Procedia Computer Science

Volume 29, 2014, Pages 1580–1589

ICCS 2014. 14th International Conference on Computational Science

Enhancing Monte Carlo Preconditioning Methods for Matrix Computations

Janko Straßburg[1] and Vassil Alexandrov[2]

[1] The University of Reading, UK and Barcelona Supercomputing Centre, Barcelona, Spain
janko.strassburg@bsc.es

[2] ICREA Research Professor in Computational Science at Barcelona Supercomputing Centre, Barcelona, Spain
vassil.alexandrov@bsc.es

Abstract

An enhanced version of a stochastic SParse Approximate Inverse (SPAI) preconditioner for general matrices is presented. This method is used in contrast to the standard deterministic preconditioners computed by the deterministic SPAI, and its further optimized parallel variant-Modified SParse Approximate Inverse Preconditioner (MSPAI). Thus we present a Monte Carlo preconditioner that relies on the use of Markov Chain Monte Carlo (MCMC) methods to compute a rough matrix inverse first, which is further optimized by an iterative filter process and a parallel refinement, to enhance the accuracy of the preconditioner. Monte Carlo methods quantify the uncertainties by enabling us to estimate the non-zero elements of the inverse matrix with a given precision and certain probability. The advantage of this approach is that we use sparse Monte Carlo matrix inversion whose computational complexity is linear of the size of the matrix. The behaviour of the proposed algorithm is studied, its performance measured and compared with MSPAI.

Keywords:

1 Introduction

Solving systems of linear algebraic equations (SLAE) in the form of $Ax = b$ or inverting a real matrix A is of unquestionable importance in many scientific and engineering applications. They can be found in digital signal processing, stochastic modelling or communications, and many physical problems involving partial differential equations.

Iterative solvers are used widely to compute the solutions of these systems and such approaches are often the method of choice due to their predictability and reliability when considering accuracy and speed. They are, however, prohibitive for large-scale problems as they can be very time consuming to compute. These methods are dependent on the size of the matrix

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.143

and so the computational effort grows with the problem size. The complexity of these methods is $O(kn^2)$ for dense matrices in the iterative case and $O(n^3)$ for direct methods with dense matrices while solving SLAE if common elimination or annihilation schemes (e.g. Gaussian elimination, Gauss-Jordan methods) are employed [18].

Therefore, these algorithms often rely on preconditioners to speed up the computations and/or to ensure faster convergence.

Monte Carlo (MC) methods on the other hand can quickly yield a rough estimate of the solution. This is done by performing random sampling of a certain variable whose mathematical expectation is the desired solution. For some problems an estimate is sufficient or even favourable, due to the accuracy of the underlying data. For example we would not need to process data with a higher precision than the one of the input data. In addition, Monte Carlo methods help to qualify the uncertainties while performing matrix inversion. For example, Monte Carlo approaches enable us to estimate the non-zero elements of the inverse matrix with a given precision, and with certain probability, and also enables us to estimate the structure of the inverse matrix. Therefore, we concentrate on Monte Carlo methods for matrix inversion (MI) that only require $O(NL)$ steps to find a single element or a row of the inverse matrix. Here N is the number of Markov chains and L is an estimate of the chain length in the stochastic process. These computations are independent of the matrix size n and also inherently parallel. Note that in order to find the inverse matrix or the full solution vector in the serial case, $O(nNL)$ steps are required.

For this reason we concentrate on Monte Carlo methods to solve SLAE or to find the inverse of matrices. These algorithms are further able to produce a rough solution in cases where direct or iterative methods are too costly to implement and do not provide a feasible way to find a solution.

The class of Monte Carlo algorithms in our focus have to be able to

- be scalable and fault-tolerant, and
- run efficiently on various advanced architectures

Solving systems of linear equations is a well-known problem in engineering and sciences. Using iterative or direct methods to solve these systems may be a costly approach in both time and computational effort for certain classes of problems. One option of reducing the effort of solving these systems is to apply preconditioners before using an iterative method. Depending on the method used to compute the preconditioner, the savings and end-results vary. A very sparse preconditioner is computed quickly, but it is unlikely to improve the quality of the solution. On the other hand, computing a rather dense preconditioner is computationally expensive and might be time or cost prohibitive. Therefore, finding a good preconditioner that is computationally efficient, while still providing substantial improvement to the iterative solution process, is a worthwhile research topic.

The next section gives and overview of related work. Monte Carlo methods, and the specific matrix inversion algorithm that is discussed as a SPAI preconditioner, are presented in Section 3. Section 4 provides background information on the underlying computing systems and the experimental set ups. Results and findings from experiments with matrices of varying sizes and sparsity, as well as outcomes from running the iterative solver algorithm on SPAI preconditioned matrices, are discussed in Section 5. The last section concludes and gives an outlook on the future work.

2 Related Work

Research efforts in the past have been directed towards optimizing the approach of sparse approximate inverse preconditioners. Improvements to the Frobenius norm have been proposed for example by concentrating on sparse pattern selection strategies [10], or building a symmetric preconditioner by averaging off-diagonal entries [11]. Further, it has been shown that the sparse approximate inverse preconditioning approach is also a viable course of action on large-scale dense linear systems [3]. This is of special interest to us, as the Monte Carlo code we are proposing in this paper is part of a bigger family. It includes serial and parallel Monte Carlo algorithms for the inversion of sparse, as well as dense matrices, and their solution in systems of linear algebraic equations. The proposed Monte Carlo algorithm has been developed and enhanced upon in the last decades, and several key advances in serial and parallel Monte Carlo methods for solving such problems have been made [1, 2, 8, 13, 14, 16]. There is an increased research interest in parallel Monte Carlo methods for Linear Algebra in the past year, and recent example is the Monte Carlo Synthetic Acceleration (MCSA)developed through MCREX project at ORNL[15]. Future work that deals with a parallel implementation of the presented algorithm is being considered in Section 2.

In the past there have been differing approaches and advances towards a parallelisation of the SPAI preconditioner. The method that is used to compute the preconditioner provides the opportunity to be implemented in a parallel fashion. In recent years the class of Frobenius norm minimizations that has been used in the original SPAI implementation [4] was modified and is provided in a parallel SPAI software package. One implementation of it, by the original authors of SPAI, is the Modified SParse Approximate Inverse (MSPAI [22]).

This version provides a class of modified preconditioners such as MILU (modified ILU), interface probing techniques and probing constraints to the original SPAI, apart from a more efficient, parallel Frobenius norm minimization. Further, this package also provides two novel optimization techniques. One option is using a dictionary in order to avoid redundant calculations, and to serve as a lookup table. The second option is using an option in the program to switch to a less computational intensive, sparse QR decomposition whenever possible. This optimized code runs in parallel, together with a dynamic load balancing.

Further discussion of additional advances, which are building upon the SPAI software suite, will be presented in the next section.

2.1 SParse Approximate Inverse Preconditioner (SPAI)

The SPAI algorithm [19] is used to compute a sparse approximate inverse matrix M for a given sparse input matrix B. This is done by minimizing $||BM - I||$ in the Frobenius norm. The algorithm explicitly computes the approximate inverse, which is intended to be applied as a preconditioner of an iterative method. The SPAI application provides the option to fix the sparsity pattern of the approximate inverse a priori or capture it automatically.

Since the introduction of the original SPAI in 1996, several advances, building upon the initial implementation, have been made. Two newer implementations are provided by the original authors, the before mentioned MSPAI, and the highly scalable Factorized SParse Approximate Inverse (FSPAI [21]). The intended use of both differs depending on the problem at hand.

Whereas MSPAI is used as a preconditioner for large sparse and ill-conditioned systems of linear equations, FSPAI is applicable only to symmetric positive definite systems of this kind. FSPAI is based around an inherently parallel implementation, generating the approximate inverse of the Cholesky factorization for the input matrix. MSPAI on the other hand is using

an extension of the well-known Frobenius norm minimization that has been introduced in the original SPAI.

2.2 Solving Systems of Linear Equations

For solving systems of linear equations, the SPAI application provides a preconditioner, employing minimization of the Frobenius norm, and a solver that is based on the biconjugate gradient stabilized method (BiCGSTAB). The algorithm, introduced in [24], is an iterative method to solve non-symmetric linear systems by finding a numerical solution. The BiCGSTAB solver is an extended and optimized version of the biconjugate gradient method (BiCG) [17], enabling a quicker and smoother convergence. It is one of the best known Krylov subspace methods [20].

The idea of using a Frobenius norm minimization as a direct preconditioner is based on computing a sparse approximate inverse as a matrix M, minimizing $||I - MA||$. Within this process it is possible to split the problem into n independent linear least-squares problems, with n being the number of columns of M. These problems can then be solved, for example, by using a dense QR decomposition.

The algorithm attempts to solve a system of linear equations of the form $Bx = b$ for the variable x. Its input is a sparse, square coefficient matrix B. The solution vector b can either be provided by the user, or is arbitrarily defined by the software implementation. In the case of the SPAI application suite, if no right hand side vector is handed to the algorithm, it constructs one by multiplying matrix B with a vector consisting of all ones.

In a general case, an input matrix B is passed to SPAI as a file. The program then computes a preconditioner using the Frobenius norm, afterwards it uses this intermediate result as an input to the BiCGSTAB solver.

3 Monte Carlo Approach

Monte Carlo methods are probabilistic methods, that use random numbers to either simulate a stochastic behaviour or to estimate the solution of a problem. They are good candidates for parallelisation because of the fact that many independent samples are used to estimate the solution. These samples can be calculated in parallel, thereby speeding up the solution finding process. We design and develop parallel Monte Carlo methods with the following main generic properties:

- efficient distribution of the compute data
- minimum communication during the computation
- increased precision achieved by adding extra refinement computations

Consideration of all these properties naturally leads to scalable algorithms.

3.1 Algorithm

The following procedure has been presented in [7] and allows to extend the Monte Carlo algorithm for processing diagonally dominant matrices, that is used as the foundation for this work (compare [23]), to the case of general matrices.

For this, assume the general case where $\|B\| > 1$ and consider the splitting

$$B = \hat{B} - C, \tag{1}$$

where the off-diagonal elements of $\hat{B}$ are the same as those of B, and the diagonal elements of $\hat{B}$ are defined as $\hat{b}_{ii} = b_{ii} + \alpha_i ||B||$, choosing in most cases $\alpha_i > 1$ for $i = 1, 2, ..., n$. For the simplicity of the algorithm it is often easier to fix α rather than altering it over the rows of the matrix ([6, 9, 16]).

In the general case, $\|B\| > 1$, make the initial split $B = \hat{B} - C$. From this compute $A = B_1^{-1} B_2$, which satisfies $\|A\| < 1$. Further, by careful choice, of $\hat{B}$, it is possible to make $\|A\| < \frac{1}{2}$, which gives faster convergence of the MC. Then generate the inverse of $\hat{B}$ by

$$m_{rr'}^{(-1)} \approx \frac{1}{N} \sum_{s=1}^{N} \left[\sum_{(j|s_j = r')} W_j \right], \tag{2}$$

where $(j|s_j = r')$ means that only

$$W_j = \frac{a_{rs_1} a_{s_1 s_2} \dots a_{s_{j-1} s_j}}{p_{rs_1} p_{s_1 s_2} \dots p_{s_{j-1} s_j}},$$

for which $s_j = r'$ are included in the sum (2). Calculating $\|B\|$ can be an expensive operation and, so, any a priori information allowing for a reasonable estimate here is useful. From this it is then necessary to work back and recover B^{-1} from $\hat{B}^{-1}$. To do this an iterative process $(k = n-1, n-2, \dots, 0)$ is used on $\hat{B}^{-1}$:

$$B_k^{-1} = B_{k+1}^{-1} + \frac{B_{k+1}^{-1} S_{k+1} B_{k+1}^{-1}}{1 - trace\left(B_{k+1}^{-1} S_{k+1}\right)}, \tag{3}$$

where $B_n^{-1} = \hat{B}^{-1}$ and S_i is all zero except for the $\{ii\}^{th}$ component, which is from the matrix $S = \hat{B} - B$. Then $B_0^{-1} = B^{-1}$.

The make up of matrix S means that while (3) looks complicated it is, in fact, reasonably simple. This means that it is not as computationally complex and when transferred to code there are obvious simplifications possible to make sure that many multiplications by zero are not performed. This method of splitting and recovery leads to Algorithm 1, which details a MC algorithm for inverting general matrices.

Algorithm 1. *Monte Carlo Algorithm for Inverting General Matrices*

Step 1. *Read in matrix B*

1: *Input matrix B, parameters ε and δ*

Step 1. *Calculate intermediate matrices ($\hat{B}$, B_1, B_2)*

1: *Split $B = \hat{B} - (\hat{B} - B)$, where $\hat{B}$ is a diagonally dominant matrix*

Step 1. *Apply the algorithm for inverting diagonally dominant matrices from [23] with $B = \hat{B}$ to obtain $\hat{B}^{-1}$*

Step 1. *Recovery of B^{-1} from $\hat{B}^{-1}$*

1: *Compute $S = \hat{B} - B$*

1.1: *Let S_i for $i = 1, 2, \dots, n$ where each S_i has just one of the non-zero elements of the matrix S*

1.1: *Set $B_n^{-1} = \hat{B}^{-1}$*

1.1: *Apply* $B_{i-1}^{-1} = B_i^{-1} + \frac{B_i^{-1} S_i B_i^{-1}}{1 - trace\left(B_i^{-1} S_i\right)}$ *for* $i = n, n-1, \ldots, 1$

1: *Then* $B^{-1} = B_0^{-1}$

Through the use of the enhancement the algorithm is able to generate rough inverses of input matrices efficiently. This result can then be used directly as a preconditioner for solving a system of linear algebraic equations or further improved. We propose the use of an iterative refinement process, a parallel filter, or a combination of the two to further enhance the quality of the preconditioner. The decision if those additional steps are taken is based upon the required accuracy and can be freely selected, depending on user requirements. The parameters to control the behaviour of the algorithm can be passed to the applications as command line arguments.

For the experimental setup it was necessary to modify and enhance the SPAI software application, allowing for measurements to be taken, as well as run-time information to be gathered. The existing source code was modified in order to apply the Monte Carlo generated matrix inverse as a stochastic preconditioner to the BiCGSTAB solver that is provided by SPAI.

4 Experiments

In earlier work it has been shown that our proposed Monte Carlo algorithm is well suited for calculating preconditioners for SLAE solvers. In this paper we focus on the extended version of our algorithm that works on general matrices. Several test cases from differing matrix sets have been selected and will exemplary presented to show the validity of our approach for this class of matrices. In particular we focus on non-diagonally dominant, non-symmetric matrices as input data.

We compared matrices from different sets that have been obtained from two collections - The Matrix Market [5] and The University of Florida Sparse Matrix Collection [12]. These matrices are used as inputs to both the MSPAI and our Monte Carlo based application to compute preconditioners. The results from those calculations are two intermediate matrices M_{SPAI} and M_{MCSPAI}, one for each type of preconditioner. In the last step these preconditioners are used as an input to the BiCGSTAB solver that is provided by the SPAI application.

To analyse a system of linear equation solutions, the implementation of BiCGSTAB in SPAI offers an arbitrary right hand side (RHS) vector. If no solution vector b is provided, the BiCGSTAB algorithm will use $A \times \tau$ as a RHS, where τ is a vector consisting of all ones. Due to the selection of different matrices, not all originate from systems of linear equations that provide a RHS. To provide a better reproducibility, all RHS vectors for the chosen matrices have been configured to be arbitrarily generated in this way.

Numerical experiments have been executed on the MareNostrum supercomputer, located at the Barcelona Supercomputing Center (BSC). It currently consists of 3056 compute nodes that are each equipped with 2 Intel Xeon 8-core processors, 64GB RAM and are connected via an InfiniBand FDR-10 communication network. The experiments have been run multiple times to account for possible external influences on the results. The computation times for both the preconditioner calculated by MSPAI, as well as our Monte Carlo based result, have been noted. While conducting the experiments, we configured the parameters for probable errors in both programs to produce preconditioners with similar properties and therefore producing residuals within similar ranges when used as preconditioners for BiCGSTAB. A random starting pattern has been chosen in MSPAI for best analogy to the stochastic nature of the Monte Carlo approach.

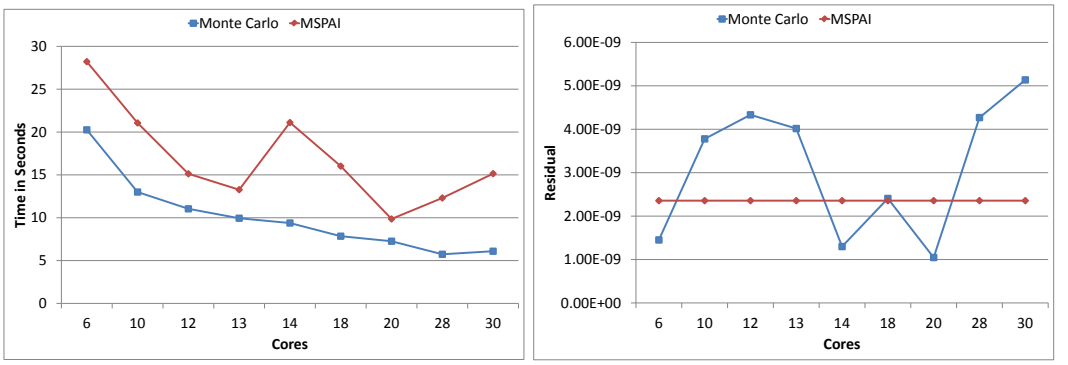

Figure 1: Run times and residuals for preconditioning psmigr_3

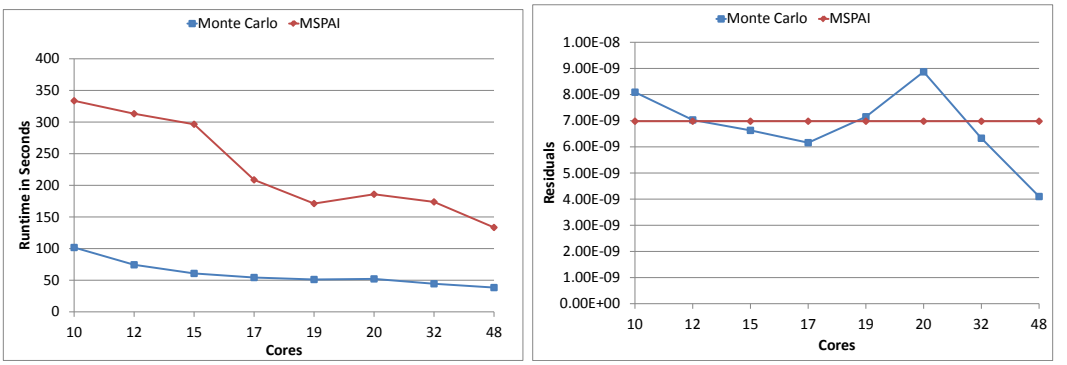

Figure 2: Run times and residuals for preconditioning Appu

The latest MSPAI application version 1.2 has been slightly modified to allow for time measurements. Also we used a modified version of SPAI 3.2 to allow for inclusion of our Monte Carlo based preconditioner. To test the quality of the computed preconditioner, the BiCGSTAB implementation in SPAI has been used to solve systems of linear algebraic equations by generating the right hand side vector from the input matrices. MSPAI has been set up with the default configuration parameters and was linked to local mathematical libraries, such as LAPACK, BLAST and ATLAS.

The matrices used as a test set are from the Psmigr set from the Harwell-Boeing Collection and the Simon group from the UF Sparse Matrix Collection. They are non-diagonally dominant, non-symmetric and the data is stored in real data type. Psmigr_3 is a $3140x3140$ matrix with 543162 non-zero entries, depicting data from US inter-county migration. Whereas Appu is a $14,000x14,000$ matrix with $1,853,104$ nonzeros from NASAs app benchmark set.

The number of compute cores used to calculate the preconditioners has been selected to match the problem size and provide meaningful comparisons between the two differing approaches. For the small test set experiments have been run on 6 to 30 cores of the compute system. The larger example was calculated from 32 up to 256 cores to show the differing scaling of the deterministic method and our stochastic algorithm.

5 Evaluation

Experiments have been run with on a selected number of square matrices of varying sizes and sparsity. The matrices have been successfully preconditioned using the enhanced version of the original SParse Approximate Inverse, MSPAI, and our proposed Monte Carlo approach. The resulting preconditioner matrices M from the computations have then been used as an input for the BiCGSTAB solver provided by SPAI. The remainder residual from the solver as well as the computation time of the preconditioners have been noted.

The resulting residuals for both the MSPAI and Monte Carlo based preconditioners are presented in Figure 1 for the Psmigr_3 matrix and Figure 2 for the Appu matrix. It is worth mentioning that the obtained values for the residuals are quite different between two compared approaches. The MSPAI based deterministic algorithm produces the same result for all computational runs whereas the Monte Carlo approach, due to its stochastic nature, generates unique preconditioners of a varying quality within the same preconfigured range every time they are recomputed.

From the results it can be seen that our proposed stochastic Monte Carlo approach is able to generate preconditioners of a good quality and within the same margin of error as the deterministic approach taken by MSPAI. When comparing the time needed to compute a preconditioner it is noticeable in the case of Psmigr_3 that both algorithms follow roughly the same scaling pattern. The Monte Carlo approach is significantly faster in computing, especially in the case of fewer processors used.

When considering the larger test case of the Appu matrix a very interesting and differing behaviour in scalability of both applications can be noticed. The Monte Carlo algorithm outperforms the MSPAI approach quite significantly. Throughout the range of researched test cases it is able to return a successfully calculated preconditioner within a shorter time frame than the MSPAI application. Both approaches make use of more available processor cores, with the deterministic MSPAI calculations taking noticeably longer to complete than our proposed stochastic approach.

During the experiments it was noted that the quality of the calculated Monte Carlo preconditioners was already high enough when parameters of the Monte Carlo preconditioner have been selected with the resulting quality of the rough inverse matrix in mind. They are therefore producing preconditioners that produce residuals in the following solving process with BiCGSTAB that are in the same order of magnitude that the results obtained with the SPAI Frobenius norm preconditioner. Due to this the additional refinement procedures, using the optional iterative filter process and parallel refinement method, could be omitted.

6 Conclusions and Future Work

An Monte Carlo based preconditioner for general matrices has been proposed as an alternative to the MSPAI algorithm and its applicability demonstrated. It has been shown that the stochastic Monte Carlo approach is able to produce preconditioners of comparable quality to the deterministic approach used in MSPAI. Our approach has been demonstrated to generate preconditioners in less time and that its scaling probabilities are outperforming the deterministic approach, especially for larger problem sizes. Our extended approach works for non-diagonally dominant, non-symmetrical sparse matrices and can also be applied to dense matrices. When compared to an improved version of SPAI, in the form of MSPAI, it has been shown that the execution time of the algorithms is quite similar for smaller problem sizes.

The algorithm we propose is able to generate good quality preconditioners by generating a

rough inverse using Markov Chain Monte Carlo methods. Advanced improvement techniques to refine the accuracy of the preconditioner even further are discussed. Through numerical experiments it has been shown that even without those optimization options the proposed algorithm is able to generate preconditioners of a high standard.

The findings and results depicted in this effort have been gathered as a trial and to prove the usefulness and validity of our approach. It has been shown that traditional methods are subject to restrictions when it comes to execution times, especially for larger problem sizes. One possible optimization has been demonstrated, using the proposed Monte Carlo based preconditioner to speed up the computation of the preconditioned matrix M. The main advantage of our method is a less costly (e.g. more computationally efficient) preconditioner that can be computed in a fraction of the time of the deterministic approach, and thus producing fast and efficient hybrid algorithms. This saving could be traded off for enhanced accuracy of the preconditioner, therefore leading to even better convergence of the algorithms.

For growing problem sizes, new restrains are foreseeable. With an increased size of the input data, handling this information in the main memory of a computer will be a challenge. To overcome these limitations, and with the inherent parallelism in Monte Carlo methods in mind, a parallel data replication approach for even larger matrices is worth investigating, yet out of the scope of this paper. Parallel approaches are usually applied for larger matrices. Due to the increasing availability of multi-core and many-core processors, as well as general purpose graphics cards (GPGPUs) with hundreds of cores each, parallel computing is becoming common practice. This also holds true for the class of work that has been presented in this paper.

Acknowledgements

This research work was partially supported by High Performance Computing VI project, number TIN2012-34557, of the Spanish Ministry of Economics and Competitiveness .

References

[1] V. Alexandrov, E. Atanassov, I. Dimov, S. Branford, A. Thandavan, and C. Weihrauch. Parallel Hybrid Monte Carlo Algorithms for Matrix Computations. In V. Sunderam, G. Albada, P. Sloot, and J. Dongarra, editors, *Computational Science ICCS 2005*, volume 3516 of *Lecture Notes in Computer Science*, pages 752–759. Springer Berlin / Heidelberg, 2005.

[2] V. Alexandrov and A. Karaivanova. Parallel Monte Carlo algorithms for sparse SLAE using MPI. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 283–290. Springer, 1999.

[3] G. Alléon, M. Benzi, and L. Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numerical Algorithms*, 16(1):1–15, 1997.

[4] M. Benzi, C. Meyer, and M. Tuma. A Sparse Approximate Inverse preconditioner for the Conjugate Gradient Method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[5] R. F. Boisvert, R. Pozo, K. A. Remington, R. F. Barrett, and J. Dongarra. Matrix market: a web resource for test matrix collections. In *Quality of Numerical Software*, pages 125–137, 1996.

[6] S. Branford. The Parallel Hybrid Monte Carlo Algoritm. Master's thesis, Schools of Systems Engineering, The Univerity of Reading, September 2003.

[7] S. Branford. *Hybrid Monte Carlo Methods for Linear Algebra Problems.* PhD thesis, School of Systems Engineering, The University of Reading, April 2009.

[8] S. Branford, C. Sahin, A. Thandavan, C. Weihrauch, V. Alexandrov, and I. Dimov. Monte Carlo Methods for Matrix Computations on the Grid. *Future Generation Computer Systems*, 24(6):605 – 612, 2008.

[9] S. Branford, C. Weihrauch, and V. Alexandrov. A Sparse Parallel Hybrid Monte Carlo Algorithm for Matrix Computations. In V. Sunderam, G. Albada, P. Sloot, and J. Dongarra, editors, *Computational Science ICCS 2005*, volume 3516 of *Lecture Notes in Computer Science*, pages 743–751. Springer Berlin / Heidelberg, 2005.

[10] B. Carpentieri, I. Duff, and L. Giraud. Some sparse pattern selection strategies for robust Frobenius norm minimization preconditioners in electromagnetism. *Numer. Linear Algebra Appl*, 7:667–685, 2000.

[11] B. Carpentieri, L. Giraud, et al. Experiments with sparse preconditioning of dense problems from electromagnetic applications. Technical report, CERFACS, Toulouse, France, 2000.

[12] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[13] I. Dimov and V. Alexandrov. A New Highly Convergent Monte Carlo Method for Matrix Computations. *Mathematics and Computers in Simulation*, 47(2-5):165–181, Aug 1998.

[14] I. Dimov, V. Alexandrov, R. Papancheva, and C. Weihrauch. Monte Carlo Numerical Treatment of Large Linear Algebra Problems. In *Lecture Notes in Computing Sciences: Computational Science - ICCS 2007*, volume 4487, pages 747–754, Berlin, 2007. Springer-Verlag GmbH.

[15] T. Evans, S. Hamilton, W. Joubert, and C. Engelmann. MCREX - Monte Carlo Resilient Exascale Project. http://www.csm.ornl.gov/newsite/documents/CSMDSummer2013Newsletter.pdf, 2013.

[16] B. Fathi Vajargah, B. Liu, and V. Alexandrov. Mixed Monte Carlo Parallel Algorithms for Matrix Computation. In *Lecture Notes in Computational Science 2330 - ICCS 2002*, pages 609–618, Berlin Heidelberg, 2002. Springer Verlag.

[17] R. Fletcher. Conjugate gradient methods for indefinite systems. *Numerical Analysis*, pages 73–89, 1976.

[18] G. Golub and C. Loan. *Matrix computations*. Johns Hopkins studies in the mathematical sciences. Johns Hopkins University Press, 1996.

[19] M. Grote and M. Hagemann. SPAI: SParse Approximate Inverse Preconditioner. *Spaidoc. pdf paper in the SPAI*, 3:1, 2006.

[20] M. Gutknecht. Variants of BiCGSTAB for Matrices with Complex Spectrum. *SIAM Journal on Scientific Computing*, 14(5):1020–1033, 1993.

[21] T. Huckle. Factorized sparse approximate inverses for preconditioning. *The Journal of Supercomputing*, 25(2):109–117, 2003.

[22] T. Huckle, A. Kallischko, A. Roy, M. Sedlacek, and T. Weinzierl. An efficient parallel implementation of the MSPAI preconditioner. *Parallel Computing*, 36(56):273 – 284, 2010. Parallel Matrix Algorithms and Applications.

[23] J. Strassburg and V. Alexandrov. On Scalability Behaviour of Monte Carlo Sparse Approximate Inverse for Matrix Computations. In *Proceedings of the Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, page 6. ACM, 2013.

[24] H. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.