SLAC-PUB-10320
January 2004
Revised May 2004

Solving Large Sparse Linear Systems in End-to-end Accelerator Structure Simulations*

Lie-Quan Lee, Lixin Ge, Marc Kowalski, Zenghai Li, Cho-Kuen Ng, Greg Schussman, Michael Wolf, and Kwok Ko

Stanford Linear Accelerator Center, Menlo Parck, CA 94025

Abstract

This paper presents a case study of solving very large sparse linear systems in end-to-end accelerator structure simulations. Both direct solvers and iterative solvers are investigated. A parallel multilevel preconditioner based on hierarchical finite element basis functions is considered and has been implemented to accelerate the convergence of iterative solvers. A linear system with matrix size **93,147,736** and with 3,964,961,944 non-zeros from 3D electromagnetic finite element discretization has been solved in less than 8 minutes with 1024 CPUs on the NERSC IBM SP. The resource utilization as well as the application performance for these solvers is discussed.

Contributed to the 18th International Parallel and Distributed Processing Symposium, Santa Fe, NM, USA, 4/26/2004 – 4/30/2004

*Work supported by Department of Energy contract DE-AC03-76SF00515

Solving Large Sparse Linear Systems in End-to-end Accelerator Structure Simulations

Lie-Quan Lee, Lixin Ge, Marc Kowalski, Zenghai Li, Cho-Kuen Ng,
Greg Schussman, Michael Wolf, and Kwok Ko
Stanford Linear Accelerator Center, Menlo Park, CA 94025
{liequan,lge,mkowalsk,lizh,cho,schussman,mwolf,kwok}@slac.stanford.edu

Abstract

This paper presents a case study of solving very large sparse linear systems in end-to-end accelerator structure simulations. Both direct solvers and iterative solvers are investigated. A parallel multilevel preconditioner based on hierarchical finite element basis functions is considered and has been implemented to accelerate the convergence of iterative solvers. A linear system with matrix size **93,147,736** *and with 3,964,961,944 non-zeros from 3D electromagnetic finite element discretization has been solved in less than 8 minutes with 1024 CPUs on the NERSC IBM SP. The resource utilization as well as the application performance for these solvers is discussed.*

1 Introduction

Parallel computing has dramatically advanced the role of computer modeling in the design of high-energy particle accelerators, especially in the research and development of accelerating structures. New numerical tools have been shown to be capable of modeling realistic accelerator components to an unprecedented level of complexity, speed, and accuracy (orders of magnitude beyond what was previously possible). There are parallel electromagnetic system simulation(ESS) codes, **Omega3P**, **S3P**, and **T3P**, under development at Stanford Linear Accelerator Center (SLAC) as part of the United States Department of Energy SciDAC project "Advanced Computing for 21st Century Accelerator Science and Technology".

The suite of parallel three-dimensional ESS codes consists of **Omega3P**, an eigensolver for computing resonant modes in accelerator cavities, **S3P**, a scattering matrix solver for open structures, and **T3P**, a time-domain solver for wave propagation and beam excitation. All three codes require an efficient linear solver for solving the very large sparse linear systems resulting from three-dimensional finite element discretization of complex accelerator structures. State-of-the-art parallel direct solver packages such as WSMP [8], SuperLU [11], MUMPS [2], and SPOOLES [4] have demonstrated that a high level of performance can be achieved in solving sparse linear systems on parallel computers.

On the other hand, iterative linear solvers usually require less memory so they are preferred choices for very large sparse linear systems that cannot fit into memory using direct solvers. Due to the fact that linear systems from our applications are highly indefinite, the Conjugate Gradient (CG) method with incomplete factorization-based preconditioners fails to converge. CG has a very slow convergence rate with symmetric

Gauss-Seidel preconditioners. In this paper, we describe a multilevel preconditioner that leads to significant improvement in the convergence rate of iterative solvers. In Particular, a case study is presented on the utilization of both direct solvers and iterative solvers in large scale electromagnetic simulations, with special emphasis on memory utilization and parallel performance when applied to extremely large linear systems derived from realistic modeling of accelerating structures.

2 Electromagnetic Modeling

2.1 Eigenmode Analysis

In designing accelerating cavities, evaluating cavity resonances is extremely important. In a perfectly conducting cavity Ω, Maxwell's equations in the frequency domain can be combined to give the curl-curl equation in volume Ω with an electric boundary condition on Γ_1 and a magnetic boundary condition on Γ_2.

$$\nabla \times (\frac{1}{\mu}\nabla\times \vec{\mathbf{E}}) - \omega^2\epsilon\, \vec{\mathbf{E}} = 0 \quad in\,\Omega \quad (1)$$

$$\vec{n} \times \vec{\mathbf{E}} = 0 \quad on\,\Gamma_1 \quad (2)$$

$$\vec{n} \times(\nabla\times \vec{\mathbf{E}}) = 0 \quad on\,\Gamma_2 \quad (3)$$

where $\vec{\mathbf{E}}$ is the electric field, ϵ is electric permittivity, μ is magnetic permeability, $\vec{n}$ is the normal vector to a boundary surface, and ω is the eigenfrequency to be determined. With finite element discretization, the electric field $\vec{\mathbf{E}}$ is expanded as:

$$\vec{\mathbf{E}} = \sum e_i\, \vec{\mathbf{N}}_i \quad (4)$$

where $\vec{\mathbf{N}}_i$ is the vector finite element basis function. Equation 1 becomes a generalized eigenvalue problem:

$$\mathbf{K}\mathbf{x} = \frac{\omega^2}{c^2}\mathbf{M}\mathbf{x} \quad (5)$$

$$\mathbf{K}_{i,j} = \int \frac{1}{\mu}(\nabla\times \vec{\mathbf{N}}_i)\cdot(\nabla\times \vec{\mathbf{N}}_j)d\Omega \quad (6)$$

$$\mathbf{M}_{i,j} = \int \epsilon\, \vec{\mathbf{N}}_i \cdot \vec{\mathbf{N}}_j\, d\Omega \quad (7)$$

where $\mathbf{K}$ and $\mathbf{M}$ are the stiffness and mass matrices with c being the speed of light.

Note that $\mathbf{K}$ is symmetric while $\mathbf{M}$ is symmetric positive definite. In end-to-end accelerator system simulations, the size of theses matrices can be huge, reaching hundreds of millions of degrees of freedom. In such cases, they are too large to be solved with direct solvers due to memory constraints and must be solved with iterative solvers. We have developed a parallel eigensolver, **Omega3P** [16, 15], which has been shown successful in solving large, complex accelerating cavity design problems. **Omega3P** uses the shift-and-invert Lanczos algorithm whereby in each Lanczos iteration, we solve a severely ill-conditioned shifted linear system $(\mathbf{K} - \sigma\mathbf{M})\mathbf{x} = \mathbf{b}$ with σ being the shift. The choice of an efficient linear solver is critical to the performance of the eigensolver.

2.2 Scattering Matrix Calculation

While acceleration of particles is carried out by resonance (standing wave) cavities, the transport of wave power is provided by open (traveling wave) structures. For these electromagnetic structures, the transmission properties are of interest and they are characterized by the scattering matrix (or S-parameters). The elements of the scattering matrix are the transmission or reflection coefficients of a particular wave mode at a structure opening or port due to coupling to an external waveguide. The finite element formulation of the scattering matrix problem closely follows that of the eigenvalue problem in Section 2.1. The difference here is there is an additional driving term which is provided by a known excitation at a specific port with a given frequency.

To compute the scattering matrix at a given operating frequency ω, we solve a series of linear systems with different right hand sides:

$$\mathbf{K} - \frac{\omega^2}{c^2}\mathbf{M}\mathbf{x} = \mathbf{b} \qquad (8)$$

where stiffness matrix **K** and mass matrix **M** correspond to those in eigenmode calculation of the system without ports. Each right hand side $\mathbf{b}$ represents the driving term due to the excitation at the ports. **S3P** is a parallel finite element code that contains a set of linear solvers that are specifically optimized to solve Equation 8 for finding the scattering matrix of very large and complex traveling wave structures.

2.3 Time-domain Simulation

Electromagnetic analysis can be performed in the frequency domain via eigenmode and scattering matrix calculations, or it can be carried out in the time domain through direct simulation. For accelerator applications, electromagnetic time-domain simulation will find the response of a structure to driven fields at waveguide ports, to an antenna-generated pulse inside it, or to a transient beam through the structure. The wakefields generated by beam excitations are of particular interest. These wakefields are challenging to calculate in the frequency domain because impedance spectrum of the structures spans a wide frequency range when the wakefields are excited by a very small beam. For this case, time-domain simulation following the transient response of the structure to the beam is more suitable because the wakefields are obtained directly and the impedance spectrum can then be found simply via a Fourier transform.

The time domain formulation of Maxwell's equations for an electric field leads to a second order vector wave equation that becomes a set of ordinary differential equations (ODE) when discretized with finite element basis functions. Here are the ODEs:

$$\mathbf{M}\frac{d^2\mathbf{u}}{dt^2} + \mathbf{T}\frac{d\mathbf{u}}{dt} + \mathbf{K}\mathbf{u} = \mathbf{f} \qquad (9)$$

where K is the stiffness matrix, M is the mass matrix as those in the above eigenmode analysis. T is a time-independent damping matrix due to loss. The right hand side, $\mathbf{f}$, is a loading vector.

Using an implicit time stepping scheme, Equation 9 results in a linear system that has to be solved at every time-step for the excitation given by the right hand side. For wakefield calculations involving a small beam, the simulation time is very long so as to adequately resolve the impedance spectrum, thus requiring millions of time steps. **T3P** is a parallel time-domain solver under development based on the formulation as described here to compliment the frequency domain solvers **Omega3P** and **S3P**. These three solvers form a complete tool set for analyzing accelerator structures. It is evident that efficient linear solvers play an important role in the performance of these application codes.

3 Linear Solvers

The computational time spent in the linear solvers dominates the run-time for all three of the electromagnetic codes Omega3P, S3P and T3P. Improving the performance of the linear solver will not only reduce the computational time in simulating realistic accelerator structures, but may enable the modeling of larger, more complicated structures at a much higher accuracy than what was previously possible. Next, we will discuss the use of direct and iterative solvers in large-scale electromagnetic simulations.

3.1 Direct Solvers

Given sufficient memory and reasonable performance, direct solvers would be the method of choice for solving sparse linear systems of equations. A typical direct solver comprises ordering for reducing fill-in, symbolic factorization, numerical factorization, a triangular solver, and iterative refinement. An efficient direct solver greatly helps the performance of applications be-

cause the linear systems with same matrix and different right hand sides need to be solved up to a million times as is the case in many accelerator applications. With a direct solver, a matrix needs to be ordered and factorized only once. After that, triangular solvers along with optional iterative refinement can be invoked multiple times for different right hand sides. State-of-the-art parallel direct solver packages such as SuperLU, WSMP, SPOOLES, and MUMPS have demonstrated that a high level of performance can be achieved on parallel computers. We use WSMP in our applications because we found it to be the most efficient among the solver packages.

3.2 Iterative Methods

Krylov subspace iterative linear solvers such as Conjugate Gradient (CG) require less memory than direct solvers do. They are preferred for very large sparse linear systems that cannot fit into memory using direct solvers. The linear systems resulting from finite element discretization in accelerator applications are ill-conditioned and good preconditioners are needed in order for the iterative solver to converge. As a result, a multilevel preconditioner based on hierarchical finite element basis functions has been implemented to significantly improve the convergence rate of CG solvers.

3.3 Multilevel Preconditioner

Finite element basis functions are hierarchical when the basis functions of a given order are a subset of the basis functions for a higher order. We used the hierarchical vector bases presented in [14]. Since each matrix row and column index in stiffness matrix $\mathbf{K}$ and mass matrix $\mathbf{M}$ corresponds to one particular basis function (or degree of freedom) as defined in Equation 4, we can order degrees of freedom (DOF) for lower order basis functions before those for higher order basis functions. For simplicity, assume that quadratic hierarchical finite element basis functions are used in applications. It is straightforward to apply the following discussion to higher order basis functions. We order DOFs such that matrix $\mathbf{K}$ and $\mathbf{M}$ have a 2-by-2 block structure:

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix}$$

Let us denote $\mathbf{A} = \mathbf{K} - \sigma\mathbf{M}$. Thus, the linear system $(\mathbf{K} - \sigma\mathbf{M})x = b$ can be written in the same 2-by-2 block form:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \qquad (10)$$

A good preconditioner for the above linear system has to be a good approximation of the matrix $\mathbf{A}$. In addition, the preconditioner system must be solvable in much less time than the original linear system. Multilevel preconditioners [6, 14, 3] based on hierarchical basis functions use the solution from the matrices that correspond to lower order basis functions to approximate the original linear system. For example, a block Jacobi stationary method can be used as follows:

$$x_1 = \mathbf{A}_{11}^{-1} b_1 \qquad (11)$$

$$x_2 = \mathbf{A}_{22}^{-1} b_2 \qquad (12)$$

where sub-matrix $\mathbf{A}_{11}$, which corresponds to the lower order finite element basis function, is factorized using parallel sparse direct solvers. Thus, Equation 11 is solved using triangular solvers with factorized matrices. In contrast, Equation 12 is solved approximately using diagonal scaling or Symmetric Gauss-Seidel iterations.

In the above block Jacobi method, the unknowns for higher order basis functions are not coupled with those of lower order basis functions. The preconditioning scheme can be improved by coupling the unknowns through the block symmetric Gauss-Seidel stationary method,

$$y_1 = \mathbf{A}_{11}^{-1} b_1 \quad (13)$$

$$x_2 = \mathbf{A}_{22}^{-1}(b_2 - \mathbf{A}_{21} y_1) \quad (14)$$

$$x_1 = \mathbf{A}_{11}^{-1}(b_1 - \mathbf{A}_{12} x_2) \quad (15)$$

By directly factorizing $\mathbf{A}_{11}$, Equation 13 and 15 are solved using triangular solvers with factorized matrices, whereas diagonal scaling or Symmetric Gauss-Seidel is used to approximate Equation 14.

3.4 Linear Solver Framework

We have implemented a linear solver framework in which either the direct solver from the Watson Sparse Matrix Package (WSMP) [7] or Krylov subspace methods from the Iterative Template Library (ITL) [10] can be used for solving large sparse linear systems.

WSMP is a general purpose library for the direct factorization of large sparse matrices of linear equations. For symmetric linear systems, WSMP provides a highly scalable multi-frontal algorithm [9] for sparse Cholesky (or LDL^T) factorization, a moderately scalable parallel sparse triangular solver, and a nested dissection algorithm for computing fill-in reduction ordering. WSMP uses both Message Passing Interface (MPI) and multi-threading programming models. MPI is used for cross network communication while the Pthreads library is used to exploit more efficient parallelism within multiprocessor computing nodes.

The Iterative Template Library (ITL) is a generic iterative linear solver library written in C++ with emphasis on both reusability and efficiency. Algorithms in ITL are written to be independent of matrix, vector, or preconditioner data structures. As long as a specific data structure conforms to a set of minimal requirements defined by each concept [5], ITL algorithms can be used with that data structure. ITL includes a set of generic Krylov subspace iterative methods, a set of preconditioners, and a set of interfaces to use different linear algebra packages including the Matrix Template Library [13], Blitz++ [17], A++/P++ [12], and Fortran BLAS. It has been demonstrated that scientific applications with ITL can achieve high performance on parallel computers [10]. We implemented the multilevel preconditioner using the same generic design so that we used it with the Conjugate Gradient algorithm and the parallel linear algebra interface in ITL to have a new parallel hybrid sparse linear solver.

4 Results and Discussions

All computations were performed on NERSC's IBM SP computer, a distributed memory machine with 6,080 Power3 375MHz processors. The processors, each with a peak performance of 1.5 gigaflops, are distributed among 380 computing nodes, each comprise of 16 processors. Each node has between 16 and 64 gigabytes of memory.

The target application is the scattering matrix calculation for an **entire** accelerating section shown in Figure 1. Under consideration as the base line design for the Next Linear Collider, this 55-cell tapered structure, named H60VG3, has varying dimensions from cell-to-cell to provide detuning for higher-order modes and consists of two dual-feed couplers for power input and output respectively. Scattering parameters are evaluated at the input/output ports by solving the linear system described by Equation 8 in Section 2.2.

4.1 Direct and Iterative Solver Comparison

The sample calculation is for a frequency of 31.44 GHz to find the reflections and transmissions of a set of higher order modes. Table 1 compares the execution time and memory usage between the Conjugate Gradient with Symmetric Gauss-Seidel (SGS) preconditioner and the direct solver WSMP. Both solvers used 64 CPUs on linear systems consisting of 1.3 million degrees of freedom. The result shows that in terms of speed, WSMP was more than two orders of mag-

Figure 1. Model of the detuned structure, H60VG3, which is the baseline design of Next Linear Collider [1]. It consists of 53 cells and two couplers. Each cell has different dimensions.

nitude faster than Conjugate Gradient with 4 iterations of SGS. WSMP, however, required six times as much memory. The comparison indicates that given sufficient physical memory, direct solver WSMP can deliver much higher performance than iterative solvers do and should be the method of choice in our applications.

Table 1. Execution time and memory usage comparison between Conjugate Gradient with Symmetric Gauss-Seidel (SGS) preconditioner and WSMP in simulating H60VG3 structure with 1.3 million degrees of freedom. The execution time is for solving 16 right hand sides. 64 CPUs were used.

	CD with SGS(4)	WSMP
Time (s)	14582.1	82.6
Memory (GB)	2.9	19

4.2 Scalability Studies for WSMP

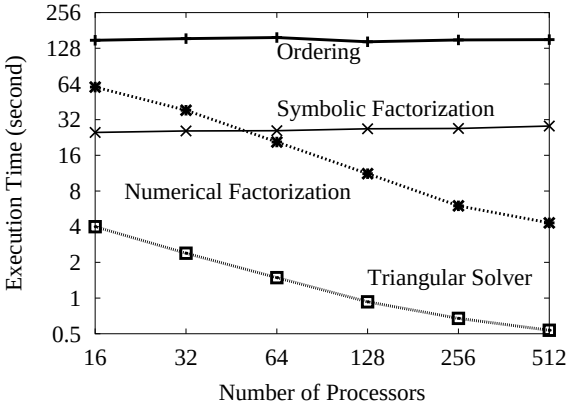
We examined the scalability of execution time and memory usage of WSMP direct solver by performing computations first with a fixed number of DOFs, which was set at 3.1 million. Figure 2(a) shows execution time versus number of processors with the results broken down into time spent in four steps: ordering, symbolic factorization, numerical factorization, and triangular solution. The ordering and symbolic factorization steps show little or no speedup while the numerical factorization and triangular solver scale very well with increase in processors. Figure 2(b) plots the aggregate memory used which shows a slow increase with increase in CPUs.

From these results, one can make the following observation. For a given problem size with multiple right hand sides, WSMP requires a fairly constant computation overhead due to ordering and symbolic factorization even as the number of CPUs is increased. However, numerical factorization and triangular solution (which have good scalability) are executed multiple times as long as the non-zero structure of the matrix does not change. As the overhead becomes a smaller fraction of the total computation when the number of right hand sides becomes large, WSMP presents a very scalable linear solver that is very attractive for our applications.

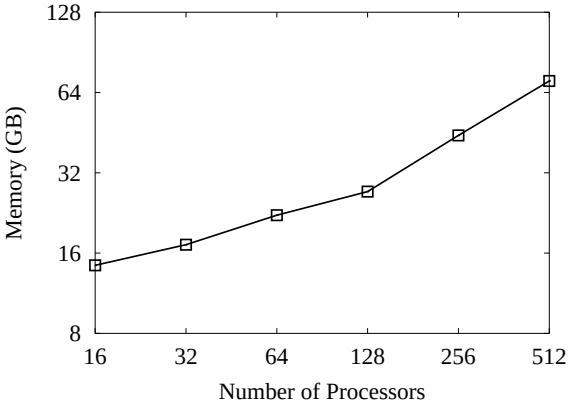
Next we fix the number of CPUs but increase the problem size or the number of DOFs by generating meshes of decreasing mesh size for the same physical model. Figures 3(a) and 3(b) show the execution time and memory usage versus DOFs using 16 CPUs, respectively. In both plots we use line fitting to find the complexity. In Figure 3(a) the dependence of ordering plus factorization and triangular solve are $O(N^{1.44})$ and $O(N^{1.20})$. Figure 3(b) shows an almost linear increase in memory usage, $O(N^{1.14})$, which is very promising since a faster increase would severely hamper WSMP's use in solving much larger systems due to memory limitations.

4.3 Multilevel Preconditioner Performance

We have implemented the newly-developed multilevel preconditioner (as described in Section 3.3) for the Conjugate Gradient algorithm

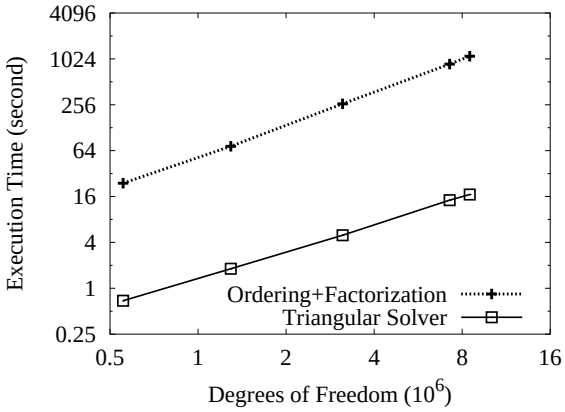


(a) Execution time

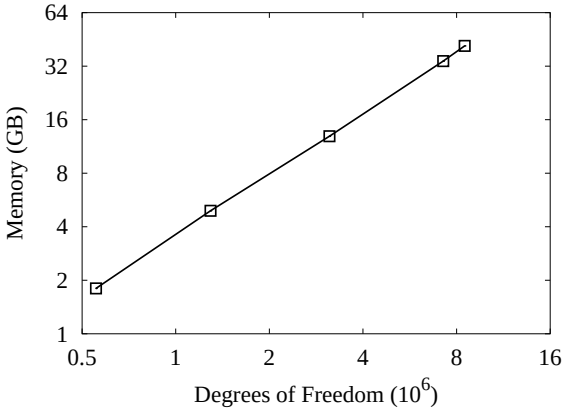


(b) Memory usage

Figure 2. Execution time and memory usage for solving a linear system with 3.1 million degrees of freedom that correspond to 48.8 million non-zeros in the matrix.



(a) Execution time



(b) Memory usage

Figure 3. Execution time and memory usage versus DOFs with 16 CPUs.

from ITL and applied it successfully to a linear system with a matrix size of **93,147,736**. We used block Jacobi stationary method in multilevel preconditioner. The first level given by Equation 11 is solved with WSMP while the next level given by Equation 12 is solved approximately using 2 Symmetric Gauss-Seidel iterations. The multilevel preconditioner is found to be very efficient in reducing the iteration count in the CG solver. In solving 14 different right hand sides, the average number of CG iterations is around 50.

Table 2 lists the computing resources used in solving the 93 million DOF linear system with CG and the Multilevel Preconditioner. The amortized solving time per right hand side for that 93 million DOF linear system is less than 8 minutes. As a comparison, the requirements for WSMP to solve a 30 million DOF system are also shown. It would not have been possible for WSMP to solve the larger system of 90 million plus DOFs due to memory limitation. The memory usage would be over 1.5 Terabytes based on the near linear scaling derived above. CG with the Multilevel Preconditioner shows superior performance both in terms of memory requirement and execution time. This new powerful solver enables large, complex structures to be modeled with much higher resolution than any existing codes.

4.4 Wakefields in the H60VG3 Structure

End-to-end modeling of the H60VG3 structure has been carried out using **Omega3P** to calculate the eigenmodes in the lowest dipole band. This system-scale study has been made possible by the integration of the Multilevel Preconditioner into **Omega3P**. As seen in Figure 4(a) the mode spectrum is tightly clustered so that a very large number of DOFs is needed to resolve the small mode separation of 0.1%. We have compared the wakefields shown in Figure 4(b) obtained by summing the eigenmodes in the spectrum with the results from time domain simulations and found remarkable agreement. This represents the first ever successful effort in modeling an entire accelerating structure with actual dimensions.

5 Summary

In this paper, a case study of solving large sparse linear systems derived from accelerator structure simulations is presented. Both direct solvers and iterative solvers are investigated. A parallel multilevel preconditioner based on hierarchical finite element basis functions is discussed and has been implemented to accelerate the convergence of iterative solvers. The application performance and memory requirements are evaluated for our solvers on NERSC IBM SP parallel computers. With the capability of solving linear system with over 90 millions of DOFs, we have been able to perform end-to-end simulation of a realistic accelerator structure like the H60VG3 at a much higher accuracy than what was previously possible.

Acknowledgments

This work is supported by U.S. Department of Energy under contract number DE-AC03-76SF00515. Authors appreciate Anshul Gupta at IBM T. J. Watson Research Center for his technical support of using the Watson Sparse Matrix Package. Authors appreciate Din-Kow Sun for the hierarchical vector bases.

References

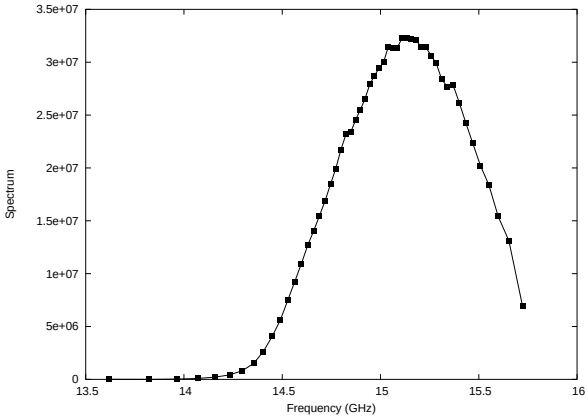
[1] Next Linear Collider homepage. http://www-project.slac.stanford.edu/nlc/home.html.

[2] P. R. Amestoy, I. S. Duff, J. Koster, and J. Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

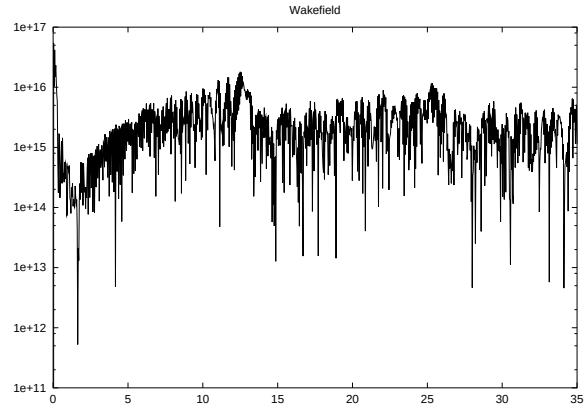
[3] P. Arbenz and R. Geus. Multilevel preconditioners for solving eigenvalue problems occuring in the design of resonant cavities. Technical

Table 2. Comparison of computing resources for the largest problem size attempted in modeling H60VG3 involving 14 different right hand sides using WSMP and the Conjugate Gradient with Multi-level Preconditioner (CGMP).

Solver	Number of DOFs	Number of nonzeros	Number of CPUs	Memory(GB)	Time (s)
WSMP	**30298905**	484912043	1024	487	4462.9
CGMP	**93147736**	3964961944	1024	836	6456.8



(a) The mode spectrum of the lowest dipole band in the H60VG3 structure



(b) The dipole wakefield obtained by summing 55 eigenmodes computed by Omega3P.

Figure 4. The results of The lowest dipole band in the H60VG3.

report, Institute of Scientific Computing, ETH Zrich, 2003.

[4] C. Ashcraft and R. Grimes. SPOOLES: An object-oriented sparse matrix library. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[5] M. H. Austern. *Generic Programming and the STL*. Professional computing series. Addison-Wesley, 1999.

[6] R. E. Band. Hierarchical bases and the finite element method. *Acta Numerica*, 5:1–43, 1996.

[7] A. Gupta. *WSMP: Watson Sparse Matrix Package*. IBM T. J. Watson Research Center, November 2000. http://www.cs.umn.edu/ agupta/wsmp.html.

[8] A. Gupta. Recent advances in direct methods for solving unsymmetric sparse systems of linear equations. *ACM Transactions on Mathematical Software*, 28(3):301–324, 2002.

[9] A. Gupta, G. Karypis, and V. Kumar. A highly scalable parallel algorithm for sparse matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, 8(5):502–520, 1997.

[10] L.-Q. Lee and A. Lumsdaine. Generic programming for high performance scientific applications. In *Proceedings of Java Grande and International Symposium on Computing in Object-oriented Parallel Environments*, 2002.

[11] X. S. Li and J. W. Demmel. Making sparse gaussian elimination scalable by static pivoting. In *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*, pages 1–17, 1998.

[12] D. Quinlan. *A++/P++ Manual*. Lawrence Livermore National Laboratory.

[13] J. Siek, A. Lumsdaine, and L.-Q. Lee. Generic programming for high performance numerical linear algebra. In *Proceedings of the SIAM*

Workshop on Object Oriented Methods for Interoperable Scientifi c and Engineering Computing (OO'98). SIAM Press, 1998.

[14] D.-K. Sun, J.-F. Lee, and Z. Cendes. Construction of nearly orthogonal nedelec bases for rapid convergence with multilevel preconditioned solvers. *SIAM Journal on Scientifi c Computing*, 23(4):1053–1076, 2001.

[15] Y. Sun. *The Filter Algorithm For Solving Large-Scale Eigenproblems From Accelerator Simulations*. PhD thesis, Stanford University, 2003.

[16] Y. Sun, N. Folwell, Z. Li, and G. Golub. High precision accelerator cavity design using the parallel eigensolver Omega3P. In *Proceedings of the 18th Annual Review of Progress in Applied Computational Electromagnetics*, 2002.

[17] T. Veldhuizen. Blitz++ home page. http://oonumerics.org/blitz.