Computational Electromagnetics and Parallel Dense Matrix Computations

Kimmo Forsman[*] **William Gropp**[†] **Lauri Kettunen**[*] **David Levine**[†]

Abstract

We present computational results using CORAL, a parallel, three-dimensional, nonlinear magnetostatic code based on a volume integral equation formulation. A key feature of CORAL is the ability to solve, in parallel, the large, dense systems of linear equations that are inherent in the use of integral equation methods. Using the `Chameleon` and `PSLES` libraries ensures portability and access to the latest linear algebra solution technology.

1 Introduction

For the past twenty years most electromagnetic field computations have been done using programs based on the finite element method and run on sequential computers. However, limitations of accuracy, problem size, and solution time make further advances using this approach more and more difficult. We believe an approach that combines integral equation methods and massively parallel computers can address many of the current limitations and provide timely and accurate solutions to large, computationally demanding problems.

In this paper we discuss CORAL, a volume integral equation code that runs on massively parallel processors as well as workstation networks and can solve three-dimensional, nonlinear magnetostatic problems. A key component of CORAL is the use of the `Chameleon` and `PSLES` libraries to provide parallel portability and access to a variety of parallel linear system solvers to solve the large, dense systems of linear equations that are inherent in the use of integral equation methods.

2 Integral Equation Formulation in CORAL

In this section we briefly discuss the integral equation formulation used in CORAL. More complete details may be found in [6]. We denote magnetic flux density with B, magnetic field strength with H, magnetization with M, permeability with μ, and susceptibility with χ. The formulation behind CORAL is based on the idea of superposition of fields from current and magnetization sources. Let H^s stand for the magnetic field strength from current sources in the absence of magnetic materials. The field component from magnetization of materials is denoted with H^m. The field from source currents can be integrated from Biot-Savart's law and is therefore known a priori. Arranging the unknown terms on the left and the source terms on the right, we get

$$H - H^m = H^s \ . \tag{1}$$

[*]Tampere University of Technology, Tampere, Finland.

[†]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4801. This work was supported by the Office of Scientific Computing, U. S. Department of Energy, under Contract W-31-109-Eng-38.

The H^m-field at point r, resulting from a distribution of M in $V(r')$, can be integrated such that

$$H^m(r) = grad\left[\frac{-1}{4\pi}\int\limits_V \frac{M(r')\cdot(r-r')}{|r-r'|^3}dv'\right] . \tag{2}$$

Since M can be given in terms of H, namely,

$$M = \chi(|H|)H, \tag{3}$$

we get

$$H - H^m(\chi, H) = H^s \ , \tag{4}$$

and hence the fields on the left-hand side can be written in terms of H, if χ is known.

The formulation employed in CORAL is now developed by multiplying the divergence condition of B,

$$div\,B \ = \ div\,\mu H \ = \ 0\ , \tag{5}$$

with an appropriate test function, substituting Eq. (4) into (5), and applying integral relationships (i.e., theorems analogous to Green's first identity) to get

$$\int\limits_V \mu H\cdot h' - \int\limits_V \mu H^m(\chi,H)\cdot h' = \int\limits_V \mu H^s\cdot h', \quad \forall h', \tag{6}$$

where h' is the gradient of the test function.

In order to establish a discrete problem, the magnetic field H is approximated in the space of "edge elements," that is, Whitney elements of degree $p = 1$ [1]. The system of equations is developed from Eq. (6) by choosing h' the same as the basis functions of the space spanned by the edge elements. The numerical problem implies the tangential continuity of H at all points, whereas the normal continuity of B is satisfied only in the weak sense.

In magnetostatics the problems are typically nonlinear, because the properties of the materials depend on the field strength. These kinds of problem are solved iteratively; an initial guess for χ is inserted, and the nonlinearity is taken into account by solving successive problems updating the material data at each cycle. After a few cycles the Newton-Raphson method is applied to accelerate the convergence.

3 Parallelization Aspects

The parallel version of CORAL code is built on top of two libraries from `PETSc` (Portable and Extensible Tools for Scientific computing) [3], a large toolkit of software for portable, parallel (and serial) scientific computation. One of these, the `Chameleon` [2] library, is a second-generation message-passing system that provides a uniform way to access third-party message-passing libraries such as `p4`, PICL, and PVM, as well as vendor-specific message-passing libraries such as Intel's NX and IBM's EUI-H.

The other `PETSc` library used was the Parallel Simplified Linear Equation Solvers (`PSLES`) package [3]. This library provides access to a wide variety of iterative solvers, preconditioners, and accelerators in an easy-to-use, efficient, and portable manner. `PSLES` supports direct methods using sparse or dense LU, and a wide variety of iterative methods and preconditioners. `PSLES` allows the specification of options such as the solver algorithm, choice of preconditioning matrix, and setting of the solution tolerance.

```
ctx = PSVCreate(mat,PSVLU)        ctx = PSVCreate(mat,PSVBDD)
                                  PSVSetAccelerator(ctx,ITGMRES)
PSVSetUp(ctx)                     PSVSetUp(ctx)
PSVSolve(ctx,b,x)                 its = PSVSolve(ctx,b,x)
PSVDestroy(ctx)                   PSVDestroy(ctx)
```

FIG. 1. PSLES *Code for LU factorization and GMRES with block diagonal preconditioning*

The key data structures in CORAL are the integral equation and Jacobian matrices. Each nonlinear iteration requires the generation of a new integral equation matrix or Jacobian matrix. We decomposed the matrices rowwise such that each processor generates and updates certain rows of the integral equation matrix or the Jacobian matrix. During matrix generation no data broadcasting between the processors is needed. In addition, only a small amount of overlapping data is computed on two or more processors.

The elements of the integral equation matrix are computed from the solution to the linear system from the previous nonlinear iteration and some terms, which arise from Eq. (2), that depend only on the problem geometry. Because of the lowest-order "edge element" approximation, magnetization becomes a constant vector within each element, and therefore one has to integrate terms of the form $(r - r')/|r - r'|^3 dv'$. We compute the appropriate integrals for each processor once, store them to disk files, and read them during each successive nonlinear iteration. This approach saves a considerable amount of processing time.

The systems that arise in CORAL have several special features. First, the linear system that arises is asymmetric. Second, each system of linear equations arises from an outer nonlinear problem and so may need only a relatively low accuracy solution. Third, the actual matrix, while dense, has many "small" elements. Finally, the size of the matrices to be solved varies significantly according to the mesh refinement and desired solution accuracy.

The classical method for solving dense systems of linear equations is LU factorization. However, the solution time for LU factorization scales as $O(n^3)$ and can be prohibitive for large values of n. In CORAL, there may be computational advantages to using iterative methods, such as using the solution computed at the previous nonlinear iteration as the starting guess to solve the next linear system or allowing an early exit with an approximate solution. There may also, however, be cases where it is useful to use a direct method, such as solving the linear system on the first nonlinear iteration.

Ideally, we would like the capability to switch easily between different solution methods. The PSLES library provides that capability. PSLES allows the user to easily experiment with the choice of different solvers. For example, the PSLES code fragments in Figure 1 use LU factorization and GMRES with block diagonal preconditioning, to solve a linear system. Note the similarity of the two code fragments.

4 Computational Experiments

In this paper we focus on linear algebra and parallel computing experiments. Details of the accuracy of the calculated electromagnetic field have been given in other papers [4, 5, 6].

Table 1
TEAM 13 Timings, 2041 Equations

Preconditioner	LU	GMRES	Bi-CGStab
None	99.1		
BDD(4)		25.8/(25)	40.8/(23)
Band(25%)		26.4/(21)	38.7/(18)
SILUF		229.5/(177)	Not converged

Table 2
TEAM 20 Timings, 2867 Equations

Preconditioner	LU	GMRES	Bi-CGStab
None	166.4		
BDD(4)		30.5/(16)	41.7/(15)
Band(25%)		28.3/(12)	39.4/(11)
SILUF		34.8/(16)	51.8/(14)

4.1 Sequential Results

The experiments in this section compare sequential implementations of LU factorization and several iterative methods. All the results in this section were computed on a DEC Alpha 3000-600 AXP workstation. The test problems used are the international TEAM (Testing Electromagnetic Analysis Methods) benchmark problems, numbers 13 [7] and 20 [8]. TEAM 13 was run using a mesh that resulted in a system of 2041 equations. TEAM 20 was run using a mesh that resulted in a system of 2867 equations.

Tables 1 and 2 compare LU factorization (from LAPACK) with two iterative methods, GMRES and Bi-CGStab, using three different preconditioners. The row *None* means no preconditioner was used. The row *BDD(4)* refers to a block diagonal LU preconditioner using four blocks of equal size. The row *Band(25%)* is an incomplete LU preconditioner that uses a band containing approximately 25% of the matrix. The row *SILUF* is a standard incomplete LU factorization (i.e., the same idea as in Cholesky factorization, but for nonsymmetric matrices).

For each method and preconditioner, the table entries are the maximum CPU time in seconds, over all nonlinear iterations, to solve a linear system, followed by the number of iterations required by the iterative solver to solve that linear system to an accuracy of 1.0e-9. For the iterative methods, if P is the preconditioning matrix, then the initial guess we used for the starting solution was the solution to $Px = b$.

For both problems, both GMRES and Bi-CGStab, using either the BDD(4) or Band(25%) preconditioners, are more effective than LU factorization. Also, GMRES outperforms Bi-CGStab on both problems as well.

4.2 Parallel Results

The results in this section were computed on an IBM SP1 parallel computer with 128 RS/6000 model 370 processors, each with 128 Mbytes of memory and a one Gbyte local disk.

Table 3
TEAM 13 Timings, Different Nonlinear Solution Methods, 801 Equations

Method	4 Proc.	8 Proc.
LU	59	34
GMRES	23	235

Table 4
APS Dipole Magnet Timings, 7536 equations

No. of Processors	16	32	64
LU factorization	948.2	533.0	308.1
RHS generation	396.3	204.6	71.7
Matrix generation	824.3	412.6	207.6
CPU time/sec.	18306	11208	6878

Compilation was done using the IBM `xlf` Fortran compiler with level O3 optimizations. `Chameleon` generated EUI-H message-passing calls.

For the results in Table 3 TEAM 13 was solved using a mesh that led to a system of 801 equations. The rows *LU* and *GMRES* refer to the method used (on four and eight processors, respectively) to solve the linear system that arises each nonlinear iteration. The row *LU* refers to using LU factorization. The row *GMRES* refers to using GMRES with block diagonal preconditioning. In the case of GMRES, except for the first nonlinear iteration, the solution to the linear system from the previous nonlinear iteration was used as the starting guess.

GMRES with block diagonal preconditioning is superior to LU factorization when four processors are used, but significantly worse using eight processors. Our experience is that block diagonal preconditioning works well as long as the blocks are large enough (in general, we have found it best not to use more than four blocks for problems with 1000–3500 equations.) However, the parallel version of the block diagonal preconditioner used in `PSLES` uses one block per processor. In this case, the blocks are large enough when we use four processors but too small when we use eight processors (a similar result was observed using block diagonal preconditioning with eight blocks in other sequential tests as well.)

As a third test problem we used a positron accumulator ring dipole magnet from the Advanced Photon Source (APS) at Argonne National Laboratory. The APS dipole magnet was run using a mesh with 34,645 tetrahedral mesh elements resulting in a system of 7536 equations. Results using LU factorization each nonlinear iteration on 16, 32, and 64 processors are shown in Table 4. The rows show the three most computationally demanding steps: linear system solution, right-hand-side generation, and matrix generation. Good speedups are obtained.

4.3 Discussion

Our sequential results show that iterative methods with appropriate preconditioners can outperform direct factorization, even when the underlying problem is a dense matrix.

Simple, easily parallelized preconditioners such as BDD, however, do not scale well to large numbers of processors. This result suggests two approaches for better parallel performance. One is to use efficient parallel implementations of the ILU preconditioner (using multicoloring for better parallelism). The other is to use BDD, but keep the number of BDD blocks independent of the number of processors, and use a parallel LU for each block. This approach requires that the parallel LU software run on arbitrary subsets of processors; this is relatively easy for software written using MPI or `Chameleon`.

Acknowledgments

We thank Sean Pratt, Jennifer Rovegno, Jukka Salonen, Diana Tabor, Hania Yassin, and Vector Fields Inc. for their assistance. We acknowledge use of the Argonne High-Performance Computing Research Facility.

References

[1] A. Bossavit. Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism. In *IEE Proceedings*, volume 135, Pt. A, pages 493–499, 1988.

[2] W. D. Gropp and B. Smith. Users manual for the Chameleon parallel programming tools. Technical Report ANL-93/23, Mathematics and Computer Science Division, Argonne National Laboratory, March 1993.

[3] W. D. Gropp and B. F. Smith. Scalable, extensible, and portable numerical libraries. In *Proceedings of Scalable Parallel Libraries Conference*, pages 87–93. IEEE, 1994.

[4] L. Kettunen, K. Forsman, D. Levine, and W. Gropp. Solutions of team problem #13 using integral equations in a sequential and parallel computing environment. In *Proceedings of the Fourth International TEAM Workshop*, pages 41–43, Florida International University, Miami, 1994.

[5] L. Kettunen, K. Forsman, D. Levine, and W. Gropp. Solutions of team problems 13 and 20 using a volume integral formulation. In *Proceedings of Aix-les-Bains TEAM Workshop*, Aix-les-Bains, France, 1994.

[6] L. Kettunen, K. Forsman, D. Levine, and W. Gropp. Volume integral equations in nonlinear 3-d magnetostatics. Technical Report MCS-P460-0894, Argonne National Laboratory, 1994.

[7] T. Nakata, N. Takahashi, K. Fujiwara, K. Muramatsu, T. Imai, and Y. Shiraki. Numerical analysis and experiments of 3-D non-linear magnetostatic model. In *Proceedings of TEAM Workshop on Computation of Applied Electromagnetics in Materials*, pages 308–310, Okayama, Japan, 1990.

[8] T. Nakata, N. Takahashi, and H. Morishige. Analysis of 3-d static force problem. In *Proceedings of TEAM Workshop on Computation of Applied Electromagnetics in Materials*, Sapporo, Japan, 1993.