Parallelization of the Parametric Patch Moment Method Code*

Xianneng Shen
Cornell Theory Center
Cornell University
Ithaca, NY 14853-3801

Gerald E. Mortensen, Chung-Chi Cha
Syracuse Research Corporation
Merrill Lane
Syracuse, NY 13210

Gang Cheng, Geoffery C. Fox
Northeast Parallel Architecture Center
Syracuse University
Syracuse, NY 13244

December 1994

Abstract

In this paper, we discuss the development and implementation of computational electromagnetics techniques on massively parallel architectures. We focus on advanced numerical formulations and parallel implementation for electromagnetic scattering problems. The goal of this work is to combine state-of-the-art numerical algortihms and massively parallel processing to increase the maximum target size that can realistically be solved using the method of moments (MoM) technique.

We have parallelized ParaMoM, a MoM program which utilizes the parametric patch formulation developed at the Syracuse Research Corporation (SRC), on different multiple instruction and multiple data (MIMD) distributed memory architectures. ParaMoM-MPP, the parallel implementation of ParaMoM, has been demonstrated to produce good performance and accuracy.

1 Introduction

The problem we are addressing is the prediction of the radar cross section (RCS) of 3D arbitrarily shaped electrically large targets. We report results obtained by combining state-of-the-art CEM techniques and state-of-the-art massively parallel processing technologies.

*This work was supported by the Plans and Programs Directorate (Signal Technology), Wright Laboratory Air Force Materiel Command (ASC); U.S. Air Force, Wright-Patterson AFB OH 45433-5000

Practical RCS prediction using numerical methods has long been thought of as unrealistic. This is because numerical solutions, while exact in concept, demanded amounts of computation and memory too large to accomplish in the past. The RCS prediction of a fighter-sized aircraft using MoM, for example, requires the solution of a matrix equation whose dimension can easily exceed 100,000. The impossibility of such computations also discouraged efforts to improve other aspects of CEM techniques. Successful developments of massively parallel processing (MPP) technologes have moved us into a position from which the opportunity now looks much better for solving the above-mentioned problem. Parallel computing not only drastically improves speed, it also prompts new developments in CEM techniques by bettering the prospects of real problem solutions.

In this paper, we discuss the development and implementation of an MoM code which computes RCS and antenna radiation on a Massively Parallel Processing (MPP) architecture. We utilize the parametric patch model proposed by Wilkes and Cha [2].The parametric formulation has been implemented by Syracuse Research Corporation (SRC) in a program called ParaMoM. ParaMoM utilizes curvilinear surface patches in conjunction with a new type of basis function suited to the curved domains. A brief overview of the parametric formulation is presented in Section 2.

The ParaMoM code has been parallelized on various MIMD distributed memory systems including Thinking Machine Corporation's CM-5 and the Intel Paragon. We present the details of the implementation of ParaMoM-MPP, the parallel implementation of ParaMoM, in Section 3.

The performance and scalability of ParaMoM-MPP are discussed in Section 4. Results for some of the Electromagnetic Code Consortium (EMCC) benchmark cases are shown and discussed in Section 4. The results obtained by ParaMoM-MPP are in good agreement with the EMCC benchmark measurement data.

We present our conclusions in Section 5.

2 An Overview of the ParaMoM Formulation

In this section we summarize the parametric patch formulation, originally proposed by Wilkes and Cha [2], used in the ParaMoM-MPP code. The parametric MoM formulation is defined for a class of surfaces which can be represented by two surface parameters (u and v). Each point on the target surface corresponds to a (u,v) coordinate pair. Most popular geometry modeling programs can produce such a parametric surface representation (e.g. NURBS or bicubic spline representations).

The basis functions used in the parametric formulation are similar to the well-known Rao-Wilton-Glisson (RWG) [3] functions but are defined in the parametric space of the surface. The basis function domains are triangles in the parametric (u,v) space. The potential integrals are performed on the basis function domains which conform to the curved target surface rather than the usual flat facet approximation. Therefore errors introduced by facetization are eliminated. The basis functions have all the desirable properties of the RWG functions, i.e. artificial charge accumlations at patch boundaries are avoided. When the patch dimensions become small compared to the radius of curvature, this basis function approaches the RWG basis function for a flat triangle. This basis function is defined in terms of a general surface parameterization, and it is not tied to any specific surface parameterization. This feature has allowed for simple modular inclusion of several different parameterizations in the method of moment procedure.

The ParaMoM-MPP code includes a number of capabilities and features. E-field, H-field, and Combined Field formulations are supported. An IBC formulation is included in the CM-5 version for use with material-coated targets. Parametric geometries are accepted in the standard IGES 114 (bicubic) and IGES 128 (nurbs) formats, as well as the SRC-developed GPE format. Wire antennas may be attached to the target surface. Gridding and material specification are performed in the ParaMoM geometry preprocessor. Up to three symmetry planes may be utilized to reduce computational and memory requirements for symmetric targets.

3 Parallel Implementation

The ParaMoM [4] code has the following phases: Setup, Precomputation, Matrix fill, Right hand side vector fill, Matrix factorization, Back-substitution solution, and Far-field solution. The setup phase reads in the parameters required for the computation. Precomputation computes arrays of position vectors and basis functions at each integral point. The matrix fill phase fills the elements of the moment matrix which can be one of EFIE, MFIE, and CFIE decided at setup time. The right hand side vector fill phase computes the excitation (right-hand side vector). The matrix factorization phase decomposes the moment matrix into lower and upper triangular matrices, then back substitution is applied to solve the linear system. Finally far-field and RCS are calculated at the far-field solution phase.

The computational expenses of the setup phase, precomputation phase, RHS vector fill phase, and scattered-field computation phase are of order N, where N is the number of unknowns (proportional to surface area of target). The matrix fill phase is of order N^2, the matrix factorization is of order N^3, and the matrix solution is of order N^2. Because our primary interest in applying parallel computers lies in reducing the total time required to solve large problems (i.e., large N), we concentrate our energy on reducing the N^2 and N^3 processes. All phases of the program were parallelized in this effort.

The ParaMoM-MPP code is implemented on three parallel platforms (CM-5, Intel, IBM SP-1). Each of the platforms is a coarse-grained MIMD (multiple instruction multiple data) with relatively powerful nodes. Each machine supports an explicit message-passing programming model. In each case, the message-passing library may be accessed from a Fortran program. ParaMoM-MPP is written in Fortran 77 with the message-passing library calls provided either by the computer vendor or from a public domain library, such as Parallel Virtual Machine (PVM).

The explicit message-passing approach has a number of advantages for the MoM application. Performance of a message-passing program is less dependent on the interconnection topology of the parallel machine than it is in data-parallel and shared-memory approaches.

Table 1: The elapsed time of writing the moment matrix to a SDA file using CMMD global write under CMMD synchronous sequential mode (recorded using the CMMD timer).

$N^{\dagger}$	32-node		512-node	
	write	fill	write	fill
3000	1.19(s)	1207(s)	0.39(s)	94.1(s)
5000	2.93(s)	3295.2(s)	0.78(s)	237.9(s)
10000			1.47(s)	901(s)

There is more direct control over the message volume and timing. Therefore, message-passing programs are more likely to port efficiently from one parallel architecture to another. In addition, the functions of the message-passing library are quite similar on all architectures; differences are mainly in syntax rather than in philosophy.

The implementation on Intel utilizes the NX message-passing library [5], the standard PVM (Parallel Virtual Machine) message-passing library [6] and ScaLAPACK [7], a distributed-memory version of the standard LAPACK linear algebra library developed at Oak Ridge National Laboratory, for excellent portability. This PVM implemention has been ported to the IBM SP-1 with little effort.

The CM-5 implementation utilizes the software supplied by Thinking Machine Corporation: the CMMD message passing library and the CMSSL matrix solver. The CMSSL library uses a data-parallel programming model and cannot be globally interfaced with the message-passing, matrix-filling algorithm. Therefore, in our implementation, the matrix fill and matrix factorization and solution (factor/solve) are two distinct program units. A high-speed device, such as the scalable data array (SDA) or DataVault, is used to link these two stages. The message-passing MoM matrix-filling program fills the matrix and writes it to a file in the format required for the factor/solve stage. The matrix is subsequently read in by the data-parallel matrix solver stage.

There is very little performance penalty for using separate program units for the filling and factor/solve because the DataVault or the SDA has very high data rates. As an illustration, the elapsed time for writing the moment matrix to a SDA file as measured by the CMMD timer is shown in Table 1. The writing operation is executed by the CMMD global write under CMMD synchronous sequential mode. One can see that the extra effort in using a high performance storage device to utilize both the message-passing paradigm and the data-parallel paradigm is justified.

There are a number of methods available to perform the solution of dense linear system of equations. In specialized problems, iterative approaches can be quite efficient in terms of memory and computation time. However, iterative techniques suffer when used to treat many right-hand side vectors. In this study, it is of interest to solve systems with a large number of right-hand side vectors (i.e., scatterers illuminated by multiple incident waves) and iterative methods were not considered. The Gaussian elimination method is employed under these conditions, since the computational intensive factorization need only be performed once.

In addition to computational cost concerns, it is necessary to consider memory requirements. On the class of computers we are considering, the main memory (RAM) is distributed among the nodes. The moment matrix is generally much too large to fit in the memory of one node, so it must be distributed equitably among the nodes. The matrix size is of order N^2, but all other arrays used in the ParaMoM code are of order N or lower.

To parallelize the ParaMoM code, we apply the problem partition approach which has received the most attention in scientific computing applications. In this approach, each processor executes substantially the same program, but on a portion of the problem data. Processors are loosely coupled throughout the computation, exchanging information whenever necessary. It is very suitable for solving large problems, where all available memory is required. The implementation difficulties are how to partition the problem to have a good load balance.

The task of filling the moment matrix is distributed among the nodes in the system. Each node has the responsibility of computing a number of columns of the matrix. The implementation of the setup phase utilizes parallel I/O to achieve parallel read. The implementation of the precomputation phase distributes the tasks of computing arrays between nodes, each node only compute a portion of the array then all nodes gether the results from all the nodes. The matrix fill implementation is illustrated in Figure 1. The RHS vector fill implementation distributes the vectors among the compute nodes. The matrix factorization and back substitution have different implementations on the different architectures. The Intel and IBM SP-1 implementation uses ScaLAPACK routines and the CM-5 implementation uses CMSSL subroutines. The far-field and RCS computation implementation lets each node compute the far-field in terms of its local solution vector obtained by solving the moment matrix equation, then the contributions are summed and put in one node.

4 Numerical Results

In this section we present the performance and scalability of ParaMoM-MPP and some numerical results computed by the ParaMoM-MPP code. We only show architectures here. Table 2 shows a performance comparison of ParaMoM on one processor SGI versus ParaMoM-MPP on various architecures.

The test targets used here were modeled using ACAD and gridded using the ParaMoM geometry preprocessor. All numerical results are compared with the measurements published by the EMCC (see references [9, 10]). For the results shown, the patch sampling rates used for the benchmark comparison runs are somewhat conservative and would be reasonable on a large problem. In most cases, the maximum edge length specified during target gridding was one-eighth of a wavelength.

The first example is the RCS from a flat plate in the shape of a wedge cylinder with gap. The incident wave has a frequency of 5.9 GHz with horizontal polarization. The receiver's polarization is also horizontal. The parametric patch model of the target has the maximum edge length $\frac{\lambda}{8}$, and the number of triangular patches is 1008 with 553 nodes and 1560 edges.

Parallel Moment Matrix Fill Algorithm

1. Loop over source patches from one to N_f;
2. Loop over three possible edges on the source patch;
   (a) Find out the global index of the edge;
   (b) Compute the index of the node in whose slab includes the column whose number is the same as the global edge index;
   (c) Compute the local index for the location of the column in the node;
   (d) The selected node increases its job counter by one and stores the information of the edge (the global edge index, the local edge index, and the local position in the node);
3. Those nodes whose job counter is 0 go back to 1;
4. Loop over field patches from one to N_f;
   (a) Compute all the integration points for the source-field patch pair interaction (Green's function);
   (b) Loop over source basis functions for that node, from 1 to its job counter;
      - Loop over all possible field basis functions (or edges on the field patch)
      - Use the formula derived in Chapter 2 to compute the elements of the moment matrix;
      - Sum all the contributions from the four pairs of patches associated with each source-field basis function;

End of the source patch loop.

Figure 1: The pseudo code of the parallel fill algorithm

Table 2: Time Comparisons

	SGI Indigo R4000 (estimated)	Paragon (98 nodes)	CM5 (512 ndoes)	SP1 (EST.) (58 nodes)
Fill Time(in secs)	57,000	888	901	871
speedup rel. to SGI	1	64	63	65
Factor Time(secs)	142,000	358	156	
speedup rel. to SGI	1	397	910	
Total Time(secs) Fill + Factor + I/O	199,000	1715	1067	
speedup rel. to SGI	1	116	186	

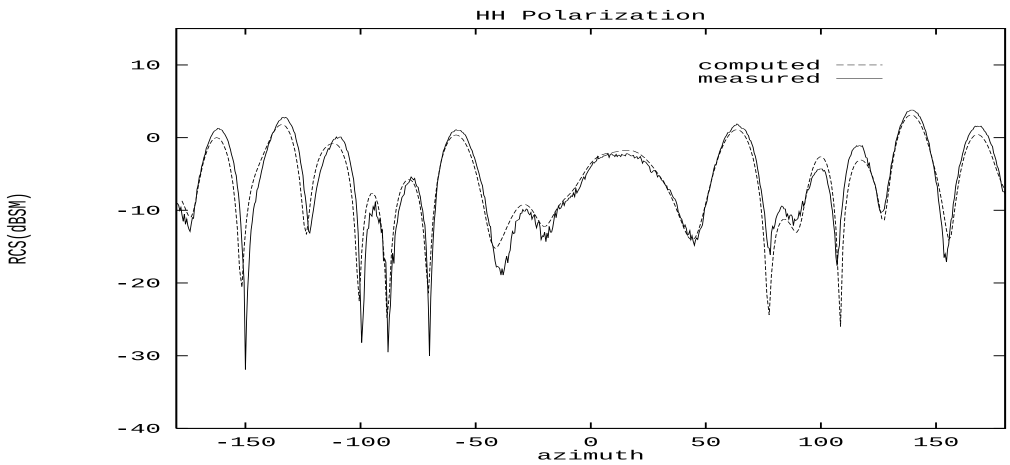

Figure 2: The RCS with HH polarization of a wedge cylinder with gap

The radius of circular part of the wedge cylinder is λ and the length of its straight side is two λ.

The monostatic radar cross section for horizontal transmitter and receiver polarizations (HH) is shown in Figure 2. The result is obtained by running the ParaMoM-MPP on a 80-node partition Intel NAS paragon. Six Mbytes of memory per node are required to run this target. There are 360 excitation vectors to compute the monostatic radar cross section. The computation takes 309 seconds by a wall clock. The RCS, in dB λ^2, plotted against the azimuthal angle is shown in Figure 2 in a 10^o-elevation conical cut.

The parametric patch model for the "single ogive" target with three symmetry planes are used so one eighth of the body is actually modeled. The ogive is illuminated by an incident wave at 9 GHz. The single ogive is gridded into two different models. The first model utilizes three symmetry planes, so only one eighth of the single ogive needs to be modeled. This

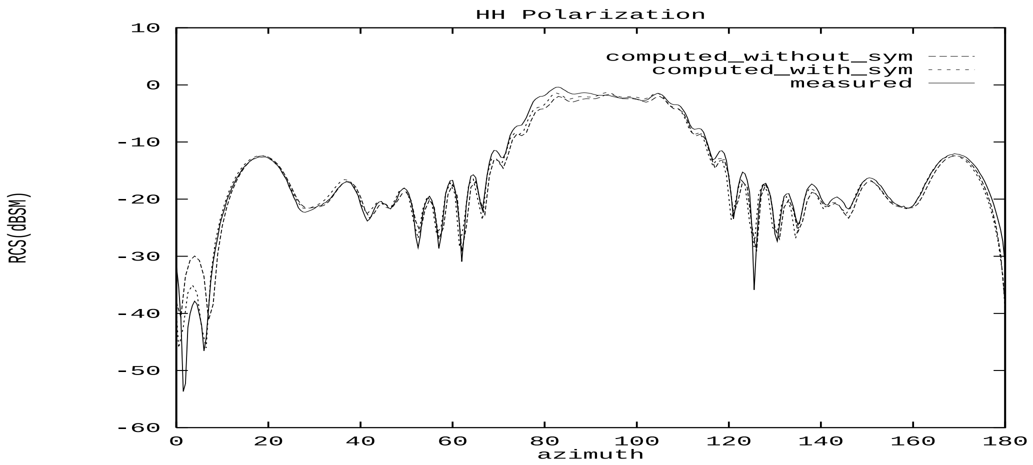

Figure 3: The RCS with HH polarization of a "single ogive"

one-eighth single ogive is modeled by 2088 curved triangles with 3208 edges and 1121 nodes. The maximum edge length is $\frac{\lambda}{12}$. The second model uses no symmetry plane. It consists of about 5000 curved patches, and 7332 edges.

The monostatic radar cross section characteristics for both HH and VV polarizations are plotted in dBSM as functions of the azimuthal angle in Figures 3 and 4. The elevation angle is zero. In Figures 3 and 4, and computed_without_sym denotes the model uses no symmetry planes, computed_with_sym denotes the target modeled with three symmetry planes. Although the number of the curved patches of the model without symmetry planes is much less dense than that with symmetry planes, Figures 3 and 4 shows good agreement. This demonstrates that the parametric patch method of moments with relatively fewer unknowns still obtains good accuracy. There are 360 RHS vectors for this problem. This problem is run on a 64-node partition of the NAS Paragon. The symmetry model requires 9 Mbytes per node and takes 9648 seconds to complete the run. The model without symmetry planes requires 21 Mbytes per node and takes 3918 seconds to complete the run.

5 Conclusion

In this paper we describe the parallelization of ParaMoM, a MoM code which computes the RCS of 3-D arbitrarily shaped conducting bodies utilizing a parametric curved patch model.

We describe the development and implementation of ParaMoM-MPP, the parallel version of ParaMoM, a MoM code which computes the RCS of 3-D arbitrarily shaped conducting bodies utilizing a parametric curved patch model. ParaMoM-MPP was implemented on three different MIMD distributed memory architectures: the Thinking Machine's CM-5, Intel machines, and IBM SP-1

The performance data listed in Section 4 shows good performance on three platforms. We

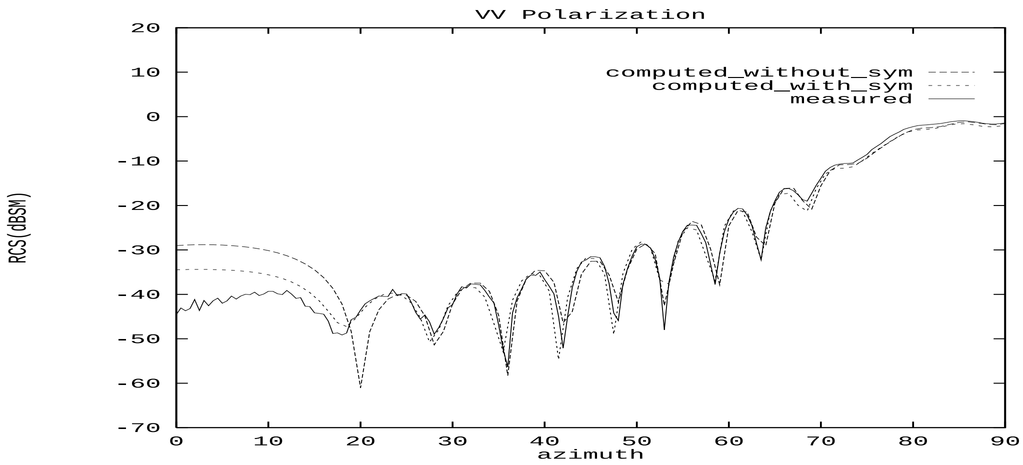

Figure 4: The RCS with VV polarization of a "single ogive"

demonstrate the accuracy of the code through comparison of computed data with published EMCC test case measurements.

References

[1] Roger F. Harrington, *Matrix Methods for Field Problems*, Proc. IEEE, Vol. 55, No. 2, pp. 136–149, Feb. 1967.

[2] Debra L. Wilkes and Chung-Chi Cha, *Method of Moments Solution with Parametric Curved Triangular Patches*, IEEE Antennas and Propagation International Symposium Digest, pp. 1512–1515, 1991

[3] S.M. Rao, D. R. Wilton and A. W. Glisson, *Electromagnetic Scattering by Surfaces of Arbitrary Shape*, IEEE Trans. Antennas Propagat., Vol. AP-30, pp. 409–418, May 1982.

[4] Syracuse Research Corporation, Parametric Method of Moments (ParaMoM) RCS Prediction Package User's Manual, Version 1.0, SRC Technical Report TD 92-1321, October 1992.

[5] Intel Corporation, iPSC/2 and iPSC/860 User's Guide, April 1991.

[6] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam, PVM 3 User's Guide and Reference Manual, Oak Ridge National Laboratories report ORNL/TM-12187, May 1993.

[7] J. Choi, J. J. Dongarra, R. Pozo, and D. W. Walker, *Scalapack: A scalable linear algebra library for distributed memory concurrent computers*, In Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation, pp. 120–127. IEEE Computer Society Press, 1992.

[8] Thinking Machines Corporation, CMSSL for CM Fortran, Version 3.1, June 1993.

[9] Alex C. Woo, Helen T. G. Wang, Michael J. Schuh, and Michael L. Sanders, *Benchmark Plate Radar Targets for the Validation of Computational Electromagnetics Programs*, IEEE Antennas Propagat. Magazine, Vol. 34, No. 6, pp. 52–56, Dec. 1992.

[10] Alex C. Woo, Helen T. G. Wang, Michael J. Schuh, and Michael L. Sanders, *Benchmark Plate Radar Targets for the Validation of Computational Electromagnetics Programs*, IEEE Antennas Propagat. Magazine, Vol. 35, No. 1, pp. 84–89, Feb. 1993.