to modify only a part of the non-zero entries of $\mathbf{L}_n$ in order to obtain $\mathbf{L}_n \rightarrow \mathbf{L}_{n+1}$. This results in a significant reduction in the computational time. Once the factorization $\mathbf{L}_{n+1}$ of $\mathbf{A}_{n+1}$ is obtained, the solution vector $\mathbf{x}_{n+1}$ is obtained from $\mathbf{L}_{n+1}\mathbf{L}_{n+1}^t\mathbf{x}_{n+1} = \mathbf{b}_{n+1}$ by two triangular solves (13). Using this algorithm (13), the authors have reported numerical simulation results for large 2D lattice systems (e.g., $L = 1024$), 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 solver algorithm presented in (13) is 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 favor iterative solvers.

2.4 *Optimal* and *superoptimal* circulant preconditioners

The performance of iterative solvers depends crucially on the condition number and clustering of the eigenvalues of the preconditioned system. In general, the more clustered the eigenvalues are, the faster the convergence rate is. The main observation behind developing preconditioners for the iterative schemes is that the operators on discrete lattice network result in a circulant block structure. Hence, a fast Poisson type solver with a circulant preconditioner can be used to obtain the solution in $O(N\ logN)$ operations using FFTs of size N, where N denotes the number of degress of freedom. However, as the lattice bonds are broken successively, the initial uniform lattice grid becomes a diluted network. Consequently, although the matrix $\mathbf{A}_0$ is Toeplitz (also block Toeplitz with Toeplitz blocks) initially, the subsequent matrices $\mathbf{A}_n$, for each n, are not Toeplitz matrices.

However, depending on the pattern of broken bonds, $\mathbf{A}_n$ may still possess block structure with many of the blocks being Toeplitz blocks.

The *optimal* circulant preconditioner $c(\mathbf{A})$ (21; 22; 23; 25) of a matrix $\mathbf{A}$ is defined as the minimizer of $\|\mathbf{C}-\mathbf{A}\|_F$ over all $N\times N$ circulant matrices $\mathbf{C}$. Given a matrix $\mathbf{A}$, the *optimal* circulant preconditioner $c(\mathbf{A})$ is uniquely determined by

$$c(\mathbf{A}) \;=\; \mathbf{F}^{*}\delta\left(\mathbf{F}\mathbf{A}\mathbf{F}^{*}\right)\mathbf{F} \tag{2.7}$$

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 diagonals of $\mathbf{F}\mathbf{A}\mathbf{F}^{*}$ represent the eigenvalues of the matrix $c(\mathbf{A})$ and can be obtained in $O(N\ log\ N)$ operations by taking the FFT of the first column of $c(\mathbf{A})$. The first column vector of T. Chan's *optimal* circulant preconditioner matrix that minimizes the norm $\|\mathbf{C}-\mathbf{A}\|_F$ is given by

$$c_i \;=\; \frac{1}{N}\sum_{j=1}^{N} a_{j,(j-i+1)\bmod N} \tag{2.8}$$

If the matrix $\mathbf{A}$ is Hermitian, the eigenvalues of $c(\mathbf{A})$ are bounded below and above by

$$\lambda_{min}(\mathbf{A}) \le \lambda_{min}(c(\mathbf{A})) \le \lambda_{max}(c(\mathbf{A})) \le \lambda_{max}(\mathbf{A}) \tag{2.9}$$

where $\lambda_{min}(\cdot)$ and $\lambda_{max}(\cdot)$ denote the minimum and maximum eigenvalues, respectively. Based on the above result, if $\mathbf{A}$ is positive definite, then the circulant preconditioner $c(\mathbf{A})$ is also positive definite. The computational cost associated with the solution of the 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 Eq. (2.8) during the first iteration, and $O(N\ log\ N)$ during every iteration step.

The *superoptimal* circulant preconditioner $t(\mathbf{A})$ (24) 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

$$t(\mathbf{A}) = c(\mathbf{A}\mathbf{A}^{*})c(\mathbf{A})^{-1} \tag{2.10}$$

The preconditioner obtained by Eq. (2.10) is also positive definite if $\mathbf{A}$ 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 same as $c(\mathbf{A})$ for large N system.

2.5 Block-circulant preconditioner

Alternatively, block-circulant preconditioners may also be used as preconditioners since many of the block matrices in $\mathbf{A}_n$ for $n>0$ may still retain the circulant or Toeplitz property. Let the matrix $\mathbf{A}_n$ be partitioned into r-by-r blocks such that each block is an s-by-s matrix. That is, $N=rs$. Since each of the blocks of $\mathbf{A}_n$ are Toeplitz (or even circulant), the block-circulant preconditioner is obtained by using circulant approximations for each of the blocks of $\mathbf{A}_n$. It is the minimizer of $\|\mathbf{C}-\mathbf{A}_n\|_F$ over all matrices $\mathbf{C}$ that are r-by-r block matrices with s-by-s circulant blocks. In addition, we have

$$\lambda_{min}(\mathbf{A}_n) \le \lambda_{min}(c_B(\mathbf{A}_n)) \le \lambda_{max}(c_B(\mathbf{A}_n)) \le \lambda_{max}(\mathbf{A}_n) \tag{2.11}$$

In particular, if $\mathbf{A}_n$ is positive definite, then the block-preconditioner $c_B(\mathbf{A}_n)$ is also positive definite. In general, the average computational cost of using the block-circulant preconditioner per iteration is $O(rs\ log\ s) + delops$, where $delops$ represents the operational cost associated with solving a block-diagonal matrix with $r \times r$ dense blocks. For 2D and 3D discrete lattice network with periodic boundary conditions in the horizontal disrection, this operational cost reduces significantly.

For 3D lattice systems, the block-circulant preconditioner has been shown to exhibit superior performance over the sparse direct solvers and the related incomplete Cholesky preconditioned CG solvers (33). In addition, for both 2D and 3D lattice systems, 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 used in Refs. (16; 17).

Using this block circulant preconditioner (33), we were able to simulate fracture in lattice systems of sizes up to $64 \times 64 \times 64$, which is the largest ever discrete lattice system used for fracture simulation of broadly disordered materials. Table 1 presents a summary of computational times required for simulating the fracture of one sample configuration of a 3D cubic lattice system. These simulations have been performed on a IBM 1.3 GHz Power4 processor. The simulation begins with an intact lattice system and is carried out by breaking one bond at a time until the macroscopic fracture occurs. The computational times presented in Table 1 represent the CPU times required to simulate one such sample configuration, and the number of iterations represent the cumulative number of CG iterations required to fracture one such typical sample configuration. It should be noted that the simulation of a $L = 48$ cubic lattice system took approximately 1.5 days (130522 seconds) of computational time. In comparison, a typical fracture simulation of a $64 \times 64 \times 64$ lattice system approximately took 14 days of CPU time using the *optimal* block circulant preconditioner as it was necessary to solve a system of equations with $N \sim 64^3$ variables for about $n_f = 126577$ number of times. Using the block-circulant preconditioner, the CPU times necessary for simulating a cubic lattice system of size $(L \times L \times L)$ appear to scale as $\mathscr{O}(L^{6.5})$, which poses severe computational requirements for simulating fracture using large lattice system sizes. This is precisely the problem where massively parallel simulation using CG algorithm with block-circulant preconditioner may be explored.

Table 1: Block Circulant PCG (3D Cubic Lattice)

Size	CPU(sec)	Iterations
10	16.54	16168
16	304.6	58756
24	2154	180204
32	12716	403459
48	130522	1253331
64	1180230	

3. Future Directions and Computational Challenges

As mentioned earlier, the performance data presented in the previous section represents the performance of algorithms on a single processor. Alternatively, it is possible to employ paral-

lel sparse or iterative solvers to perform numerical simulation of fracture. Since iterative solvers exhibit excellent scalability with respect to number of processors, parallel iterative solvers are especially suitable for performing large scale fracture simulations using three-dimensional lattice networks. Even then, large scale 3D simulations of the order $L = 200$ may still be beyond the current computational capability. For example, assuming a perfectly linear scaleup, a system of size $L = 128$ would require 4.5 days of computational time on 512 IBM 1.3 GHz Power4 processors or its equivalents, whereas a system of size $L = 200$ requires 20 days of CPU time on 2048 IBM 1.3 GHz Power4 processors. These numbers suggest that large scale 3D simulations using brute force parallel solvers alone may not be practical at present, if not impossible. Moreover, the communication overhead between processors may not result in a linear scaleup of the problem size with the number of processors. In addition to the above computational requirements, the computational complexity of analyzing many sample configurations to obtain a realistic ensemble averaged response becomes even more significant for large system sizes. In this sense, along with massive parallelization, it may be worthwhile to look into alternative algorithmic strategies/preconditioners for solving the system of equations that arise in the simulation of large 3D lattice systems.

In the following, we propose three alternate ways in which efficient simulation of large-scale 3D lattice systems can be achieved:

- Efficient preconditioners for solving each $\mathbf{A}_n\mathbf{x}_n = \mathbf{b}_n$ using either algebraic multigrid (AMG) methods or Kronecker products.
- Recycling of Krylov subspaces to obtain the solution of each of the subsequent systems $\mathbf{A}_{n+1}\mathbf{x}_{n+1} = \mathbf{b}_{n+1}$ with minimal effort.
- Massive parallelization of the preconditioned iterative scheme.

3.1 Krylov subspace recycling

The main idea is to develop an algorithm based on recycling of the Krylov subspace determined while solving $\mathbf{A}_n\mathbf{x}_n = \mathbf{b}_n$ and use it to reduce the cost of solving the $\mathbf{A}_{n+1}\mathbf{x}_{n+1} = \mathbf{b}_{n+1}$ (34). Such a recycling process amounts to a reduction of iteration count required to solve the new linear system $\mathbf{A}_{n+1}\mathbf{x}_{n+1} = \mathbf{b}_{n+1}$, and hence increases the overall efficiency of the algorithm. In the case of random fuse and spring models, the successive $\mathbf{A}_n$ matrices differ by a rank-1 matrix with a very sparse structure, and this enables us to develop efficient subspace recycling algorithms. In our preliminary analysis using a 3D cubic lattice system of size $L = 64$, the krylov subspace recycling algorithm reduced the computational time by as much as 30%. This percentage is expected to be higher for larger lattice system sizes. Currently, we are investigating various approaches that take advantage of the rank-1 structure of the matrix downdates.

3.2 Massively parallel algorithms

As mentioned earlier, a trivial way to achieve speedup is through massive parallelization of the current state-of-the-art algorithms presented in Section 2 on supercomputers with thousands of processors. However, quite often, this modification alone, i.e., simply parallelizing the algorithms such as those presented in Section 2, will not result in a capability that can simulate as large three-dimensional systems as $L = 200$. The main reason for this inefficiency is the inter-processor

communication overhead, which starts to dominate the computational CPU time. Although an iterative or a direct sparse solution of any of the $\mathbf{A}_n\mathbf{x}_n = \mathbf{b}_n$ systems can be performed efficiently, the main computational bottleneck arises due to the fact that one needs to solve such systems $n_f \sim \mathcal{O}(L^{2.7})$ (in 3D) number of times. Whenever communication overhead dominates the computational CPU time, the algorithm fails to scaleup, i.e., for a given computational time, it is not possible to increase the problem size simply by increasing the number of processors. Alternatively, for a given problem size, it is not possible to decrease the computational time simply by increasing the number of processors.

As an example, we considered the 3D cubic fuse model of size $L = 64$, which took approximately 14 days of computational time on a single processor. The number of equations in the linear system $\mathbf{A}_n\mathbf{x}_n = \mathbf{b}_n$ are of the order of $N \sim 64^3$, and for a typical sample configuration such linear systems were solved approximately $n_f = 126577$ number of times. Using the block-circulant preconditioner, the algorithm scaled-up until 16 processors, beyond which the communication overhead dominated the computational expense. Although the parallelization significantly reduced the computational time, from 14 days on a single processor to about a day on sixteen processors, further increase in efficiency was not achieved simply by increasing the number of processors. It should however be noted that much larger lattice systems will scaleup to many more processors before the communication overhead once again dominates the computational complexity. Currently, we are considering possible approaches of combining efficient preconditioners and Krylov subspace recycling techniques with massive parallelization to develop an algorithm suitable for simulating large 3D systems of the order of $L = 200$.

4. CONCLUSIONS

In this paper, we present three alternate approaches to increase the efficiency of algorithms used in large scale 3D simulation of fracture using discrete lattice networks such as fuse, spring and beam networks. These approaches, namely, efficient preconditioners, *krylov subspace recycling*, and massive parallelization of the algorithms significantly reduce the computational time required for simulating large 3D lattice systems. Our preliminary studies on a 3D cubic lattice system of size $L = 64$ indicate that krylov subspace recycling can reduce the computational time by as much as 30%, and a straight forward massive parallelization of our iterative schemes using block-circulant preconditioners can reduce the computational time from 14 days to a day. However, even with these novel approaches, analysis of a 3D lattice system of size $L = 200$ appears to be impractical, if not impossible. Currently, we are exploring the alternatives of combining massive parallelization with krylov subspace recycling techniques and efficient preconditioners, the combination of which promises to significantly reduce the computational cost associated with simulating large 3D lattice systems of the order of $L = 200$.

Acknowledgment
PKVVN is sponsored by the Mathematical, Information and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy under contract number DE-AC05-00OR22725 with UT-Battelle, LLC.

References

[1] H. J. Herrmann and S. Roux (eds.), *Statistical Models for the Fracture of Disordered Media*, (North-Holland, Amsterdam, 1990). B. K. Chakrabarti and L. G. Benguigui, *Statistical physics of fracture and breakdown in disordered systems* (Oxford Univ. Press, Oxford, 1997).

[2] B. B. Mandelbrot, D. E. Passoja, and A. J. Paullay, Nature (London) **308**, 721 (1984).

[3] For a review see E. Bouchaud, J Phys. C **9**, 4319 (1997)

[4] K.J. Maloy, A. Hansen, E.L. Hinrichsen, and S. Roux, Phys. Rev. Lett. 68, 213 (1992); E. Bouchaud, G. Lapasset, J. Planés, and S. Navéos, Phys. Rev. B 48, 2917 (1993).

[5] P. Daguier, B. Nghiem, E. Bouchaud, and F. Creuzet, Phys. Rev. Lett. 78, 1062 (1997).

[6] J. Schmittbuhl, S. Roux, and Y. Berthaud, Europhys. Lett. 28, 585 (1994). J. Schmittbuhl, F. Schmitt, and C. Scholz, J. Geophys. Res. 100, 5953 (1995).

[7] J.J. Mecholsky, D.E. Passoja, and K.S. Feinberg-Ringel, J. Am. Ceram. Soc. 72, 60 (1989).

[8] L. de Arcangelis, S. Redner, and H. J. Herrmann, J. Phys. (Paris) Lett. **46** 585 (1985).

[9] M. Sahimi and J. D. Goddard, Phys. Rev. B, **33**, 7848 (1986).

[10] P. M. Duxbury, P. D. Beale, and P. L. Leath, Phys. Rev. Lett. **57**, 1052 (1986).

[11] P. M. Duxbury, P. L. Leath, and P. D. Beale, Phys. Rev. B **36**, 367 (1987).

[12] A. Hansen and S. Roux, *Statistical toolbox for damage and fracture*, 17-101, in book *Damage and Fracture of Disordered Materials*, eds. D. Krajcinovic and van Mier, Springer Verlag, New York, 2000.

[13] P. K. V. V. Nukala, and S. Simunovic, J. Phys. A: Math. Gen. **36**, 11403 (2003).

[14] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*, SIAM, Philadelphia, 1994.

[15] W. L. Briggs, Van Emden Henson and S. F. McCormick, *A Multigrid Tutorial, 2nd Edition*, SIAM, Philadelphia, 2000.

[16] G. G. Batrouni, A. Hansen, and M. Nelkin, Phys. Rev. Lett. **57**, 1336 (1986).

[17] G. G. Batrouni, A. Hansen and M. Nelkin, J. Stat. Phys. **52**, 747 (1988).

[18] G. G. Batrouni and A. Hansen, Phys. Rev. Lett. **80**, 325 (1998).

[19] O'Shaughnessy and I. Procaccia, Phys. Rev. Lett. **54**, 455 (1985).

[20] O'Shaughnessy and I. Procaccia, Phys. Rev. A **32**, 3073 (1985).

[21] T. Chan, SIAM J. Sci. Stat. Comput. **9**, 766 (1988).

[22] R. H. Chan, SIAM J. Matrix Anal. Appl. **10**, 542 (1989).

[23] R. Chan and T. Chan, Numerical Linear Algebra Applications, **1**, 77 (1992).

[24] E. Tyrtyshnikov, SIAM J. Matrix Anal. Appl. **13**, 459 (1992).

[25] R. H. Chan and M. K. Ng, SIAM Review **38**(3), 427-482 (1996).

[26] A. Gupta, M. Joshi, and V. Kumar, Technical Report RC 22038 (98932), IBM T. J. Watson Research Center, Yorktown Heights, NY (2001)

[27] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, SIAM J. Matrix Anal. Appl. **20**, 720 (1999).

[28] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster SIAM J. Matrix Anal. Appl. **23**(1), 15 (2001).

[29] S. Toledo, D. Chen, and V. Rotkin TAUCS: A library of sparse linear solvers, http://www.tau.ac.il/ stoledo/taucs/ (2001).

[30] P. K. V. V. Nukala, S. Simunovic, and M. N. Guddati, Int. J. Numer. Meth. Engng. **62**, 1982 (2005).

[31] T. A. Davis and W. W. Hager, SIAM J. Matrix Anal. Appl. **20**(3), 606-627 (1999).

[32] T. A. Davis and W. W. Hager, SIAM J. Matrix Anal. Appl. **22**(4), 997-1013 (2001).

[33] P. K. V. V. Nukala, and S. Simunovic, J. Phys. A: Math. Gen. **37**, 2093 (2004).

[34] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, SIAM Journal on Scientific Computing (in press, 2006).