LARGE SCALE SIMULATION OF FRACTURE NETWORKS*

Phani Kumar V.V. Nukala and Srdan Simunovic
Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6359, USA

ABSTRACT

Computational modeling of fracture in disordered (heterogeneous) media using discrete lattice models is often limited to small system sizes due to high computational cost involved in re-solving the governing system of equations every time a new lattice bond is broken. For two-dimensional simulations, this paper proposes an efficient algorithm based on multiple-rank sparse Cholesky downdating scheme. Based on the proposed algorithm, the authors present simulation results for large 2D lattice systems (e.g., L=1024), which to the authors knowledge, is so far the largest lattice system used in studying the damage evolution. For three-dimensional simulations, we propose efficient algorithms based on iterative schemes. The block-circulant preconditioner and the current algorithm based on *superfast Toeplitz solver* are intended for large-scale 3D discrete lattice simulations.

1 INTRODUCTION

Progressive damage evolution leading to fracture of disordered (heterogeneous) quasi-brittle materials has been studied extensively by material scientists, physicists, and engineers for many years (Herrmann [1], Hansen [2], Krajcinovic [3], Sahimi [4], Bazant [5], van Mier [6]). Qualitatively, progressive damage evolution in disordered (heterogeneous) quasi-brittle materials is characterized by three distinct stages of damage evolution, namely, diffusive damage, crack growth, and crack coalescence (localization) stages. The diffusive damage accumulation stage reflects the pre-existing material disorder (heterogeneity), and the later stages of crack growth and coalescence reflect the dominance of stress concentrations over the pre-existing material disorder.

Discrete lattice models have often been used in simulating damage evolution in broadly disordered quasi-brittle materials (de Arcangelis [7], Sahimi [8], Herrmann [1], Hansen [2]). In these models, damage is accumulated progressively by breaking one bond at a time until the lattice system falls apart. Large scale numerical simulation of these discrete lattice networks has often been limited due to the fact that a new large set of equations has to be solved every time a lattice bond is broken. Since the number of broken bonds at failure, n_f, increases with increasing lattice system sizes, L, i.e., $n_f \sim O(L^{1.9})$, numerical simulation of large lattice systems becomes prohibitively expensive. Furthermore, in fracture simulations using discrete lattice networks, ensemble averaging of numerical results is necessary to obtain a realistic representation of the lattice system response. This further increases the computational cost associated with modeling fracture simulations in disordered quasi-brittle materials using large discrete lattice networks.

To address this problem, the present study proposes two alternatives. The first algorithm is based on Davis and Hager's (Davis [9], Davis[10]) multiple-rank sparse Cholesky downdating scheme using sparse direct solvers, and is especially suitable for two-dimensional simulations. Using this algorithm, the authors present simulation results for large 2D lattice systems (e.g., L=1024), which to the authors knowledge, is the largest lattice system used for studying the

damage evolution in disordered materials starting with initially intact lattice systems.

The second class of algorithms is based on iterative techniques. Although the above algorithm based on sparse direct solvers is superior to iterative solvers in two-dimensional simulations, for 3D systems, the memory demands brought about by the amount of *fill-in* during sparse Cholesky factorization favor iterative solvers. However, *critical slowing down* associated with the iterative solvers close to the macroscopic fracture often hinders large-scale simulation of fracture using iterative techniques. The *critical slowing down* of the iterative solvers is associated with the increasing condition number of the system of linear equations as the lattice system gets closer to macroscopic fracture. That is, the number of iterations required to attain a fixed accuracy increases close to macroscopic fracture. This paper presents two iterative algorithms that alleviate the critical slowing down. The first one is based on block-circulant preconditioner (Chan [11], Chan [12], Nukala[13]), using which the authors have been able to simulate large 3D systems (e.g., L=48). The second algorithm is based on *superfast Toeplitz solver* (Stewart [14]).

2 MODEL

The discrete lattice beam model considered in this work is a *random thresholds model*, and has been studied extensively for simulating fracture of quasi-brittle materials (de Arcangelis [7], Sahimi [10], Herrmann [1], Hansen [2], Sahimi [4], van Mier [6]). In the random thresholds model with (linear) beam elements, the lattice is initially fully intact with bonds having the same stiffness, but the bond breaking stress thresholds, t, are randomly distributed based on a thresholds probability distribution, $p(t)$. The breaking of a bond occurs irreversibly, whenever the effective stress in the bond exceeds the breaking threshold stress value, t, of the bond. Periodic boundary conditions are imposed in the horizontal direction to simulate an infinite system and a constant displacement, Δ, is applied at the top while constraining the bottom row of nodes of the lattice system.

Numerically, a unit displacement, Δ=1, is applied at the top of the lattice, and the resulting system of linear equations ($\boldsymbol{A}\,\boldsymbol{x}=\boldsymbol{b}$, where $\boldsymbol{A}$ is the assembled global stiffness matrix, $\boldsymbol{x}$ the nodal displacement vector, and $\boldsymbol{b}$ the external nodal force vector) is solved to determine the $\boldsymbol{x}$. Using the nodal displacements $\boldsymbol{x}$, the local forces and the effective stress in each of the bonds is evaluated. Subsequently, for each bond j, the ratio between the effective stress s_j and the bond breaking threshold t_j is evaluated, and the bond j_c having the largest value, $max_j(s_j/t_j)$, is irreversibly removed (broken). The stresses are redistributed instantaneously after a bond is broken implying that the stress relaxation in the lattice system is much faster than the breaking of a bond. Each time a bond is broken, it is necessary to re-calculate the stress (or force) redistribution in the lattice to determine the subsequent breaking of a bond. The process of breaking of a bond, one at a time, is repeated until the lattice system falls apart. In this work, we consider both a uniform bond breaking thresholds probability distribution, which is constant between 0 and 1, and also a mapping of the heterogeneity based on a randomly generated computer particle images.

This study considered numerical simulations on triangular lattice topology for 2D systems, and cubic (FCC) lattices for 3D systems. For many lattice system sizes, the number of sample configurations, N_{config}, used are excessively large to reduce the statistical error in the numerical results. Each numerical simulation was performed on a single processor of *Eagle* (184 nodes with four 375 MHz Power3-II processors) supercomputer at the Oak Ridge National Laboratory. The statistically independent N_{config} number of configurations were simulated simultaneously on number of processors available for computation.

3 ALGEBRAIC PROBLEM

Algebraically, fracture of discrete lattices by breaking one bond at a time is equivalent to solving a new set of linear equations ($\mathbf{A}_n\mathbf{x}_n=\mathbf{b}_n$, for all n=0,1,2,…) every time a new lattice bond is broken. In the above expression, each matrix $\mathbf{A}_n$ is an *N-by-N* symmetric and positive definite matrix, $\mathbf{b}_n$ is the *N-by-1* (given) applied nodal current or force vector, $\mathbf{x}_n$ is the *N-by-1* (unknown) nodal potential or displacement vector, and N is the number of degrees of freedom (unknowns) in the lattice. The subscript n indicates that $\mathbf{A}_n$ and $\mathbf{b}_n$ are evaluated after the n^{th} bond is broken. The solution $\mathbf{x}_n$, obtained after the n^{th} bond is broken, is used in determining the subsequent ($(n+1)^{th}$) bond to be broken. The matrices $\mathbf{A}_n$ and $\mathbf{b}_n$ are obtained from the element (bond) matrices using the standard finite element assembly procedure (Hughes [15]).

Mathematically, breaking of a bond in a discrete lattice model is equivalent to downdating of the matrix **A**. For example, in the case of electrical fuse and spring models, breaking of a bond is equivalent to a rank-one downdate of the matrix **A**, while for a 3D beam model, breaking of a bond is equivalent to multiple-rank (rank-6) downdate of the matrix **A**. Furthermore, it is noted that if the Cholesky factorizations are

$$\mathbf{P}\mathbf{A}_n\mathbf{P}^t = \mathbf{L}_n\mathbf{L}_n^t \qquad (1)$$

for each n=0,1,2,…, where **P** is a permutation matrix chosen to preserve the sparsity of $\mathbf{L}_n$, then the sparsity pattern of $\mathbf{L}_{n+1}$ is contained in that of $\mathbf{L}_n$. Hence, for all n, *the sparsity pattern of* $\mathbf{L}_n$ *is contained in that of* $\mathbf{L}_0$.

Based on the above description of successive $\mathbf{A}_n$ for n=0,1,2,…, an updating scheme of some kind is therefore likely to be more efficient than solving the new set of equations formed by $\mathbf{A}_n\mathbf{x}_n=\mathbf{b}_n$ for each n. In particular, since the successive matrices $\mathbf{A}_n$, for each n=0,1,2,…, differ by a rank-six matrix, we successively update the Cholesky factorizations $\mathbf{L}_n$ of $\mathbf{A}_n$, using the sparse Cholesky factorization update algorithm of Davis and Hager (Davis [9], Davis[10]). That is, we use multiple-rank sparse Cholesky updating scheme to update $\mathbf{L}_n \rightarrow \mathbf{L}_{n+1}$ for each n.

3.1 Sparse Cholesky Factorization Update: $\mathbf{L}_n \rightarrow \mathbf{L}_{n+1}$

The algorithm presented in (Davis [9], Davis [10]) is based on the analysis and manipulation of the underlying graph structure of the stiffness matrix **A** and on the methodology presented in (Gill [16]) for modifying a dense Cholesky factorization. In particular, since the breaking of bonds is equivalent to removing the edges in the underlying graph structure of the stiffness matrix **A**, the new sparsity pattern of the modified **L** must be a subset of the sparsity pattern of the original **L**. Denoting the sparsity pattern of **L** by $\mathcal{L}$, we have $\mathcal{L}_m \supseteq \mathcal{L}_n$, $m<n$. Therefore, we can use the modified dense Cholesky factorization update (Davis [9]) and work only on the non-zero entries in **L**. Furthermore, since the changing non-zero entries in **L** depend only on the on the variables associated with the i^{th} and j^{th} nodes of the bond *ij* that is broken, it is only necessary to modify the non-zero elements of a submatrix of **L**.

The multiple-rank sparse Cholesky update algorithm updates the Cholesky factorization $\mathbf{L}_n$ of the matrix $\mathbf{A}_n$ to $\mathbf{L}_{n+1}$ of the new matrix $\mathbf{A}_{n+1}$, where $\mathbf{A}_{n+1} = \mathbf{A}_n + \sigma\mathbf{Y}\mathbf{Y}^t$, and **Y** represents a *N*-by-*p* rank-p matrix. The pseudo-code in Algorithm 1 follows the **matlab** syntax (Golub [17]) and its sparse matrix functionalities very closely. The following notation is used. ***zeros***(*m*,*n*): an *m*-by-*n* matrix of zero entries; $\mathbf{L}(i_1{:}i_2,j) = \mathbf{L}_{(i1:i2,j)}$: refers to entries corresponding to i_1^{th} to i_2^{th} rows of column *j* of the matrix **L**; **[ilist, jlist, val] = find**($\mathbf{L}(i_1{:}i_2,j)$): extracts the sparsity pattern of $\mathbf{L}(i_1{:}i_2,j)$. That is, the non-zeros of $\mathbf{L}(i_1{:}i_2,j)$ are stored in ***val***, and the corresponding row and column indices are stored in ***ilist*** and ***jlist***, respectively; *j*+***ilist***: increment each of the entries of ***ilist***

by *j*; ***length***(***ilist***): length of the vector ***ilist***; ***Sparse***(***ilist***,***jlist***,***val***,*m*,*n*): create a *m*-by-*n* sparse matrix with non-zero entries ***val*** located at the row and column indices given by ***ilist*** and ***jlist*** respectively. Using the above notation, the multiple-rank sparse Cholesky factor update (σ=+1) or downdate (σ=-1) algorithm, $\mathbf{L}_{n+1}$ = ***SpChol***$(\mathbf{L}_m, \mathbf{Y}, \sigma)$, where $\mathbf{L}_{n+1}\mathbf{L}_{n+1}^t = \mathbf{L}_m\mathbf{L}_m^t + \sigma\mathbf{Y}\mathbf{Y}^t$, and ***Y*** represents a *N*-by-*p* rank-p matrix, is given by

Algorithm 1 *SpChol*$(\mathbf{L}, \mathrm{Y}, \sigma)$: Rank-p sparse Cholesky update/downdate algorithm

Convert $\mathrm{LL}^t \rightarrow \tilde{\mathrm{L}}\mathrm{D}\tilde{\mathrm{L}}^t$
Initialize: ilist =**jlist** = vlist = $\mathrm{zeros}(\mathrm{nnz}(\tilde{\mathrm{L}}), 1)$; $\quad ifree = 1$
Set $\alpha_i = 1$ for all $i = 1, 2, \cdots, p$
for $j = 1$ to N **do**
  Algorithm 5 from Davis et al. ? (insert *Algorithm 2* from below)
  Update the non-zeros of column j of $\tilde{\mathrm{L}}$ (insert pseudo-code 3 from below)
end for
$nz = ifree - 1$
Update $\tilde{\mathrm{L}} = \mathrm{I} + \mathrm{Sparse}(\mathrm{ilist}(1:nz), \mathrm{jlist}(1:nz), \mathrm{vlist}, N, N)$
Convert $\tilde{\mathrm{L}}\mathrm{D}\tilde{\mathrm{L}}^t$ to LL^t

Algorithm 2 *Algorithm 5* from Davis and Hager's

for $i = 1$ to p **do**
  if $Y_{ji} \neq 0$ **then**
    $\bar{\alpha} = \alpha_i + \sigma Y_{ji}^2 / D_{jj}$ $\quad \{(\sigma = +1$ for update or -1 for downdate$)\}$
    $D_{jj} = \bar{\alpha} D_{jj}$
    $\gamma_i = \sigma\, Y_{ji} / D_{jj}$
    $D_{jj} = D_{jj} / \alpha_i$
    $\alpha_i = \bar{\alpha}$
  end if
end for

Algorithm 3 Pseudo-code for updating the non-zeros of column j of $\tilde{\mathrm{L}}$

if $j < N$ **then**
  $[\mathrm{iylist}, \mathrm{jylist}, \mathrm{yval}] = \mathrm{find}(\mathrm{L}_{(j+1):N,j})$
  for $i = 1$ to p **do**
    if $Y_{ji} \neq 0$ and yval $\neq \emptyset$ **then**
      $\mathrm{Y}_{j+\mathrm{iylist},i} = \mathrm{Y}_{j+\mathrm{iylist},i} - Y_{ji}\,\mathrm{yval}$
      $[\mathrm{mm}, \mathrm{nn}, \mathrm{ww}] = \mathrm{find}(\mathrm{Y}_{(j+1):N,i})$
      $\mathrm{vv} = \mathrm{zeros}(N, 1)$;
      $\mathrm{vv}_{j+\mathrm{iylist}} = \mathrm{yval}$; $\quad \mathrm{vv}_{j+mn} = \mathrm{vv}_{j+mn} + \gamma_i\,\mathrm{ww}$;
      $[\mathrm{iylist}, \mathrm{jylist}, \mathrm{yval}] = \mathrm{find}(\mathrm{vv}((j+1):N)$
    end if
  end for
  $nz = \mathrm{length}(\mathrm{iylist})$
  $\mathrm{ilist}(ifree : (ifree + nz - 1)) = \mathrm{iylist} + j$
  $\mathrm{jlist}(ifree : (ifree + nz - 1)) = j$
  $\mathrm{vlist}(ifree : (ifree + nz - 1)) = \mathrm{yval}$
  $ifree = ifree + nz$
end if

Given the factorization $\mathbf{L}_m$ of $\mathbf{A}_m$, rank-1 sparse Cholesky update/downdate (Algorithm 1) is used to update the factorization $\mathbf{L}_{n+1}$ for all subsequent values of n=m,m+1,…. Once the factorization $\mathbf{L}_{n+1}$ of $\mathbf{A}_{n+1}$ is obtained, the solution vector $\mathbf{x}_{n+1}$ is obtained by a backsolve operation of $\mathbf{L}_{n+1}\mathbf{L}_{n+1}^{t}\mathbf{x}_{n+1} = \mathbf{b}_{n+1}$.

3.2 Iterative Algorithms

As mentioned earlier, although the algorithm based on sparse direct solvers achieve superior performance over iterative solvers in 2D lattice simulations, the available memory poses a severe constraint over the usage of sparse direct solvers for 3D lattice simulations due to the amount of fill-in during sparse Cholesky factorization. In this regard, we propose the block-circulant preconditioner (Chan [11], Chan [12], Nukala [13]) and an algorithm based on *superfast Toeplitz solver* (Stewart [14]).

The main observation behind the iterative schemes developed in here is that the initial stiffness matrix $\mathbf{A}_0$ is a Toeplitz matrix since the initial lattice grid is uniform. Hence, a fast Poisson type solver with a circulant preconditioner can be used to obtain the solution in $O(N \log N)$ operations using FFTs of size N. However, as the lattice bonds are broken successively, the initial uniform lattice grid becomes a fractal 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, $\mathbf{A}_n$ may still possess block structure with many of the blocks being Toeplitz blocks depending on the pattern of broken bonds.

3.2.1 Block Circulant Preconditioner

Let the matrix $\mathbf{A}$ 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), it is possible to to choose a block-circulant preconditioner obtained by using circulant approximations for each of the blocks of $\mathbf{A}_n$. It is the minimizer of $|\mathbf{C} - \mathbf{A}|_F$ over all matrices $\mathbf{C}$ that are r-by-r block matrices with s-by-s circulant blocks. In addition, we have

$$\lambda_{\min}(\mathbf{A}) \le \lambda_{\min}(c_B(\mathbf{A})) \le \lambda_{\max}(c_B(\mathbf{A})) \le \lambda_{\max}(\mathbf{A}) \qquad (2)$$

In particular, if $\mathbf{A}$ is positive definite, then the block-preconditioner $c_B(\mathbf{A})$ 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-by-r dense blocks. For 2D and 3D discrete lattice network with periodic boundary conditions in the horizontal disrection, this operational cost reduces significantly. The reader is referred to (Nukala[13]) for further details on block-circulant preconditioners for discrete lattice networks.

3.2.2 Algorithm Based on Superfast Toeplitz Solver

Since breaking of bonds successively is equivalent to $\mathbf{A}_{n+1} = \mathbf{A}_n + \sigma\mathbf{Y}\mathbf{Y}^t$, where $\mathbf{Y}$ is a N-by-p rank-p matrix with at most twelve non-zero entries (for beam models,, p=6) in each of the columns, we propose to use the following algorithm to update the solution vector $\mathbf{x}_{n+1}$. Note that $\mathbf{A}_{n+1} = \mathbf{A}_n + \sigma\mathbf{V}_n\mathbf{V}_n^t$, where $\mathbf{V}_n$ is a N-by-$(n+1)\,p$ matrix obtained by concatenating the

each of the **Y** matrices corresponding to the (n+1) broken bonds respectively.

Algorithm Solution update using superfast Toeplitz solver, $TzSolve$

1: Solve $\mathrm{A}_0\mathrm{z}_1 = \mathrm{b}_{n+1}$ using $\mathrm{z}_1 = TzSolve(\mathrm{A}_0, \mathrm{b}_{n+1})$
2: Form $\mathrm{z}_2 = \mathrm{V}_n^t\mathrm{z}_1$
3: Solve $(\mathrm{I}+\mathrm{B})\,\mathrm{z}_3 = \mathrm{z}_2$ where $\mathrm{B} = \sigma\mathrm{V}_n^t\mathrm{A}_0^{-1}\mathrm{V}_n$ using CG: $\mathrm{z}_3^{(i+1)} = \mathrm{z}_2 - \mathrm{B}\mathrm{z}_3^{(i)}$.
4: Form $\mathrm{z}_4 = \mathrm{V}_n\mathrm{z}_3$
5: Solve $\mathrm{A}_0\mathrm{z}_5 = \mathrm{z}_4$ using $\mathrm{z}_5 = TzSolve(\mathrm{A}_0, \mathrm{z}_4)$
6: $\mathrm{x}_{n+1} = \mathrm{z}_1 - \mathrm{z}_5$

It should be noted that $\mathbf{B}\,\mathbf{z}_3^{(i)}$ can be calculated successively as $\mathbf{B}\,\mathbf{z}_3^{(i)} = \mathrm{s}\,\mathbf{V}_n^{\,t}\,\boldsymbol{TzSolve}(\mathbf{A}_0, (\mathbf{V}_n\,\mathbf{z}_3^{(i)}))$

4 SUMMARY

This paper proposes two classes of efficient algorithms based on sparse disrect solvers and iterative solvers for large scale simulation of fracture in broadly disordered (heterogeneous) media using discrete lattice networks. The proposed methodology is applicable for scalar models such as random fuse models as well as vector models based on spring, beam and born models. For two-dimensional simulations, the algorithm based on multiple-rank sparse Cholesky downdating scheme is computationally most efficient. Based on the proposed algorithm, the authors present simulation results for large 2D lattice systems (e.g., L=1024), which by far the largest lattice system used in studying the damage evolution in quasi-brittle materials. However, the memory constraints imposed by the amount of fill-in during sparse Cholesky factorization limits the applicability of sparse direct solvers for large scale 3D simulations. The present study proposes two efficient iterative techniques based on block-circulant preconditioner and an algorithm based on superfast Toeplitz solver to alleviate the critical slowing down associated with the iterative techniques close to macroscopic fracture. The block-circulant preconditioner has been used in simulating a cubic lattice network of size L=48. Further research is underway in extending the simulation capabilities for 3D discrete lattice systems.

ACKNOWLEDGEMENTS

This research 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] Herrmann H. J. and Roux S., *Statistical Models for the Fracture of Disordered Media*. North-Holland, Amsterdam, 1990.

[2] Hansen A. and Roux S., *Statistical toolbox for damage and fracture*, pages 17–101. Springer, New York, 2000.

[3] Krajcinovic D., *Damage Mechanics*. Elsevier, Amsterdam, 1996.

[4] Sahimi M., Non-linear and non-local transport processes in heterogeneous media from long-range correlation percolation to fracture and materials breakdown. *Physics Reports*, 306:213–395, 1998.

[5] Bazant Z. P. and Planas J., *Fracture and Size Effect in concrete and other quasibrittle materials*. CRC Press, 1998.

[6] van Mier J. G. M., *Fracture processes in concrete: Assessment of material parameters for fracture models*. CRC Press, 1996.

[7] de Arcangelis L., Redner S., and Herrmann H. J., A random fuse model for breaking processes. *Journal of Physics (Paris) Letters*, 46(13):585–590, 1985.

[8] Sahimi M. and Goddard J. D., Elastic percolation models for cohesive mechanical failure in heterogeneous systems. *Physical Review B*, 33:7848–7851, 1986.

[9] Davis T. A. and Hager W. W., Modifying a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 20(3):606–627, 1999.

[10] Davis T. A. and Hager W. W., Multiple-rank modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 22(4):997–1013, 2001.

[11] Chan T., An optimal circulant preconditioner for Toeplitz systems. *SIAM J. Sci. Stat. Comput.*, 9:766–771, 1988.

[12] Chan R. H. and Ng M. K., Conjugate gradient methods for Toeplitz systems. *SIAM Review*, 38(3):427–482, 1996.

[13] Nukala P. K. V. V. and Simunovic S., An efficient block-circulant preconditioner for simulating fracture using large fuse networks. *J. Phys. A: Math. Gen.*, 37:2093–2103, 2004.

[14] Stewart M., A superfast toeplitz solver with improved numerical stability. *SIAM J. Matrix Anal. Appl.*, 25(3):669–693, 2003.

[15] Hughes T.J.R., *The Finite Element Method*. Prentice-Hall, 1987.

[16] Gill P. E., Golub G. H., Murray W., and Saunders M. A., Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[17] Golub G. H. and van Loan C. F., *Matrix Computations*. The Johns Hopkins University Press, 1996.