UCRL-CONF-216989

A parallel computer implementation of fast low-rank QR approximation of the Biot-Savart law

D. A. White, B. J. Fasenfest, M. L. Stowell

November 10, 2005

Progress in Electromagnetics Research Symposium
Cambridge, MA, United States
March 26, 2006 through March 29, 2006

A Parallel Computer Implementation of Fast Low-Rank QR Approximation of the Biot-Savart Law

D. A. White, **B. J. Fasenfest**, **M. L. Stowell**
Lawrence Livermore National Laboratory, USA

Abstract—In this paper we present a low-rank QR method for evaluating the discrete Biot-Savart law on parallel computers. It is assumed that the known current density and the unknown magnetic field are both expressed in a finite element expansion, and we wish to compute the degrees-of-freedom (DOF) in the basis function expansion of the magnetic field. The matrix that maps the current DOF to the field DOF is full, but if the spatial domain is properly partitioned the matrix can be written as a block matrix, with blocks representing distant interactions being low rank and having a compressed QR representation. The matrix partitioning is determined by the number of processors, the rank of each block (i.e. the compression) is determined by the specific geometry and is computed dynamically. In this paper we provide the algorithmic details and present computational results for large-scale computations.

1. Introduction

The computation of magnetic fields from a prescribed electric current is a common problem in magnetic design and analysis. One approach is to form the problem as a Partial Differential Equation (PDE) for the unknown field with the prescribed electric current as the source term. Regardless of the particular PDE formulation, e.g. a magnetic vector potential formulation or a mixed B-H formulation, a large volumetric mesh must be employed, and some boundary condition must be applied on the outer boundary of the mesh. In contrast to the PDE approach, the Biot-Savart law can be employed to directly compute the magnetic field due to the prescribed current [1]. The advantage of the Biot-Savart law approach is that a full volume mesh is not required, and no boundary conditions need be applied. The disadvantage of the Biot-Savart approach is the computational cost, if there are $O(N)$ magnetic field observation points and $O(M)$ current samples the cost is $O(N*M)$. In this paper we review a fast low-rank QR method for compressing the $M \times N$ Biot-Savart matrix. The approach is similar to low-rank QR methods developed for boundary element electrostatics [2] [3] and for low frequency electric field integral equations [4]. The key difference with our approach is that we are concerned with volumetric current densities and implementation on parallel computers.

2. Formulation

The law of Biot and Savart is given by

$$\vec{B}\,(x) = \nabla \times \vec{A} = \frac{1}{\mu 4\pi} \int_{\Omega'} \frac{\vec{J}(x') \times (x - x')}{|x - x'|^3} d^3x'. \tag{1}$$

where $J(x')$ is the known current density at the source point x', and $B(x)$ is the desired magnetic flux density at the observation point x. We assume that we have a finite element representation for J over the volume Ω', and a finite element representation for B over a surface Γ,

$$\vec{J} = \sum_{i=1}^{N} \xi_i \vec{W}_i^2 \ \ , \vec{B} = \sum_{j=1}^{M} \beta_j \vec{W}_j^1, \tag{2}$$

where ξ_j and β_j are the ith degree-of-freedom (DOF), and $\vec{W}_i^2$ and $\vec{W}_j^1$ are vector basis functions. Inserting the basis function expansions (2) into (1) yields the discrete Biot-Savart law

$$\mathbf{M}\bar{\beta} = \mathbf{Z}\bar{\xi}, \tag{3}$$

where

$$\mathbf{Z}_{ij} = \int_{\Gamma} \int_{\Omega'} \frac{1}{\mu 4\pi} \frac{\vec{W}_i^2(x') \times (x - x') \cdot \vec{W}_j^1(x)}{|x - x'|^3} d\Omega' \ d\Gamma, \tag{4}$$

and

$$\mathbf{M}_{ij} = \int_\Gamma \vec{W}_i^1(x) \cdot \vec{W}_j^1(x)\ d\Gamma \tag{5}$$

and where $\bar{\xi}$ and $\bar{\beta}$ are the arrays of DOF. The matrix $\mathbf{M}$ is a "mass matrix" due to the fact that the basis functions are not orthogonal. The mass matrix is extremely sparse and the computational cost for forming and solving this matrix is negligible. In many applications the problem of determining the B-field can be posed in terms of the magnetic vector potential $A = \nabla \times B$ with

$$\vec{A}(x) = \frac{1}{\mu 4\pi} \int_\Omega \frac{\vec{J}(x')}{|x - x'|} d^3x'. \tag{6}$$

Using a finite element representation for A yields another version of the discrete Biot-Savart law

$$\vec{J} = \sum_{i=1}^{N} \xi_i \vec{W}_i^2 \ \ , \vec{A} = \sum_{j=1}^{M} \alpha_j \vec{W}_j^1, \tag{7}$$

$$\mathbf{M}\bar{\alpha} = \mathbf{Y}\bar{\xi}, \tag{8}$$

where

$$\mathbf{Y}_{ij} = \int_\Gamma \int_{\Omega'} \frac{1}{\mu 4\pi} \frac{\vec{W}_i^2(x') \cdot \vec{W}_j^1(x)}{|x - x'|} d\Omega'\ d\Gamma. \tag{9}$$

We will refer to the $M \times N$ matrices $\mathbf{Z}$ and $\mathbf{Y}$ as Biot-Savart matrices. The computation of these matrices involves singular and near-singular integrals. The surface integration is performed using standard Gaussian quadrature points for each surface element. The volume integration uses an adaptive integration rule, which varies the order of Gaussian quadrature based on the distance between the source point x' and the observation point x. When the surface element containing x is a face of the volume element containing x', a highly accurate height-based singularity cancellation quadrature rule is used [5]. The matrices (4) and (9) are constructed using 2-form or "face elements" for the basis functions W^2 and 1-form or "edge elements" for the basis functions W^1, see [6] for details on the construction of the basis functions.

Our primary application for the discrete Biot-Savart law is providing boundary conditions for finite element solution of multi-conductor eddy current problems. In each conductor we solve the time-dependent vector diffusion equation using an edge element based A-ϕ finite element formulation [7]. Clearly the B-field in the air surrounding the conductors is critical. The finite element formulation requires that either $\hat{n} \times \vec{A}$ or $\hat{n} \times \vec{B}$ be specified on the conductor boundaries, corresponding to inhomogeneous Dirichlet or Neumann boundary conditions, respectively. Our approach for dealing with the B-field in the air surrounding the conductors is to use the discrete Biot-Savart law (3) or (8) as the boundary condition on each conducting surface.

3. Parallel Implementation

We assume that the volume Ω has been partitioned into K partitions, where K is the number of computational processors, with each partition having an equal number of volume elements. The volume elements are distributed via the partitioning. The surface Γ is also partitioned into K equally sized surface partitions. Note however that the surface elements are not distributed via the surface partitions, each processor can access the entire surface mesh. The Biot-Savart matrix is then decomposed into a $K \times K$ block matrix, with every block $Z^{pq}, p \in \{1 : K\}, q \in \{1 : K\}$ representing the interaction of surface partition Γ_p with volume Ω_q. The qth processor computes blocks $Z^{pq}, p = 1 : K$, i.e. a column of blocks. Note that the matrix is decomposed via a partitioning of elements, hence the matrices Z^{pq} are overlapping in DOF space. The specific partitioning algorithm used to partition the elements is not critical, in the examples below we employ a graph-based algorithm [8]. The key point is that if the partitions Γ_p and Ω_q are well-separated then the sub-matrix Z^{pq} will have a low-rank QR decomposition. The procedure for computing the low-rank QR decomposition is described below. We define "well-separated" as follows: the bounding spheres for the element partitions Γ_p and Ω_q are computed, if the bounding spheres do not intersect then the partitions are considered well-separated and a low-rank QR representation of Z^{pq} is computed. We employ a recursive procedure for computing Z^{pq} when partitions Γ_p and Ω_q are not well-separated. This results in a hierarchical representation for Z. If Γ_p and Ω_q are not well separated, Ω_q is divided into eight equally sized sub-partitions, Γ_p is divided into four equally sized sub-partitions, and the "well-separated test" is applied to the sub-partitions Γ_{pi} and Ω_{qj}, $i = 1:4, j = 1:8$. A space-filling curve algorithm is used for creating the sub-partitions. The process is applied recursively, with a low-rank QR representation computed for well-separated sub-partitions. The recursion is halted when a volume sub-partition contains fewer than some number of elements, for applicationexample 512. If at the lowest level of recursion the interaction is not well separated, it is simply represented by a dense matrix.

No parallel communication is required in the construction of the hierarchical Biot-Savart matrix, each processor has the elements that it needs to perform the integrals. Each processor has the same amount of work hence the computation of is load balanced. Note, however, that in the low-rank QR approximation the rank k is computed dynamically, and the rank k depends upon the geometry. Hence the application of the hierarchical Biot-Savart matrix, i.e. the matrix-vector multiplication $\bar{\beta} = \mathbf{Z}\bar{\xi}$, may not be perfectly load balanced. Also note that the application of the hierarchical Biot-Savart matrix does require parallel communication. This communication is as follows: (1) each processor q does a gather operation to get the values of $\bar{\xi}$ that it needs, (2) each processor q loops over the sub-matrices $Z^{pq}, p = 1:K$ and computes $\bar{\beta}_q = \mathbf{Z}^{pq}\bar{\xi}_q$, (3) each processor participates in a global reduction on $\bar{\beta}_q$.

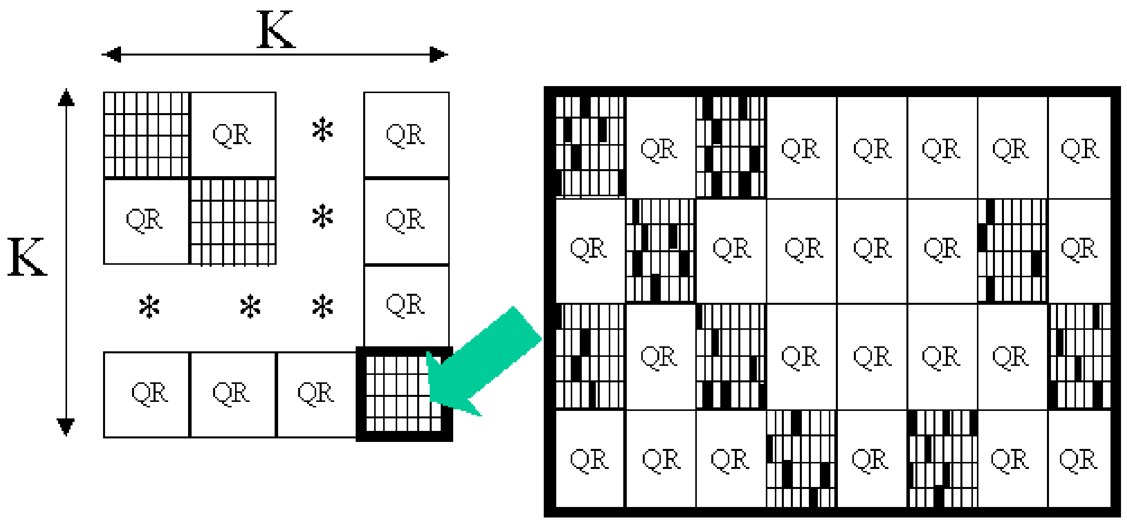

Figure 1: Hierarchical partitioning of the Biot-Savart matrix. The highest level of partitioning is based on the number of processors, as represented by the left-most matrix. The sub-matrices Z^{pq} representing near interactions are hierarchically decomposed into 8 sub-volumes and 4 sub-surfaces, as illustrated by the right-most matrix.

4. Low-Rank QR Decomposition

When Γ_p and Ω_q are well separated the matrix Z^{pq} will have a low-rank representation

$$Z^{pq}_{m\times n} \approx Q_{m\times k} \times R_{k\times n}, \tag{10}$$

where k is the rank. We do not want to form the entire Z^{pq} and then compress it, rather we sample the matrix by picking s rows and columns of Z^{pq}, where s is some predetermined number, e.g. 50. The procedure for picking the sampled rows and columns is ad-hoc, the procedure that we employ is described in [4]. The sampling procedure is solely linear algebra, it does not depend upon the particular Green's function, finite element basis functions, etc. For the ad-hoc sampling procedure to be effective we must have s greater than the expected rank. The algorithm for computing $Q_{m\times k}$ and $R_{k\times n}$ is as follows:

Step 1 : Form the sampled column matrix $S^c_{m\times s}$ and the sampled row matrix $S^r_{s\times n}$.

Step 2: Compute the rank-revealing QR decomposition $\tilde{Q}_{m\times s}\tilde{R}_{s\times s} = S^c_{m\times s}$ using LAPACK routines DGEQPF and DORGQR. The rank k is determined by the criteria $\tilde{R}_{kk} <$ *thresh*$\cdot\tilde{R}_{11}$ where *thresh* is a threshold value, we then keep only k columns of $\tilde{Q}$, denote this as $Q_{m\times k}$, and discard $\tilde{R}$.

Step 3: We form a new matrix $\hat{Q}_{s\times k}$ by taking s rows of $Q_{m\times k}$, the exact same rows as used to construct S^r.

Step 4: Compute the least-squares solution to $\hat{Q}_{s\times k} R_{k\times n} = S^r_{s\times n}$ using LAPACK routine DGELSS.

At this point we have the desired matrices $Q_{m\times k}$ and $R_{k\times n}$ which approximate $Z^{pq}_{m\times n}$. The quality of the approximation, and the amount of compression (the rank k), are determined by the value of *thresh* used in Step 2 above. Our approach, being based on highly tuned LAPACK routines, is efficient both in terms of FLOPS and memory usage. The complexity is $O(m \cdot s) + O(s \cdot n)$, using a fixed value of s yields a linear complexity in m and n.

5. Examples

In these examples we compute a hierarchical low-rank QR approximation of the matrix defined by Equation (9). For the first example consider the geometry shown in Figure 2. This geometry consists of 19000 volume elements and is partitioned for 16 processors. Therefore the Biot-Savart matrix will be a 16×16 block matrix. Each block Z^{pq} has roughly 1200 rows and 4000 columns. Using values of $s = 50$ and *thresh* $= 0.005$ gives the parallel rank map shown in (11). The compression is significant, each 1200×4000 matrix is compressed to $Q_{1200\times k} + R_{k\times 4000}$ where k is the value shown in (11). Note that the blocks labeled with rank 00 are near interactions and have full rank. These blocks were decomposed further as explained in Section 3 above. For example, the Z^{11} near-interaction matrix will be decomposed into 8 sub-volumes and 4 sub-surfaces, each resulting sub-matrix has roughly 270 rows and 560 columns. The resulting rank map for the Z^{11} sun-matrix is shown in (12). Again the blocks labeled with rank 00 are near-interactions and have full rank. In this specific case the sub-partitions have around 150 volume elements each, so they will not be partitioned further. The total compression was $60\times$ for this specific example.

The second example is shown in Figure 3. The geometry consists of three conducting coils, the center coil is driven with an independent current source, and we wish to compute the eddy currents in the coils due to the B-field in the surrounding air. The problem consists of 20736 volume elements and was partitioned for 24 parallel processors, therefore the Biot-Savart matrix is a 24×24 block matrix. The parallel rank map for this is too large to show here, but the results were as follow: Each processor had 24 matrices to compute at the highest level, on average 19 of these corresponded to well-separated regions and were compressed with an average rank of 10. The remaining 5 full-rank matrices were further partitioned into $4 \cdot 8 = 32$ sub-matrices, and on average 29 of these corresponded to well-separated regions and were compressed with an average rank of 25. At the lowest level of the hierarchy, the near interactions were represented, on average, by dense matrices of dimension 335×439, there were a total of $24 \cdot 3 = 72$ of these. The total compression was $109\times$. This compression represents both the memory savings and the reduction in CPU time required to apply the Biot-Savart interaction.

$$\begin{bmatrix}
00 & 14 & 16 & 00 & 12 & 10 & 5 & 8 & 5 & 4 & 12 & 20 & 12 & 7 & 7 & 7 \\
5 & 6 & 6 & 5 & 8 & 12 & 00 & 21 & 00 & 21 & 6 & 4 & 9 & 11 & 18 & 11 \\
19 & 00 & 21 & 00 & 00 & 17 & 7 & 12 & 11 & 7 & 00 & 12 & 24 & 12 & 10 & 10 \\
11 & 00 & 12 & 12 & 00 & 00 & 11 & 00 & 15 & 9 & 17 & 7 & 13 & 14 & 12 & 19 \\
6 & 13 & 11 & 9 & 16 & 00 & 14 & 00 & 00 & 12 & 6 & 12 & 16 & 14 & 23 & 5 \\
6 & 6 & 5 & 8 & 12 & 00 & 21 & 00 & 21 & 6 & 4 & 9 & 11 & 18 & 11 & 5 \\
9 & 7 & 7 & 10 & 34 & 00 & 00 & 00 & 15 & 8 & 5 & 9 & 12 & 18 & 17 & 21 \\
12 & 00 & 16 & 9 & 9 & 4 & 8 & 5 & 4 & 12 & 00 & 15 & 9 & 6 & 8 & 19 \\
16 & 00 & 17 & 17 & 11 & 5 & 9 & 6 & 6 & 45 & 00 & 44 & 12 & 8 & 10 & 13 \\
00 & 12 & 14 & 00 & 00 & 9 & 23 & 12 & 8 & 25 & 9 & 16 & 11 & 11 & 13 & 11 \\
21 & 21 & 12 & 24 & 11 & 8 & 12 & 12 & 9 & 00 & 12 & 00 & 00 & 12 & 29 & 10 \\
14 & 12 & 10 & 16 & 19 & 9 & 17 & 12 & 10 & 00 & 9 & 00 & 00 & 15 & 00 & 5 \\
8 & 5 & 7 & 7 & 11 & 20 & 12 & 15 & 00 & 6 & 5 & 8 & 12 & 00 & 18 & 6 \\
9 & 11 & 8 & 10 & 18 & 9 & 22 & 14 & 16 & 13 & 6 & 12 & 00 & 45 & 00 & 5 \\
8 & 7 & 6 & 10 & 11 & 15 & 20 & 16 & 00 & 9 & 4 & 11 & 37 & 00 & 00 & 15 \\
38 & 00 & 16 & 00 & 16 & 11 & 7 & 11 & 10 & 7 & 00 & 15 & 00 & 14 & 12 & 11
\end{bmatrix} \tag{11}$$

$$\begin{bmatrix}
00 & 11 & 19 & 38 & 00 & 12 & 31 & 21 \\
23 & 16 & 21 & 16 & 00 & 36 & 17 & 27 \\
26 & 00 & 35 & 00 & 14 & 23 & 43 & 33 \\
26 & 13 & 00 & 42 & 41 & 13 & 00 & 00
\end{bmatrix} \tag{12}$$

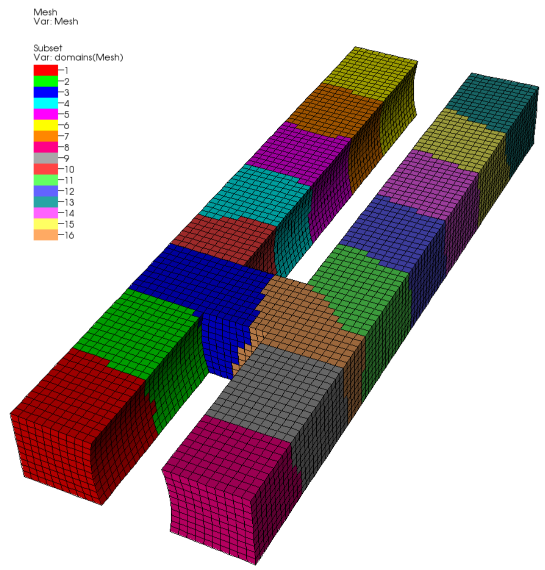

Figure 2: Computational mesh for a linear induction motor partitioned for 16 parallel processors.

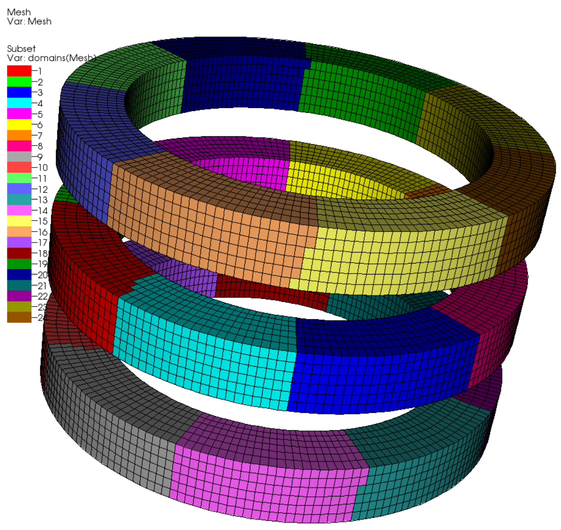

Figure 3: Computational mesh for an inductive coupling application partitioned for 24 parallel processors.

If the computational mesh were refined, the well-separated interactions would still have the same rank, hence the cost of the well-separated interactions is $O(N)$. Each near-interaction is recursively decomposed into $4 \cdot 8 = 32$ interactions, most of which are again well-separated and have low-rank. The dominant cost is the near-interactions which are represented as dense $m \times n$ matrices, where m and n are determined by fixed parameters (e.g. the recursion halting parameter of 512 elements) which are independent of the global dimensions M and N. The number of near interactions is, asymptotically, $O(N\,log(N))$, hence the overall method is $O(N\,log(N))$.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

REFERENCES

1. J. D. Jackson. *Classical Electrodynamics*. 1962.
2. S. Kapur and D. Long. IES3: Efficient electrostatic and electromagnetic solution. *IEEE Comp.. Sci. Eng.*, 5(4):60–67, 1998.
3. D. Gope and V. Jandhyala. PILOT: A fast algorithm for enhanced 3d parasitic capacitance extraction. *Micro. Opt, tech. Lett.*, 41(3):169–173, 2004.
4. D. Gope and V. Jandhyala. Efficient solution of EFIE via low-rank compression of multilevel predetermined interactions. *IEEE Trans. Ant. Prop.*, 53(10):3324–3333, 2005.
5. M. A. Khaya and D. R. Wilton. Numerical evaluation of singular and near-signular potential integrals. *IEEE Trans. Ant. Prop.*, 53(10):3180–3190, 2005.
6. P. Castillo, J. Koning, R. Rieben, and D. White. A discrete differential forms framework for computational electromagnetics. *Computer Modeling in Engineering & Sciences*, 5(4):331–346, 2004.
7. R. Rieben and D. White. Verification of high-order mixed FEM solution of transient magnetic diffusion problems. *IEEE Trans. Mag.*, October 2005. article in press.
8. G. Karypis and V. Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *J. Parallel Distr. Comp.*, 48(1):71–95, 1998.