A Multilevel Iterative Field Solver for Implicit Plasma Simulation Codes

D.A. Knoll †, G. Lapenta *, and J.U. Brackbill

Los Alamos National Laboratory

Los Alamos, NM 87545

† M.S. D413, (505) 667-7467, nol@lanl.gov

* Permanent address: Dipartimento di Energetica, Politecnico di Torino

1 Introduction

Much effort has gone into the development of implicit methods for electromagnetic kinetic plasma simulation. One such implicit method is the implicit moment method [1]. The maximum allowable time step in this method has often been found to be governed by the iterative elliptic field solver, and not the implicit moment method itself [2]. In ref. [2] it was shown, in one dimension, that by using a direct method to solve the field equations, significantly larger time steps could be achieved. We have initiated research into the development of an efficient and robust iterative elliptic solver which will allow similar time steps, and fine grid simulations, in two and three dimensions. Our algorithm uses the multigrid method [3] as a preconditioner to a Krylov subspace based iterative method, GMRES [4] in this work.

Our goal is application of the proposed algorithm to three dimensional fully electromagnetic simulation. As a starting point we will work on the CELEST2D code, an unpublished modification of CELEST1D [2]. CELEST2D is electrostatic but contains a spatially varying (time independent) magnetic field. Indeed, it is the electrostatic potential equation, from the implicit moment method, which provides the greatest challenge to iterative solvers. The elliptic equation for electrostatic potential, ϕ, in the implicit moment method of CELEST2D is,

$$\nabla \cdot [(\mathbf{I} + \mu(\vec{r})) \cdot \nabla\phi] = \rho(\vec{r}). \tag{1}$$

Here, $\vec{r}$ is the position vector, $\mu(\vec{r})$ is a spatially varying electric susceptibility tensor, and $\rho(\vec{r})$ is the space charge. The functional form of $\mu(\vec{r})$ in CELEST2D is identical to that presented in [2].

The potential equation is discretized using finite volumes, in a general coordinate system, which produces a 9-point numerical stencil [5]. The major difficulty arises from the fact that the off diagonal components of $\mu(\vec{r})$, which are proportional to the simulation time step δt, cause the resulting matrix equation for ϕ,

$$\mathbf{A}\phi = \rho \tag{2}$$

to be non-symmetric. Not only is $\mathbf{A}$ non-symmetric, but in some cases diagonal dominance may be lost.

2 Multilevel Iterative Solver

The base solver for Eq.(2) in CELEST2D is a preconditioned Krylov method. The Krylov method is GMRES [4] since the matrix equation is non-symmetric. The base preconditioner is a Jacobi iteration, otherwise refered to as diagonal scaling. Preconditioned Krylov methods are modern, robust, general elliptic solvers, and they can be implemented in a matrix-free fashion (linear or nonlinear). However, they do not scale well with grid refinement. By this we mean that as the number of grid points increases, the number of iteration, to achieve the same level of convergence, also increases.

Multigrid methods [3] promise optimal scaling with problem size. By this it is meant that the number of iterations required, for a given convergence tolerance, is independent of grid dimension. However, developing a robust multigrid method for "realistic" problems is a challenge, and can be very time consuming, often requiring significant problem specific tuning.

Here we present results from a hybrid approach which uses "simple" multigrid methods as a preconditioner to the Krylov solver. By "simple" we mean that nothing has been done to tune a standard multigrid algorithm to work efficiently for Eq. (1). Furthermore, straight forward, operator independent, inter-grid interpolation is used. It is shown that this hybrid approach has produced a robust, and scalable (efficient), elliptic solver for Eq. (1).

To define a multigrid preconditioner one must define restriction, $\mathcal{R}$, and prolongation, $\mathcal{P}$, operators, and a method for constructing the required coarse grid matrices. In the multigrid method, $\mathcal{R}$ is used to transfer residuals (i.e. $\rho - \mathbf{A}\phi^*$) down to a coarse grid, and $\mathcal{P}$ is used to transfer solution vector corrections (i.e. $\delta\phi$) up to a fine grid. In this study, piece-wise constant interpolation is used for both $\mathcal{R}$ and $\mathcal{P}$, with $\mathcal{R}$ being volume weighted. This is a very simple choice and most likely will not produce an optimal multigrid method [6]. Using a piece-wise

linear $\mathcal{P}$ may produce a near optimal multigrid method [6], and this will be investigated in the future.

Next, a method for calculating the coarse grid representation of $\mathbf{A}$ needs to be defined. There are at least two distinct options. The first is to restrict the physics ($\mu(\vec{r})$ in this case) to the coarse grid and re-discretize Eq. (1). A second approach is to use the restriction and prolongation operators, along with the fine grid linear operator, to construct a coarse grid operator. This would come from

$$\mathbf{A}_c = \mathcal{R}\mathbf{A}_f\mathcal{P}. \tag{3}$$

Here $\mathbf{A}_c$ is the $(N/2) \times (N/2)$ coarse grid operator and $\mathbf{A}_f$ is the $N \times N$ fine grid operator. This approach is often refered to as a Galerkin coarse grid operator [6] and is related to the Black Box multigrid concept [7]. In this study the second approach is employed.

Finally, a smoother for the multigrid algorithm must be chosen. We will use a multi-sweep damped Jacobi iteration where the damping coefficient may change from sweep to sweep. The preconditioner will be one V-cycle [3] with an equal number of pre- and post- smoothing steps (sweeps), a V(ν, ν) cycle. Traditionally, on the coarsest grid, a direct solve is used, but that is not done here.

3 Performance Results

For the first performance result the single grid Jacobi based preconditioner is compared to the multigrid Jacobi based preconditioner. The number of sweeps is 3 ($\nu = 3$) and the damping coefficient is 2/3, for both the single grid preconditioner and the multigrid smoother. Five GMRES vectors are stored and restart is employed with a linear convergence tolerance of 1.0×10^{-6}. This problem is on a uniform grid and the boundary conditions on ϕ are Dirichlet on three sides and Neumann on the fourth side. Table 1 contains performance data as a function of grid refinement for $32 \times 32, 64 \times 64, 128 \times 128$ and a 256×256 grid. This data was averaged over 5 time steps, with the same time step on each cycle and on each grid.

It is quite apparent that the multigrid preconditioner has not only significantly reduced the required number of GMRES iterations, it has also decreased the growth of required iterations as a function of grid refinement. In fact, the number of required GMRES iterations appears to be independent of the grid dimension.

4 Conclusions

The use of a simple multigrid method as a preconditioner to GMRES appears to make a robust and efficient method for the elliptic field solver required in implicit plasma simulation codes, on the uniform grid problem considered. Next we will investigate retaining more than five GMRES vectors. We will also investigate performance on a nonuniform grid.

Acknowledgments

This work was supported under the auspices of the U.S. Department of Energy under DOE contract W-7405-ENG-36 at Los Alamos National Laboratory.

References

[1] J.U. Brackbill and D.W. Forslund. *J. Comput. Phys.*, 46:271, 1982.

[2] H.X. Vu and J.U. Brackbill. *Comput. Phys. Comm.*, 69:253, 1992.

[3] A. Brandt. *Math. Comp.*, 31:333, 1977.

[4] Y. Saad and M.H. Schultz. *SIAM J. Sci. Stat. Comput.*, 7:856, 1986.

[5] D. Sulsky and J.U. Brackbill. *J. Comput. Phys.*, 96:339, 1991.

[6] Pieter Wesseling. *An Introduction to Multigrid Methods.* John Wiley & Sons, Chichester, 1992.

[7] J.E. Dendy. *J. Comput. Phys.*, 48:366, 1982.

Table 1: Algorithm performance as a function of grid refinement: Average number of GMRES iterations per time step, and ratio of CPU time.

Grid	Single Grid (SG) Precond.	Multigrid (MG) Precond	CPU ratio SG / MG
32×32	48	8.7	1.8
64×64	116	9	2.8
128×128	398	11	5.65
256×256	691	9	10.5