Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2307 – 2316

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

A Monte Carlo Approach to Sparse Approximate Inverse Matrix Computations

J. Straßburg[a], V.N. Alexandrov[b]

[a]*The University of Reading, UK and Barcelona Supercomputing Centre, Barcelona, Spain*
[b]*ICREA Research Professor in Computational Science at Barcelona Supercomputing Centre, Barcelona, Spain*

Abstract

In this paper we present a stochastic SPAI pre-conditioner. In contrast to the standard deterministic SPAI pre-conditioners that use the Frobenius norm, we present a Monte Carlo pre-conditioner that relies on the use of Markov Chain Monte Carlo methods to compute a rough matrix inverse (MI). 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 complexity is linear to the size of the matrix. The behaviour of the proposed algorithm is studied, its performance is measured and compared with the standard deterministic SPAI approach, as well as the optimized and parallel MSPAI version. An analysis of the results is also presented.

Keywords: Monte Carlo; SPAI; MSPAI; Preconditioners; Sparse Matrix Inverse

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 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 [1].

Therefore, these algorithms often rely on pre-conditioners 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.

*Corresponding author
E-mail address: janko.strassburg@bsc.es.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.402

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 measured 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 pre-conditioners before using an iterative method. Depending on the method used to compute the pre-conditioner, the savings and end-results vary. A very sparse pre-conditioner is computed quickly, but it is unlikely to improve the quality of the solution. On the other hand, computing a rather dense pre-conditioner is computationally expensive and might be time or cost prohibitive. Therefore, finding a good pre-conditioner 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 pre-conditioner, 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 pre-conditioned matrices, are discussed in Section 5. The last section concludes and gives an outlook on future work.

2. Related Work

Research efforts in the past have been directed towards optimizing the approach of sparse approximate inverse pre-conditioners. Improvements to the Frobenius norm have been proposed for example by concentrating on sparse pattern selection strategies [2], or building a symmetric pre-conditioner by averaging off-diagonal entries [3]. 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 [4]. 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 [5–10]. 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 pre-conditioner. The method that is used to compute the pre-conditioner 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 [11] 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 [12]).

This version provides a class of modified pre-conditioners 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 to use a dictionary in order to avoid redundant calculations, and to serve as a lookup table. The second option is for 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 [13] 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 pre-conditioner 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 [14]). The intended use of both differs depending on the problem at hand.

Whereas MSPAI is used as a pre-conditioner 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 pre-conditioner, employing minimization of the Frobenius norm, and a solver that is based on the biconjugate gradient stabilized method (BiCGSTAB). The algorithm, introduced in [15], 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) [16], enabling a quicker and smoother convergence. It is one of the best known Krylov subspace methods [17].

The idea of using a Frobenius norm minimization as a direct pre-conditioner 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 pre-conditioner 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

Assume that the SLAE is presented in the standard form:

$$Bx = b, \tag{1}$$

where B is a real square $n \times n$ matrix, $x = (x_1, x_2, ..., x_n)^t$ is a $1 \times n$ solution vector, and $b = (b_1, b_2, ..., b_n)^t$.

For this first part assume that the problem matrix is diagonally dominant, i.e. $\|B\| < 1$. First, consider the splitting $B = B_1 - B_2$ where B_1 is the diagonal matrix of B, e.g. $(b_1)_{ii} = b_{ii}$ for $i = 1, 2, \ldots, n$. As shown in [18] the system (1) could be transformed to

$$x = Tx + f, \tag{2}$$

where $T = B^{-1}C$ and $f = B^{-1}b$. Now consider the possibility of finding the solution of $x = Tx + f$ using a MC method if $\|T\| < 1$ or finding B^{-1} using MC. Then, if required, obtaining the solution vector by $x = B^{-1}b$.

Consider first the stochastic approach. Assume that $\|T\| < 1$ and that the system is transformed to its iterative form (2). Consider the Markov chain given by:

$$s_0 \to s_1 \to \cdots \to s_k,$$

where the s_i, $i = 1, 2, \ldots, k$, belongs to the state space $S = \{1, 2, \ldots, n\}$. Then for $\alpha, \beta \in S$, $p_0(\alpha) = p(s_0 = \alpha)$ is the probability that the Markov chain starts at state α and $p(s_{j+1} = \beta | s_j = \alpha) = p_{\alpha\beta}$ is the transition probability from state α to state β. The set of all probabilities $p_{\alpha\beta}$ defines a transition probability matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^{n}$ ([19–21]).

The distribution $(p_1, \ldots, p_n)^t$ is called as acceptable for a given vector g, and the distribution $p_{\alpha\beta}$ is called acceptable for matrix T, if $p_\alpha > 0$ when $g_\alpha \neq 0$, and $p_\alpha \geq 0$, when $g_\alpha = 0$, and $p_{\alpha\beta} > 0$ when $t_{\alpha\beta} \neq 0$, and $p_{\alpha\beta} \geq 0$ when $t_{\alpha\beta} = 0$ respectively. It is assumed that $\sum_{\beta=1}^{n} p_{\alpha\beta} = 1$, for all $\alpha = 1, 2, \ldots, n$. Then the transition weight is defined as:

$$W_0 = 1$$

and

$$W_j = W_{j-1} \frac{t_{s_{j-1}s_j}}{p_{s_{j-1}s_j}},$$

for $j = 1, 2, \ldots, n$.

Consider now the random variable $\theta[g] = \frac{g_{s_0}}{p_{s_0}} \sum_{i=1}^{\infty} W_i f_{s_i}$. The following notation is used for the partial sum:

$$\theta_i[g] = \frac{g_{s_0}}{p_{s_0}} \sum_{j=0}^{i} W_j f_{s_j}.$$

Under the condition $\|T\| < 1$, the corresponding Neumann series converges for any given f, and $E\theta_i[g]$ tends to (g, x) as $i \to \infty$. Thus, $\theta_i[g]$ can be considered as an estimate of (g, x) for i sufficiently large. To find an arbitrary component of the solution, for example, the r^{th} component of x, it is necessary to choose, $g = e^{(r)} = (0, \ldots, \underbrace{1}_{r}, 0, \ldots, 0)$. It follows that

$$\begin{aligned}(g, x) &= \sum_{\alpha=1}^{n} (e^{(r)})_\alpha x_\alpha \\ &= x_r\end{aligned}$$

The corresponding MC method is given by:

$$\begin{aligned}x_r &= \hat{\Theta} \\ &= \frac{1}{N} \sum_{s=1}^{N} \theta_i[e^{(r)}]_s,\end{aligned}$$

where N is the number of chains and $\theta_i[e^{(r)}]_s$ is the approximate value of x_r in the s^{th} chain. It means that using an MC method makes it possible to estimate only one, few or all elements of the solution vector.

A Monte Carlo matrix inversion is obtained in a similar way [19]. To find the inverse $M^{-1} = \{m^{(-1)}_{rr'}\}^n_{r,r'=1}$ of some matrix M, it is first necessary to compute the elements of matrix $A = I - M$, where I is the identity matrix. Clearly, the inverse matrix is given by

$$M^{-1} = \sum_{i=0}^{\infty} A^i,$$

which converges if $\|A\| < 1$.

To estimate the element $m^{(-1)}_{rr'}$ of the inverse matrix M^{-1}, let the vector f be the following unit vector

$$f_{r'} = e(r').$$

Then use the following MC method for calculating elements of the inverse matrix M^{-1}:

$$m^{(-1)}_{rr'} \approx \frac{1}{N} \sum_{s=1}^{N} \left[\sum_{(j|s_j=r')} W_j \right], \tag{3}$$

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 (3).

Since W_j is included only into the corresponding sum for $r' = 1, 2, \dots, n$, then the same set of N chains can be used to compute a single row of the inverse matrix, which is one of the inherent properties of MC making them suitable for parallelisation.

The *probable error* of the method, is defined as

$$r_N = 0.6745 \sqrt{\frac{D\theta}{N}}, \text{ where } P\{\bar{\theta} - E(\theta) < r_N\} \approx \tfrac{1}{2} \approx P\{\bar{\theta} - E(\theta) > r_N\},$$

if one has N independent realisations of random variable (r.v.) θ with mathematical expectation $E\theta$ and average $\bar{\theta}$ ([22]). This description leads to Algorithm 1, which details a MC algorithm for inverting diagonally dominant matrices.

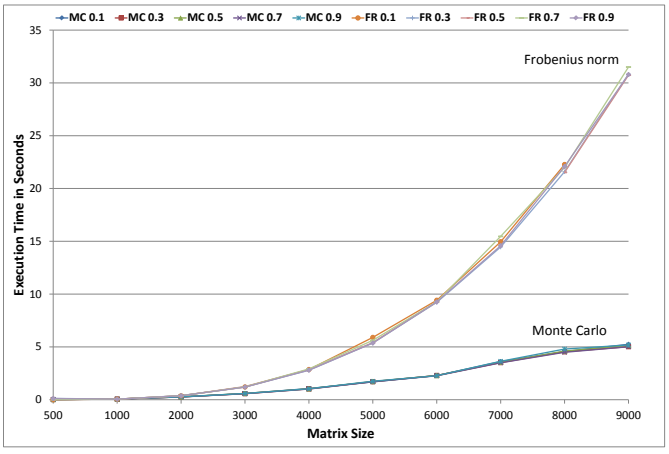


Fig. 1. Computation times Monte Carlo and Frobenius norm pre-conditioners with varying sparsity

Algorithm 1. *Monte Carlo Algorithm for inverting diagonally dominant matrices*

Step 1. *Read in matrix B*

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

Step 2. *Calculate intermediate matrices (B_1, B_2)*

1: *Split $B = B_1 - B_2$, where $B_1 = diag(B)$ and $B_2 = B_1 - B$*

Step 3. *Calculate matrix A and $||A||$*

1: *Compute the matrix $A = B_1^{-1} B_2$*

2: *Compute $||A||$ and the number of Markov chains $N = \left(\frac{0.6745}{\varepsilon(1-||A||)}\right)^2$*

Step 4. *Calculate matrix P*

1: *Compute the probability matrix, P*

Step 5. *Calculate matrix M, by MC on A and P*

1: *For $i = 1$ to n*

1.1: *For $j = 1$ to N*

Markov chain Monte Carlo Computation

1.1.1: *Set $W_0 = 1$, $point = i$, and $SUM[k] = \begin{cases} 1 \text{ if } i = k \\ 0 \text{ if } i \neq k \end{cases}$*

1.1.2: *Select a nextpoint, based on transition probabilities in P, so that $A[point][nextpoint] \neq 0$*

1.1.3: *Compute $W_j = W_{j-1} \frac{A[point][nextpoint]}{P[point][nextpoint]}$*

1.1.4: *Set $SUM[nextpoint] = SUM[nextpoint] + W_j$*

1.1.5: *If $|W_j| > \delta$ set $point = nextpoint$ and goto 1.1.2*

1.2: *Then $m_{ik} = \frac{SUM[k]}{N}$ for $k = 1, 2, \ldots, n$*

Step 6. *Calculate B^{-1}*

1: *Compute the Monte Carlo inverse $B^{-1} = MB_1^{-1}$*

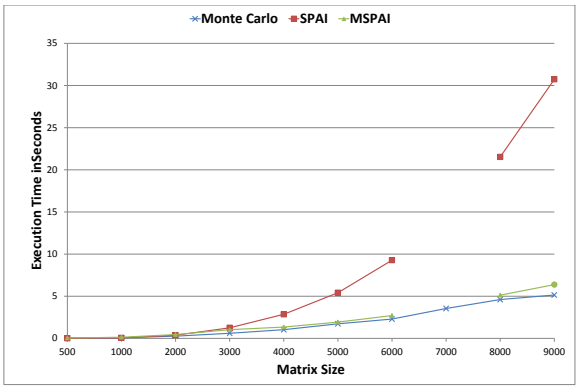
This algorithm for diagonally dominant matrices has been extended with some additional transformations to support the inversion of general sparse matrices. For this work, we concentrate on the sparse matrix version of Algorithm 1, modified for general matrices. Further, we are only interested in the rough Monte Carlo inverse that it computes. To be able to get measurements and run-time information, the SPAI software application had to be enhanced. We modified the existing source code, thus enabling us to use our Monte Carlo generated inverse as a stochastic pre-conditioner for the BiCGSTAB solver.

4. Experiments

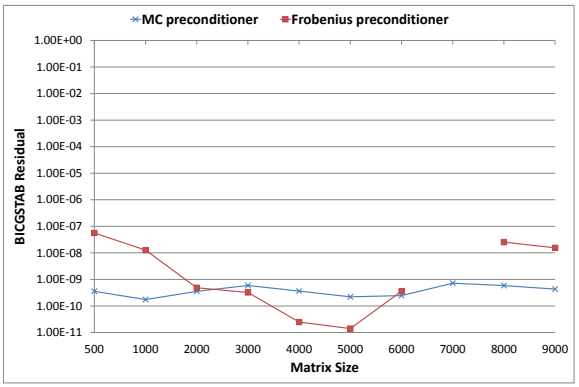
For a comparison of the proposed Monte Carlo solver and the standard SPAI pre-conditioner, relying on the Frobenius norm of the input matrix, several test runs have been executed. To be able to check the run times of both pre-conditioners and the respective results when run through the BiCGSTAB[15] solver, modifications have been made to the original SPAI application. Further tests have been conducted to investigate the behaviour of the enhanced MSPAI implementation when compared to both the original SPAI and our proposed algorithm.

All experiments have been executed on a machine with an Intel Core¢ i7-640M Processor running with 2.80 GHz and 8GB or RAM. The pre-conditioners and solvers in serial mode have been exclusively run on one core of the CPU to avoid influencing the measures by system activity and other processes. Parallel runs where executed by utilizing all four cores of the system. The latest SPAI application version 3.2 has been slightly modified to allow for time measurements and inclusion of our Monte Carlo based pre-conditioner. To test the quality of the computed pre-conditioner, 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.

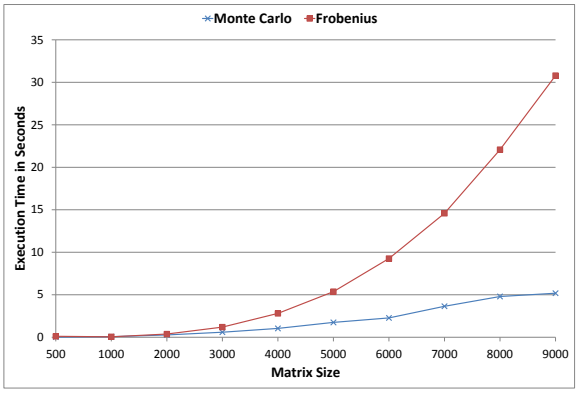
The test data consists of randomly generated squared non-singular matrices of sizes 500-30.000 and varying sparsity levels from 0.1 to 0.9. These matrices are used as inputs to both the pre-conditioner in SPAI, finding the Frobenius norm, and our Monte Carlo method, generating a rough approximate inverse of the matrix. The same



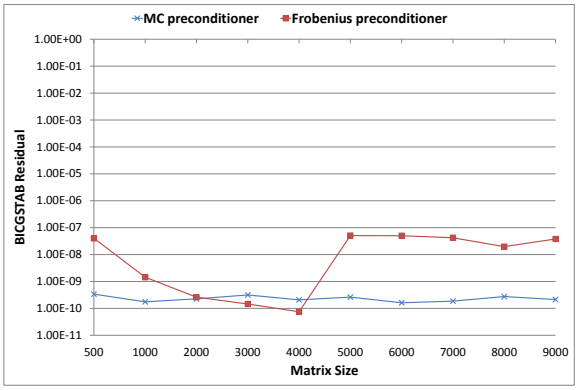
(a) Execution times 0.5



(b) Residuals 0.5



(c) Execution times 0.9



(d) Residuals 0.9

Fig. 2. Run times and residuals for matrices with varying sparsity

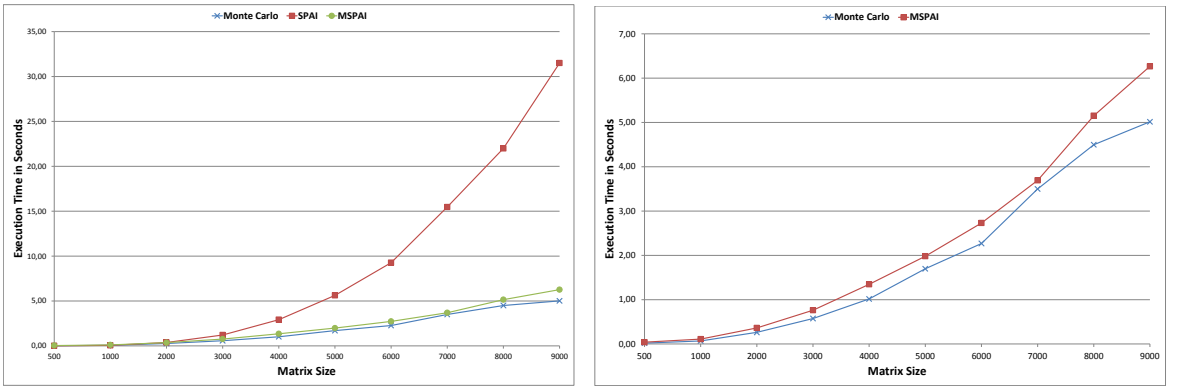
matrices are also the base for comparisons with the MSPAI algorithm, generating a pre-conditioner matrix. These computations provide two preconditioned intermediate matrices M, one for each approach.

For comparison and testing purposes the BiCGSTAB algorithm is then employed, using the pre-conditioned matrices M. To analyse a system of linear equation solutions, the implementation of BiCGSTAB in the SPAI application toolbox offers two methods. It either expects an optional dense right hand side (RHS) vector b as an input alongside the sparse matrix A, or it creates an arbitrary RHS. 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.

Additional experiments have been conducted to further study the scaling properties of two selected algorithms. From initial results it was obvious that the original SPAI algorithm is too slow to obtain a meaningful comparison with the quicker Monte Carlo approach. The improved MSPAI version however provides some code optimizations that allow for comparable computation times. Due to this reason we decided to compare the performances of MSPAI and our proposed Monte Carlo algorithm on larger sized square matrices of sizes 15.000 to 30.000.

5. Evaluation

Experiments have been run with a pre-defined number of square matrices of varying sizes and sparsity. Both serial as well as parallel executions of the MSPAI and the proposed Monte Carlo based algorithm have been carried out. The matrices have been successfully pre-conditioned using the SPAI algorithm with Frobenius norm minimizing pre-conditioner, its enhanced version MSPAI, and our proposed Monte Carlo approach. The resulting pre-conditioner matrices M from the SPAI and Monte Carlo runs have then been used as an input for the BiCGSTAB solver.



(a) SPAI, MSPAI, Monte Carlo

(b) MSPAI, Monte Carlo

Fig. 3. Execution times for varying matrix sizes with sparsity 0.7

The computation times for both the Monte Carlo and SPAI using Frobenius norm pre-conditioners are given in Figure 1. It is noteworthy that the development of the execution times is quite different between those two approaches. While the Frobenius norm approach displays an exponential increase in run time for larger problem sizes, the curve for the Monte Carlo pre-conditioner is ascending noticeably slower. Further, when it comes to absolute timings, the Monte Carlo approach is significantly quicker to calculate the pre-conditioner than the approach used in the SPAI software suite, showing almost linear behaviour. This is especially evident for matrix sizes starting from $2000x2000$. For smaller matrices both algorithms perform equally well, although it has to be kept in mind that the total runtime for these smaller systems lies only in the magnitude of tens and hundreds of milliseconds on the test machine. These results are less significant, since there is a dis-balance between the actual computations and the operating systems background noise, that influences the timings. Examples are data transfers between the hard disk and the main memory, task switches between processes and caching features. The Monte Carlo approach constantly outperforms the Frobenius norm approach, in the largest test cases that have been considered, it is about six times faster than the latter algorithm. The parameters of the Monte Carlo pre-conditioner have been selected with the resulting quality of the rough inverse matrix in mind. They are therefore producing pre-conditioners that produce residuals in the following solving process with BiCGSTAB, that are in the same order of magnitude as the results obtained with the SPAI Frobenius norm pre-conditioner.

In two cases the pre-conditioners computed by the SPAI algorithm did not converge during the BiCGSTAB solving process. The pre-conditioners calculated by our proposed Monte Carlo algorithm on the other hand did not exhibit this kind of behaviour and converged in all test cases. Results for different sparse matrices can be found in Figure 2. After additional experiments, including the enhanced MSPAI version, we could observe the same behaviour. As can be seen in Figure 2(a), both the SPAI as well as the MSPAI algorithm do not converge during the computation of the pre-conditioner and therefore do not complete their calculations.

To compare the scaling behaviour of the MSPAI pre-conditioner and our proposed Monte Carlo approach, several experiments with larger matrices have been carried out. First computational results for a subset of comparison options have been gathered and are looking promising. Due to time restraints, only preliminary parallel experiments could be run and analysed. This part of the work is currently ongoing.

Further development and enhancement of SPAI has led to an improved and optimized version that also natively supports parallel computational environments. These efforts have been presented in the MSPAI software suite. Its runtime behaviour has been significantly improved upon and it is today a better and faster choice for computing sparse pre-conditioners, when compared with the original SPAI version. As can be seen from Figure 3 its behaviour is approximate to that of our proposed Monte Carlo algorithm. The computational time needed to generate a quick, sparse approximate inverse of the input matrix lies within similar ranges for both algorithms.

This version of the MSPAI algorithm has, however, a similar peculiarity to the original SPAI software. This class of algorithm does not converge for certain matrices, therefore not being able to generate a pre-conditioner. Figure 2(a) shows the identical behaviour of these two algorithms, not constructing a pre-conditioner for a

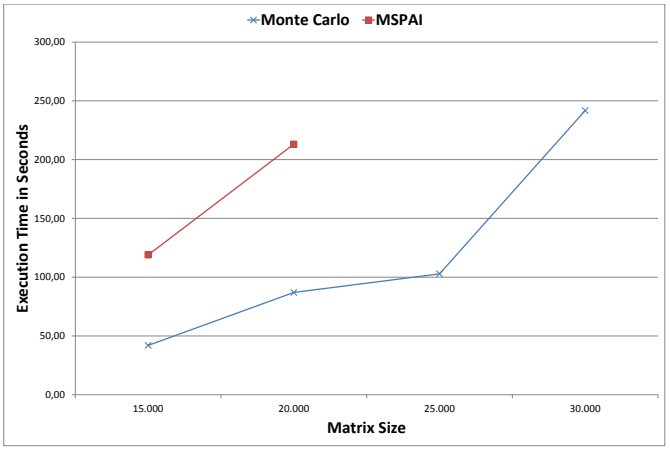


Fig. 4. Run times for MSPAI and Monte Carlo applications

7000x7000 input matrix. Our proposed algorithm, on the other hand, does not show this kind of behaviour. Applying it to the problem matrix, we were able to successfully compute a pre-conditioner. This behaviour has been noticed during the experimental phase, the Monte Carlo based algorithm returned results in the few cases where the SPAI algorithms failed.

The overall run times of both the MSPAI application and our proposed Monte Carlo approach are given in Figure 4. As mentioned before, the SPAI family of algorithms is exhibiting problematic behaviour for some matrix sizes. Apart from not converging for one particular smaller matrix of size 7000 and sparsity 0.5, the algorithm does also not converge for larger test matrices. It can be seen in the figure that the MSPAI application was not able to successfully compute a pre-conditioner for square matrices larger than size 25.000. Completed experiments using our Monte Carlo approach have been given as a comparison. In the case of MSPAI producing a pre-conditioner, the Monte Carlo application is showing a faster program runtime. For the two given cases, the Monte Carlo application is approximately twice faster than the MSPAI. Analysis has shown that the way MSPAI handles the input data is mainly responsible for the quite significant difference in overall application run time.

6. Conclusions and Future Work

A Monte Carlo based pre-conditioner as an improved alternative to the Frobenius norm minimization used in the SPAI software suite has been proposed and its value demonstrated. It has been shown that the Monte Carlo approach is able to produce pre-conditioners of comparable quality of the solution to the Frobenius norm approach in a fraction of the execution time. Further it has been discovered, that our proposed algorithm is able to generate pre-conditioners that are ensuring convergence of the BiCGSTAB solver in cases where the SPAI Frobenius norm based algorithm fails. Our approach also works for non-diagonally dominant sparse matrices, as well as 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. A big difference is noticeable in overall program runtime though. This is due to the way MSPAI handles the input data, it is much slower than our proposed algorithm. This holds especially true for matrices of bigger dimensions. Further, the MSPAI algorithm did not produce results for certain smaller matrices and also has problems with large input matrices above a certain size. The Monte Carlo approach presented in this paper on the other hand is able to compute a sparse approximate inverse even in those cases.

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. 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 lager problem sizes. One possible optimization has been demonstrated, using

the proposed Monte Carlo based pre-conditioner to speed up the computation of the pre-conditioned matrix M. 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 approach for even larger matrices is worth investigating, yet out of the scope of this paper.

Further experiments with matrices of differing structures, sizes and sparsity, as well as an enhancement to the algorithm with focus on increased accuracy of the calculations are promising future research topics. The main advantage of our approach is a less costly pre-conditioner that can be computed in a fraction of the time of the original SPAI Frobenius norm approach, and thus producing fast and efficient hybrid algorithms. This saving could be traded off for enhanced accuracy of the pre-conditioner, therefore leading to even better convergence of the algorithms.

It has been laid out earlier in this section that a parallel version of the shown approach is deemed to be promising research topic. We are currently experimenting with a parallel version of this hybrid Monte Carlo algorithm in combination with a parallel BiCGSTAB implementation. Preliminary results are promising and further effort is going to be focused on this study.

References

[1] G. Golub, C. Loan, Matrix computations, Johns Hopkins studies in the mathematical sciences, Johns Hopkins University Press, 1996. URL: http://books.google.es/books?id=mlOa7wPX6OYC.

[2] B. Carpentieri, I. Duff, L. Giraud, Some sparse pattern selection strategies for robust frobenius norm minimization preconditioners in electromagnetism, Numer. Linear Algebra Appl 7 (2000) 667–685.

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

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

[5] I. Dimov, T. Dimov, T. Gurov, A new iterative Monte Carlo approach for inverse matrix problem, Journal of Computational and Applied Mathematics 92 (1998) 15–35.

[6] I. Dimov, V. Alexandrov, A new highly convergent Monte Carlo method for matrix computations, Mathematics and Computers in Simulation 47 (1998) 165–181.

[7] S. Branford, C. Weihrauch, V. Alexandrov, A sparse parallel hybrid Monte Carlo algorithm for matrix computations, in: V. Sunderam, G. Albada, P. Sloot, J. Dongarra (Eds.), Computational Science ICCS 2005, volume 3516 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2005, pp. 743–751.

[8] V. Alexandrov, E. Atanassov, I. Dimov, S. Branford, A. Thandavan, C. Weihrauch, Parallel hybrid Monte Carlo algorithms for matrix computations, in: V. Sunderam, G. Albada, P. Sloot, J. Dongarra (Eds.), Computational Science ICCS 2005, volume 3516 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2005, pp. 752–759.

[9] C. Weihrauch, I. Dimov, S. Branford, V. Alexandrov, Comparison of the computational cost of a Monte Carlo and deterministic algorithm for computing bilinear forms of matrix powers, in: Computational Science - ICCS 2006, volume 3993, Springer-Verlag, 2006, pp. 640–647.

[10] I. Dimov, V. Alexandrov, R. Papancheva, C. Weihrauch, Monte Carlo numerical treatment of large linear algebra problems, in: Lecture Notes in Computing Sciences: Computational Science - ICCS 2007, volume 4487, Springer-Verlag GmbH, Berlin, 2007, pp. 747–754.

[11] M. Benzi, C. Meyer, M. Tuma, A sparse approximate inverse preconditioner for the conjugate gradient method, SIAM Journal on Scientific Computing 17 (1996) 1135–1149.

[12] T. Huckle, A. Kallischko, A. Roy, M. Sedlacek, T. Weinzierl, An efficient parallel implementation of the mspai preconditioner, Parallel Computing 36 (2010) 273 – 284. ¡ce:title¿Parallel Matrix Algorithms and Applications¡/ce:title¿.

[13] M. Grote, M. Hagemann, Spai: Sparse approximate inverse preconditioner, Spaidoc. pdf paper in the SPAI 3 (2006) 1.

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

[15] 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 (1992) 631–644.

[16] R. Fletcher, Conjugate gradient methods for indefinite systems, Numerical Analysis (1976) 73–89.

[17] M. Gutknecht, Variants of bicgstab for matrices with complex spectrum, SIAM Journal on Scientific Computing 14 (1993) 1020–1033.

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

[19] V. Alexandrov, Efficient Parallel Monte Carlo Methods for Matrix Computation, Mathematics and Computers in Simulation 47 (1998) 113–122.

[20] V. Alexandrov, A. Karaivanova, Parallel Monte Carlo Algorithms for Sparse SLAE Using MPI, in: LNCS 1697, Springer, 1999, pp. 283–290.

[21] V. Alexandrov, A. Rau-Chaplin, F. Dehne, K. Taft, Efficient Coarse Grain Monte Carlo Algorithms for Matrix Computation Using PVM, in: LNCS 1497, Springer, 1998, pp. 323–330.

[22] I. M. Sobol, Monte Carlo Numerical Methods, Nauka, Moscow, 1973. (in Russian).