Frequency Interpolation Methods for Accelerating Parallel EMC Analysis

K. Homma
Secure Computing Laboratory, Computer System Laboratories, Fujitsu Laboratories Ltd
homma@flab.fujitsu.co.jp

K. Nagase
Technology Development Group, Fujitsu Ltd
knagase@cs.fujitsu.co.jp

M. Noro
Secure Computing Laboratory, Computer System Laboratories, Fujitsu Laboratories Ltd
noryo@flab.fujitsu.co.jp

P.E. Strazdins
Department of Computer Science, Australian National University
peter@cs.anu.edu.au

T. Yamagajo
Technology Development Group, Fujitsu Ltd
gajo@cs.fujitsu.co.jp

Abstract

Electro-magnetic field analysis applications based on the Method of Moments can be used to simulate the emissions for electrical devices such as a printed circuit board, a combination of circuit boards and wire connecting boards, or even a cabinet.

At the heart of such applications is a solver, which solves a symmetric indefinite dense linear system of size N assembled from the model of the electrical device.

The main computational challenges lie in the solver stage, where an $O(N^3)$ computation is required for the direct solution of the linear system about a central frequency ω_c. For the direct solution we use a general symmetric matrix factorization algorithm, requiring $N^3/3 + O(N^2)$ FLOPs. This algorithm's efficiency is demonstrated by its parallel speedup of 5 for moderate sized matrices on an 8 node AP3000.

Some of this cost can be amortized using the fast frequency stepping method*, where the system can be solved for nearby frequencies ω by $O(N^2)$ iterative methods, using the solution at ω_c as a preconditioner.*

Due to the high parallel efficiency of the direct method, the frequency stepping method reduced parallel solution time by a factor of 2 for moderate-sized matrices, with larger improvements expected for large matrices.

1 Introduction

The design of electric devices is severely limited by the country-specific Electromagnetic Compatibility (EMC) requirements. Hence, minimizing the undesired radiation and avoiding malfunctions caused by intruding electromagnetic radiation is required in the design of these devices. Recently, the CPU operating speed has become faster, resulting in higher emission frequencies. In addition, the electric circuits become smaller and more highly integrated. Thus, the undesired electromagnetic wave radiation from these devices tends to increase. In such a situation, the estimation of EMC via simulation becomes more and more important.

ACCUFIELD is a 3D simulator currently being developed by Fujitsu Ltd. It simulates the electromagnetic wave radiation and the immunity of many kinds of electrical devices by using the moment method. ACCUFIELD has been used mainly to simulate relatively simple electric devices, such as printed circuit boards.

Such analysis can demonstrate mechanisms such as the shielding effectiveness of enclosures, the influence of finite ground plates, common-mode and differential mode current

0-7695-0990-8/01/$10.00 (C) 2001 IEEE

emissions and the effectiveness of components such a filters in handling EM emissions.

ACCUFIELD calculates the electromagnetic field in the frequency domain, and so, a matrix calculation, being a dense complex symmetric linear system solve, is executed repeatedly. When a large scale complicated model of an electric device is being analysed, the computing time becomes long, as a direct solution requires $O(N^3)$ floating point operations. As such an EMC analysis is normally required for several frequencies, a series of linear system solutions is required.

So, the reduction of this execution time is desired to estimate the EMC phenomenae of such devices efficiently.

As well as providing an efficiently parallelizable direct solver for ACCUFIELD, we have also parallelized an improvement of the frequency interpolation method called *fast frequency stepping* (FFS), which uses iterative methods to analyse the system at intermediate frequencies more efficiently.

Frequency interpolation methods have been developed for the EMC analysis, where the Method of Moments generates non-symmetric systems [8, 4]. The fast frequency stepping method itself was introduced by Hoyler and Unbehauen [8], giving favorable comparisons of FFS with other preconditioning techniques (eg. Block-Jacobi and GMRES preconditioners). The direct solution is used only at lowest frequency and all further linear systems is solved iteratively using this single preconditioner. More recently, FFS has been encapsulated in a more general EM framework called Model-based Parameter Estimation (MBPE) [12], where it has been shown how interpolation can reduce computational work on both the linear system assembly and solution.

Parallel iterative solutions to methods for EMC have been reported in [10], where a Conjugate-Gradient Algorithm with Block-Jacobi preconditioner was used. However, here again non-symmetric solution methods were used; also this work did not use frequency interpolation.

The main original contributions of this paper is to adapt the FFS method for general symmetric preconditioners and systems. It is also to show how to improve FFS convergence rate while economizing on the number of direct solutions by using preconditioners based on a central frequency. It is also to show how FFS can be parallelized for efficient EMC analysis with multiple frequencies.

This paper is organized as follows. Section 2 explains the moment method used by ACCUFIELD, and how it generates symmetric linear systems. The generation of the corresponding matrices is described in Section 3. Section 4 describes the direct solution methods required by the solver stage, with some detail in the components also used by the *fast frequency stepping method*. Section 5 outlines the fast frequency stepping method, which can reduce the number of direct system solutions for analysis with multiple frequencies, and describes its parallel implementation. Performance results are given in Section 6, with conclusions being given in Section 7.

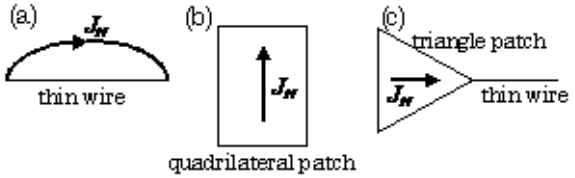

Figure 1. Expansion functions for (a) thin wire mode, (b) surface patch mode and (c) attachment mode.

2 The Moment Method

ACCUFIELD calculates the electromagnetic field of the wave radiated from the model electric device by using the moment method [7]. Here, we describe this method briefly.

The material of the analysed model is the metal or dielectric with or without loss. The analyzed model is segmented to triangular or rectangular patch elements or thin wire elements [13]. The surface current on the metal J_c^s and dielectric J_d^s and the surface magnetic current M^s on the dielectric are expanded in terms of the basis expansion functions $J_{c,n}$, $J_{d,n}$ and K_n as follows:

$$J_c^s = \sum_{n=1}^{N_c} I_{c,n} J_{c,n},\ J_d^s = \sum_{n=1}^{N_d} I_{d,n} J_{d,n},\ M^s = \sum_{n=1}^{N_d} M_n K_n \tag{1}$$

Here, $I_{c,1:N_c}$, $I_{d,1:N_d}$ and $M_{1:N_d}$ are the $N = N_c + 2N_d$ unknown coefficients. The expansion function is specified in three types i.e., thin wire mode, patch mode and attachment mode applied to wire/surface junctions. These expansion functions are shown in Figure 1.

From the surface equivalence theorem and the reaction matching moment method, the following equation is obtained:

$$\begin{bmatrix} Z^0_{c,c} & Z^0_{c,d} & B^0_{c,d} \\ Z^0_{d,c} & Z^0_{d,d}+Z^d_{d,d} & B^0_{d,d}+B^d_{d,d} \\ B^0_{d,c} & B^0_{d,d}+B^d_{d,d} & Y^d_{d,d}-Y^0_{d,d} \end{bmatrix} \begin{bmatrix} I_c \\ I_d \\ M \end{bmatrix} = \begin{bmatrix} V_i \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

The matrix of the left side of Equation 2 is an immittance matrix and each element of it represents the mutual impedance Z, the mutual admittance Y and the reaction

B between two elements. Here, superscripts 0 and d indicate the field material and corresponds to the air and the dielectric, respectively. Subscripts d and c indicate the material of the segmented elements and correspond to dielectric and conductor, respectively. The associated interactions are symmetric, so that, for example, $B^0_{c,d} = (B^0_{d,c})^T$. Explicit equations of the immittance are shown in [13, 11]. $I_{c,1:N_c}$, $I_{d,1:N_d}$ and $M_{1:N_d}$ are the unknown expansion coefficients which appear in Equation 1. V_i is the driver voltage. From Equation 2, the current distribution of the model surface and the electromagnetic field of the radiated wave are calculated.

For convenience, we will use $Z(\omega)I = V(\omega)$ as an abbreviated form of Equation 2, where ω is the frequency of the incident electric field $V(\omega)$.

3 Matrix Generation

The matrix generation requires the calculation of the electromagnetic field generated by the model with the moment method described before. The transmission line approximation method is also employed to take into account the effects of components or subsystems beyond the moment method [13, 5].

Consider the parallel implementation on a $P \times Q$ grid of processors of an ACCUFIELD analysis over a range of frequencies $\Omega = \{\omega_1, \omega_2, \ldots, \omega_{n_f}\}$, where $\omega_i < \omega_{i+1}$ for $1 \leq i < n_f$.

Firstly, the (required parts of) model's information is broadcasts to all processors. MPI is used as the communication library, enabling wide portability of the application.

The immittance matrices $Z(\omega_{s_1}), \ldots, Z(\omega_{s_{n_s}})$ are then generated, where $\{\omega_{s_1}, \ldots, \omega_{s_{n_s}}\} \subseteq \Omega$. Here $\{\omega_{s_1}, \ldots, \omega_{s_{n_s}}\}$ (typically $s_1 = 1, s_{n_s} = n_f$, and n_s is small) is a set of sampling frequencies chosen for the analysis.

As the elements of these matrices can be generated independently, this process is done in parallel so that the matrices conform to a $r \times r$ *block-cyclic matrix distribution* over a logical $P \times Q$ processor grid [2], where r is the fixed storage block size.

The remaining matrices $Z(\omega_i)$ are generated by interpolation on the sampled immittance matrices $Z(\omega_{s_1}), \ldots, Z(\omega_{s_{n_s}})$. As they will have the same distribution, this step requires no communication.

It should be noted that at any one time, there must be sufficient storage for three matrices, one being the current $Z(\omega_i)$ being interpolated, and two being the matrices used for interpolation. For large N, this is a drawback of this approach.

At this point, the solver stage can begin on each $Z(\omega_i)$. In the following two sections, we describe in detail issues in the parallelization of these stages.

The parallel routines used are coded entirely in terms of the DBLAS Distributed BLAS Library [14, 16], which is a portable version of parallel BLAS. It has been used to implement very efficient parallel matrix factorization applications using various techniques [15]. The use of the DBLAS has enabled high reliability and performance from its highly tested and optimized components.

4 The Direct Solution of Symmetric Indefinite Systems

The direct solution uses a parallelized version of the Bunch-Kaufman algorithm [3] to perform the factorization $A = PLDL^TP^T$, where L is an $N \times N$ lower triangular matrix with a unit diagonal, and D is a block diagonal matrix with either 1×1 or 2×2 sub-blocks, and P is a permutation matrix.

The Bunch-Kaufman algorithm was chosen because its implementation in LAPACK is one of the most competitive such algorithms [1], and because, compared with tridiagonal methods, it results in a reduced number of symmetric interchanges. For parallel implementation, it has been argued that such a property is very important for minimizing the high communication costs associated with these interchanges [18]; these can be further reduced by the recent introduction of a variant of the Bunch-Kaufman algorithm [17].

The direct solution also requires a back-solve stage; the corresponding parallel routine is called `pzsytrs()`. As this routine is also required by FFS, we shall give some details here.

While the back-solve stage has only $O(N^2)$ floating point operations, it has high associated overheads which makes its optimization particularly important in the distributed memory context.

As the factorization stage also applies P^{-1} fully in L during the factorization, unlike in the LAPACK implementation of the Bunch-Kaufman algorithm. For a right-hand side vector X, the back-solve stage can then be expressed as:

$$X \leftarrow P^{-1}X \; ; \; X \leftarrow L^{-1}X \; ; \; X \leftarrow D^{-1}X \; ;$$
$$X \leftarrow L^{-T}X \; ; \; X \leftarrow P^{-T}X$$

The DBLAS parallel triangular solve routine, performing $X \leftarrow L^{-1}X$ and $X \leftarrow L^{-T}X$, does most of the work of the back-solve. As it is a standard routine, it is already highly optimized. It communicates only X and so minimizes communication volume costs, while maximizing load balance. It also blocks communications of X by the storage block size r, to reduce startup costs.

Furthermore, it has the following computational optimizations: it implements blocking of the computations (in

the case of multiple RHS, most of the work is done in level 3, rather than level 2, BLAS), and it is matrix-vector multiply based (faster on most cell architectures, including the UltraSPARC, than rank-1 updates.

As the factorization stage saves D^{-1} in the form of row-replicated vectors, no further communications are required for performing $X \leftarrow D^{-1}X$.

The reduction in the number of interchanges in the factor stage similarly reduces communication startup costs in the application of $X \leftarrow P'^{-1}X$ and $X \leftarrow P'^{-T}X$.

The routine performing the factorization and the back-solve is called `pzsysv()`.

Where possible a square or near-square processor grid is preferred for the dominant matrix factorization stage, this minimizes communication volume costs and improves cache utilization. Also, for such symmetric computations, it also minimizes startup costs and improves load balance [18].

5 The Fast Frequency Stepping Method

ACCUFIELD often calculates field emissions over a range of EM wave frequencies. To avoid the cost of directly solving Equation 2 at each desired frequency in the range, the following can be performed:

- instead of obtaining a direct solution of $Z(\omega_i)I = V(\omega_i)$ for each ω_i, the fast frequency stepping method (FFS) chooses a central frequency $\omega_{i_c}, 1 \leq i_c \leq n_f$.

  For ω_{i_c}, a direct solution is obtained.

- the factorization of $Z(\omega_{i_c})$ is then used to precondition iterative solutions for $\omega_{i_c-1}, \omega_{i_c-2}, \ldots$ and $\omega_{i_c+1}, \omega_{i_c+2}, \ldots$ until convergence rates slows down to the extent that the iterative solution time exceeds half that of a direct solution.

Here, the Preconditioned Conjugate Gradients method is used [6, 8]. For a positive definite matrix A, the preconditioning is $A \rightarrow (L_0^T)^{-1}AL_0^{-1} \; : A_0 = L_0^T L_0$ where L_0 is the Cholesky factorization of the preconditioner A_0. This algorithm converges in $rank(I - (L_0^T)^{-1}AL_0^{-1})$ iterations [6]; however, in practice this quantity is difficult to estimate.

This method can be applied to frequency interpolation as follows. With sufficiently close frequencies ω_1 and ω_2, then the immittance matrices $Z_1 = Z(\omega_1)$ and $Z_2 = Z(\omega_2)$ may also be similar. Hence, if $L_1^T L_1 = Z_1$ is the Cholesky factorization of Z_1, $L_1^{-T} Z_2 L_1^{-1}$ may be close to the identity matrix. So we may solve the equation $Z_2 J = F$ by the preconditioned CG method (with preconditioner L_1) with very few iterations.

We have adopted this method for complex symmetric indefinite matrices by using the LDLT factorization for the preconditioner ie. $A_0 = Z(\omega_{i_c}) = P_0 L_0 D_0 L_0^T P_0^T$. Figure 2 describes the algorithm.

$r_0 = b - Ax_0$

$p_0 = A_0^{-1} r_0, \quad k = 0$

while $\|r_k\| \geq \epsilon$

$\alpha_k = (A_0^{-1} r_k, r_k)/(p_k, Ap_k)$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$\beta_k = (A_0^{-1} r_{k+1}, r_{k+1})/(A_0^{-1} r_k, r_k)$

$p_{k+1} = A_0^{-1} r_{k+1} + \beta_k p_k$

$k = k + 1$

Figure 2. Preconditioned Congugate Gradients for Complex Symmetric Equation $Ax = b$.

Thus, the FFS method reduces the number of $O(N^3)$ direct solutions potentially required, replacing them by $O(N^2)$ iterative methods, resulting in potentially large gains in efficiency.

The iterative method for FFS is implemented using the DBLAS parallel BLAS (see Section 4) routines for symmetric matrix-vector multiply `pzsymv()`, vector inner product and norm routines, as well as the complex indefinite solve routine `pzsytrs()` described in Section 4. The above routines have a compatible interface to ScaLAPACK's PBLAS [2].

The exploitation of symmetry here necessarily means an increase in communication costs, as the computation of Ap_k must be done by two triangular matrix vector multiplies.

To economize on storage space, the factored preconditioner L_0 and D_0 are stored in the lower half of the matrix, and the current matrix $A = Z(\omega_i)$ is stored in the upper half (with one column shift).

When executed on a $P \times Q$ processor grid, each iteration involves a vector transpose-broadcast, one vector reduction and two scalar reduction/broadcasts, as well as the communications required in the call to `pzsytrs()`. For large matrices, eg. $\frac{N}{P} \geq 1000$, the AP3000 execution time is dominated by the level-2 computational performance (matrix-vector multiply); for smaller matrices, execution time is dominated by communication startup costs.

6 Performance

This section investigates the performance of the analysis stage of ACCUFIELD, where computation time is dominated by the solver stage.

The results here are for a Fujitsu AP3000 [9] comprised of 300 MHz UltraSPARC processors with 2 GB memory. This machine has communication networks with characteristics shared by most other state-of-the-art distributed memory computers, that is, high communication costs relative to floating point speed, and row or column broadcasts having to be simulated by point-to-point messages. For the AP-Net, the AP3000's communication network, communication latencies (from software) has been measures at $\approx 20\mu s$, with a transfer rate of up to 80 MB/s sustained for large messages.

The electromagnetic wave radiated from the Note PC model is simulated as an example of the analysis of the electric devices at the design stage. The model is shown in Figure 3. In this analysis, a matrix equation with $N = 4608$ is calculated. The dependence of the computing time on the node number of AP3000 is shown in Figure 1; at 8 nodes, the speedup is approximately 5.0.

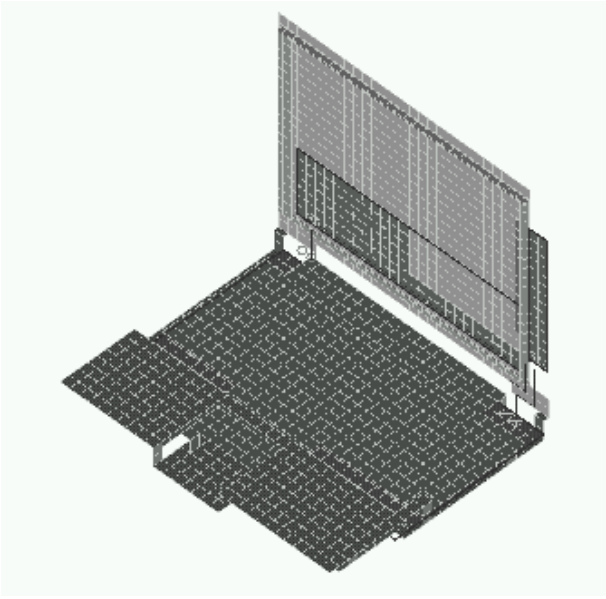

Figure 3. Note PC model. The model consists of 2706 patch elements and 33 wire elements. Analyzed frequency is 66.66MHz.

node:	1	24	8
times (s):	293	103	59

Table 1. The dependence of the current calculation time on the node number of AP3000 for the Note PC model.

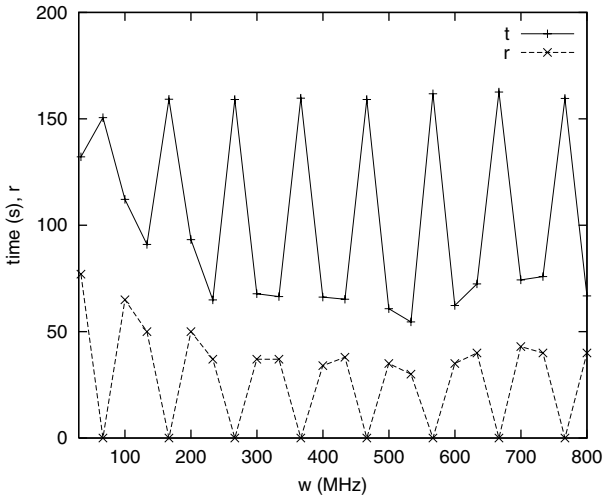

Figure 4. FFS solution time t and number of iterations r for each frequency for a POS terminal on an 8 node 360MHz AP3000.

In the result shown in Table 1, the current is calculated for the single frequency (66.66 MHz). In the actual design stage, the current is often calculated repeatedly for wide-range of frequencies as was described in the introduction. In case of the Note PC model shown in Figure 3, the calculation of the current requires few or more hours. If `pzsysv()` is utilized, the current computing time is found to be reduced to few or few ten minutes and the electronic device such as the Note PC can be designed more efficiently.

However, for analysis with multiple frequencies, further speedups are desirable, which can be afforded by the fast frequency stepping method.

As mentioned in Section 5, we heuristically stop the iterative solving with a LDLT preconditioner if its computing time exceeds the half of the time of a direct solve. In the examples below executed in parallel, this resulted in the number of iterative solves per central frequency being at most two, which are for ω_{i_c-1} and ω_{i_c+1}, where a direct solution is used at a central frequency ω_{i_c}.

Figure 4 shows the execution times and the number of iterations r of parallel FFS for a POS terminal, generating a linear system of size $N = 7017$. The frequencies are in the range 30 to 800 MHz. Note that the convergence is slow at lower frequencies, which is a common phenomenon of FFS. A value of $r = 0$ indicates a direct solution was obtained for this frequency. The averaged iterative solution time is 66.5s, and the average direct time is 160.3s; thus the application of FFS (over using a direct solution at each frequency) results in a speedup of 1.65.

Figure 5 shows the execution times of parallel FFS for a PenNote PC. Here the matrix size $N = 4608$. The set

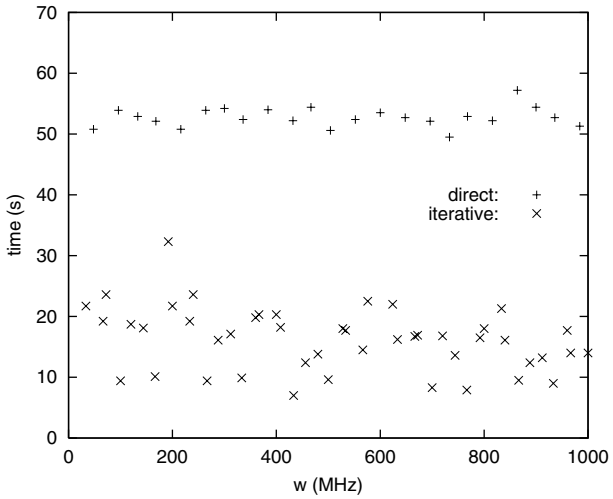

Figure 5. FFS solution time for each frequency for a PenNote PC on an 8 node 360MHz AP3000.

of frequencies is a union of two sequences $\{100n/3\}$ and $\{24n\}$, $n = 1, 2, 3, \ldots$. So some adjacent frequencies are close to each other and FFS is particularly efficient for such pairs, as more rapid convergence occurs ($r \approx 10$ for such pairs, half of the average value for this model). Here, the averaged iterative solution time is 16.2s, whereas that of the direct solution is 57.7s. Hence, due to a faster average convergence rate, a higher speedup of FFS of 2.03 results.

When applying FFS to a sequential computation, we observed that a single preconditioner covers wider range of frequencies. For example, the PenNote PC analysis of Figure 5 required only four direct solvings when sequential FFS was applied. Compared with the sequential case the speedup by the parallel FFS is not as large.

The reason for this is that the parallel speedup of level 2 functions such as `pzsymv()` and `pzsytrs()` is not as high as for level 3 computations. Over 95% of the execution time was spent in `pzsymv()` and `pzsytrs()`. Of this, between 60% and 70% was spent in `pzsytrs()`, which (even in the absence of interchanges) requires at least $12N$ communication startups on an 8 node machine.

However, for larger N, this speedup of `pzsytrs()` will improve; this, together with the $O(N)$ reduction of floating point operations enabled by FFS, means that better improvements for FFS are expected for larger models.

7 Conclusions

EMC applications such as ACCUFIELD can accurately analyse electromagnetic fields emitted by electronic devices, using a combination of the moment method and the transmission line approximation method. However, for complex electronic devices, the resulting models can be very large (up to tens of thousands of elements), requiring huge computational and memory resources for the results to be generated in an acceptable time. Distributed memory parallel computing offers a cost-effective and scalable way of providing these resources.

In this paper, we have shown how ACCUFIELD can be efficiently parallelized on a distribed memory multicomputer. The parallelization occurs mainly over the compute-intensive solve stage, which involves the solution of complex dense symmetric indefinite linear systems.

The direct solution must be efficiently parallelized; we have chosen the Bunch-Kaufman LDLT factorization algorithm for this purpose. It was particularly important to improve performance in the back-solve part of this computation, as this is also used by the iterative solver.

To obtain further speedup, we have introduced an alternate approach for the case of EMC analysis over multiple frequencies. The FFS method employed the iterative conjugate gradient method using the direct solution over a central frequency as a preconditioner. While it achieved a time reduction by a factor of only ≈ 2 for parallel solution on moderate sized systems, much greater reductions are expected for larger systems.

The exploitation of symmetry can be used to partially offset the extra memory requirements of the fast frequency stepping method, with both the preconditioner and the current linear system both being contained in an $N \times (N+1)$ area of memory. On the other hand, it also adds extra communication overheads and slightly reduced computational speed, compared with using a general matrix decomposition (eg. LU).

Future work includes investigating further methods of improving the parallel performance for the iterative solver. Applying FFS simultaneously over several (say 4) vectors could amortize the communication costs, as well as result in increased computational speed.

References

[1] C. Anderson and J. Dongarra. Evaluating Block Algorithm Variants in LAPACK. In *Fourth SIAM Conference for Parallel Processing for Scientific Computing*, Chicago, Dec. 1989. 6 pages.

[2] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, J. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK: A Linear Algebra Library for Message Passing Computers. In *SIAM Conference on Parallel Processing*, March 1997.

[3] J. R. Bunch and L. Kaufman. Some Stable Methods for Calculating Inertia and Solving Symmetric Linear Systems. *Mathematics of Computation*, 31(137):163–79, Jan. 1977.

[4] G. Burger, H.-D. Bruns, and H. Singer. Advanced Method of Moments Based on Iterative Equation Sstem Solvers. In *IEEE Symposium of EMC*, pages 236–241, Austin, 1997. IEEE Press.

[5] S. O. et al. EMC Analysis of Radiation and Immunity by Moment Method utilizing Frequency Characteristic of Mutual Impedance. In *EMC'96: The International Conference on Electromagnetic Compatibility*, Roma, Sept. 1996.

[6] G. Golub and C. V. Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, second edition, 1989.

[7] R. Harrington. Moment Method of Field Problems. *Proceedings of the IEEE*, Feb. 1967.

[8] G. Hoyler and R.Unbehauen. An efficient alogrithm for the treatment of multiple frequencies with the method of moments. In *EMC'96: The International Conference on Electromagnetic Compatibility*, Roma, Sept. 1996.

[9] H. Ishihata, M. Takahashi, and H. Sato. Hardware of the AP3000 Parallel Server. *Fujitsu Scientific and Technical Journal*, 33(1):24–29, 1997.

[10] U. Jakobus. Computation of electromagnetic fields by the method of moments on the CRAY T3E: Iterative solution techniques and large scale applications. In *High-Performance Computing in Science and Engineering: Transactions of the High Performance Computing Center Stuttgart (HLRS) 1999*, pages 413–423, Berlin, 2000. Springer Verlag.

[11] A. Kishk and L. Shafai. The Effect of Various Parameters of Circular Microstrip Antennas and their Radiation Efficiency and the Mode Excitation. *IEEE Transactions on Antenna and Propagation*, Aug. 1986.

[12] E. Miller. Model-based parametrer estimation in electromagnetics. Part III: Application to EM inegral equations. *IEEE Antennas and Propagation Magazine*, pages 49–66, June 1998.

[13] E. Newman. Electromagnetic Modelling of Wire and Surface Geometries. *IEEE Transactions on Antenna and Propagation*, Nov. 1978.

[14] P. Strazdins. Reducing Software Overheads in Parallel Linear Algebra Libraries. In *The 4th Annual Australasian Conference on Parallel And Real-Time Systems*, pages 73–84, Newcastle Australia, September 1997. Springer.

[15] P. Strazdins. Lookahead and Algorithmic Blocking Techniques Compared for Parallel Matrix Factorization. In *PDCN'98: 10th International Conference on Parallel and Distributed Computing and Systems*, pages 291–297, Las Vegas, September 1998. IASTED.

[16] P. E. Strazdins. Transporting Distributed BLAS to the Fujitsu AP3000 and VPP-300. In *Proceedings of the Eighth Parallel Computing Workshop*, pages 69–76, Singapore, Sept. 1998. School of Computing, National University of Singapore. paper P1-E.

[17] P. E. Strazdins. Accelerated Methods for Performing the LDLT Decomposition. In *Porceedings of CTAC'99: The 9th Biennial Computational Techniques and Applications Conference and Workshops*, Canberra, September 1999.

[18] P. E. Strazdins. Parallelizing Dense Symmetric Indefinite Solvers. In *PART'99: The 6th Annual Australasian Conference on Parallel And Real-Time Systems*, pages 398–410, Melbourne, Nov. 1999. Springer-Verlag.