A Parallel Three-dimensional Electromagnetic Particle-in-Cell Code for Non-Orthogonal Meshes

S.R. Karmesin, P.C. Liewer, and J. Wang

CRPC-TR96731
September 1996

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Submitted November 1997

A Parallel Three-dimensional Electromagnetic Particle-in-Cell Code for Non-Orthogonal Meshes

S. R. Karmesin *
California Institute of Technology, Pasadena CA 91125
P. C. Liewer and J. Wang
Jet Propulsion Laboratory, California Institute of Technology, Pasadena CA 91109

September 26, 1996

Abstract

We describe a new parallel three dimensional electromagnetic particle-in-cell code that uses body fitted curvilinear coordinates for modeling plasma and beam devices. Cells in the structure grid are deformable hexahedra in physical space and are mapped to unit cubes in logical space for particle interpolations. The algorithms conserve particle charge and current, and update the electromagnetic fields in a divergence preserving manner. The code is modular and portable, and we present numerical results of convergence rates and benchmarks on serial, vector and parallel computers for the components separately and together.

1 Introduction

An electromagnetic particle-in-cell (EMPIC) code seeks to simulate a plasma or charged particle beam through a direct simulation of the evolution of the electromagnetic fields and the charged particle positions and velocities. The overall accuracy of the code depends on (1) how well it models the geometry of the problem, (2) the algorithms it uses to update the quantities, and (3) how many grid-points and particles it can manage.

In this paper we describe how we are pulling together techniques to construct a code that addresses these three questions in order to model high power microwave devices. We have constructed a code which runs EMPIC simulations on serial and massively parallel computers and allows studies of the accuracy and efficiency of the various techniques to be explored.

*Now at K2 Research Inc., Pasadena, CA 91101

We have not yet addressed such issues as grid generation, data visualization, error-checking and user interface that would be required for a real engineering tool.

Because the goal is to simulate devices with curved surfaces, we use body-fitted coordinates in three dimensions, and because we wish to model them accurately we use smoothly varying curvilinear coordinates. This avoids the Cartesian grid problem of having the grid stair-step along curved surfaces, reducing the number of grid points necessary for a given level of accuracy. And because complex microwave devices can have several connected cavities and regions, we organize the grid in patches, connected by appropriate boundary conditions.

The algorithm we use to update the electromagnetic field quantities $\mathbf{E}$ and $\mathbf{B}$ is due to Gedney and Lansing [1]. This algorithm is an extension to non-uniform meshes of the classical staggered mesh Finite-difference time-domain (FDTD) algorithm and is similar to that of Madsen [2]. We also extend the method of Villasenor and Buneman [3] to non-uniform meshes in order to calculate the current $\mathbf{J}$ on the grid from the particles while preserving to machine precision the divergence condition on $\mathbf{E}$. We present here convergence studies for these methods.

Because high resolution simulations in three dimensions are very resource intensive, we have built the code from the beginning to scale up to massively parallel computers while allowing it to be used on serial computers for test problems and for development. We present benchmarks and scaling studies to show that it scales efficiently from personal computers to massively parallel supercomputers. The parallel decomposition used is a domain decomposition as used in the parallel three-dimensional Cartesian grid electromagnetic PIC code of Wang et. al[4] and the reader is referred to this paper for more background on parallel 3D EMPIC codes. Eastwood et. al [5] have also developed a sophisticated parallel 3D EMPIC code which uses body-fitted coordinates.

2 Physics Model

The electromagnetic fields satisfy Maxwell's equations with the current derived from the motion of charged particles. In differential form the curl and divergence constraint equations are

$$\begin{aligned} \partial_t \mathbf{B} &= -c\nabla \times \mathbf{E} \\ \partial_t \mathbf{E} &= c\nabla \times \mathbf{B} + J \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \cdot \mathbf{E} &= \rho \\ \partial_t \rho &= -\nabla \cdot J. \end{aligned} \tag{1}$$

For numerical solution on non-orthogonal grids, the curl equations are recast in integral form by integrating curl equations over a surface S,

$$\partial_t \int_s \mathbf{B} \cdot \mathbf{ds} = -c \oint_l \mathbf{E} \cdot \mathbf{dl} \tag{2}$$

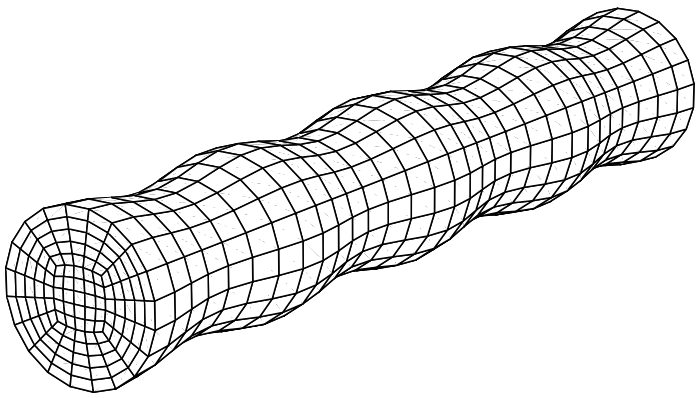

Figure 1: Cylindrical domain with cross section broken into five logically Cartesian patches to avoid the coordinate singularity at the cylinder access. The domain can also be divided along the axis. Each patch is logically Cartesian and there is a one-to-one mapping of faces and vertices across patch boundaries.

$$\partial_t \int_s \mathbf{E} \cdot \mathbf{ds} \;=\; c \oint_l \mathbf{B} \cdot \mathbf{dl} + \int_s \mathbf{J} \cdot \mathbf{ds}$$

and the the divergence equations and the conservation of charge equation over a volume V,

$$\begin{aligned} \oint_s \mathbf{E} \cdot \mathbf{ds} &= \int_v \rho dv \\ \oint_s \mathbf{B} \cdot \mathbf{ds} &= 0 \\ \partial_t \int \rho \cdot dv &= \int \mathbf{J} \cdot \mathbf{ds} \end{aligned} \tag{3}$$

In the usual PIC approximation, the particles in the simulation represent many electrons or ions, and these "macroparticles" obey the same equations of motion as a single charged particle with charge q and relativistic mass $\gamma m (\gamma = (1 - v^2/c^2)^{-1/2})$ in an electromagnetic field,

$$\begin{aligned} d_t \gamma m \mathbf{v} &= q(\mathbf{E} + (\mathbf{v}/c) \times \mathbf{B}) \\ d_t \mathbf{x} &= \mathbf{v} \end{aligned} \tag{4}$$

3 Numerics Description

The electromagnetic field portion of our code is based on the algorithm of Gedney and Lansing [1], specialized to the case of a structured grid of hexahedral cells. This is a discrete volume generalization of the standard FDTD algorithm and reduces identically to this algorithm if the grid is Cartesian. The particle update uses the current deposit formulation

Figure 2: Each hexahedral cell in configuration space is mapped to a unit cube in logical space. A simple tri-linear interpolation maps the logical coordinates to the physical coordinates.

of Villasenor and Buneman [3] generalized to non-orthogonal meshes, and the usual electromagnetic leap frog time step described in, for example, Birdsall and Langdon [6] and Wang et. al [4].

3.1 Code Geometry and Grid

We denote by Ω the domain on which the equations of motion of the fields and particles shall be computed. We then break Ω down into a set of nonoverlapping patches Ω_p such that each patch can be represented with three dimensional Cartesian curvilinear coordinates. We call these logical coordinates and denote them with the variables $\mathbf{r} = (r_1, r_2, r_3)$, and we define a mapping from $\mathbf{r}$ to the usual physical space coordinates $\mathbf{x} = (x, y, z)$. We allow the domain to be broken into several patches before laying down logical coordinates in order to allow domains like Fig. 1 to be represented without coordinate singularities. The patches are sewn together numerically by specifying internal boundary condition on the faces of Ω_p.

Each patch is discretized into a structured grid of hexahedral cells along the three coordinate directions. Since each patch is logically Cartesian, neighbors can be found by simply incrementing indices and the complications of indirect addressing of unstructured grids are avoided. We require that the discretization of the patches be conforming in that the faces and vertices of cells on the boundaries between patches must match one-to-one. We are free to choose the units in logical coordinates, and it is convenient to make the logical grid spacing unity. (In the remainder we shall consider a single patch unless specifically noted otherwise.) The vertices of these cells are then at the locations where the logical coordinates $\mathbf{r}$ are all integers. A simple tri-linear interpolation is used to map the logical coordinates to the physical coordinates. The mapping of cells from physical to logical space is illustrated in Fig. 2. These cells form what we call the primary or $\mathbf{B}$ grid because we will locate the magnetic field on its faces. We also define a dual or staggered grid (the secondary or $\mathbf{E}$ grid) to be the one with vertices where $\mathbf{r}$ are all half integers. The edges of the $\mathbf{E}$ grid go through the faces of the $\mathbf{B}$ grid, and visa versa. We will locate $\mathbf{E}$ on the faces of this grid. (It is

Figure 3: Location of electromagnetic field variables on the non-orthogonal mesh. The $\mathbf{B} \cdot \mathbf{ds}$ are located on the faces of the $\mathbf{B}$ grid and the $\mathbf{E} \cdot \mathbf{ds}$ are located on the faces of the $\mathbf{E}$ grid. The discrete line integral $\mathbf{B} \cdot \mathbf{dl}$ is located on edges of the $\mathbf{E}$ grid.

possible, however, that for extremely non-orthogonal grids, the edge will not pass through the face and this should be avoided [2].

3.2 Electromagnetic Field Update

The fundamental variables that our electromagnetic simulation code calculates are $\mathbf{B} \cdot \mathbf{ds}$ (on each face of $\mathbf{B}$ grid cells) and $\mathbf{E} \cdot \mathbf{ds}$ (on each face of $\mathbf{E}$ cells). These are the quantities that have time derivatives specified on the left hand side of Eqs. 2 and they appear in the constraint equations Eqs. 3. The integrals in Eqs. 2 are taken to be over a single face of a $\mathbf{B}$ or $\mathbf{E}$ cell, and in Eqs. 3 is taken to be an integral over all six faces of a $\mathbf{B}$ or $\mathbf{E}$ cell. The current $\mathbf{J} \cdot \mathbf{ds}$ will be co-located with $\mathbf{E} \cdot \mathbf{ds}$; we consider the current to be zero for this section.

The right hand sides of Eqs. 2 involve fields dotted with line segments ($\mathbf{dl}$) rather than face normals (parallel to $\mathbf{dS}$). We locate $\mathbf{B} \cdot \mathbf{dl}$ on the edges of the $\mathbf{E}$ grid and they pass through faces of the $\mathbf{B}$ grid. We locate $\mathbf{E} \cdot \mathbf{dl}$ on the edges of the $\mathbf{B}$ grid and they pass through the faces of the $\mathbf{E}$ grid. The placement of these quantities is shown in Fig. 3 . When the grid is Cartesian, the edges and the face normals are colinear; in the general case, they are not colinear as illustrated in Fig. 3. These locations reduce to the familiar grid staggering of the Yee lattice for Cartesian FDTD codes [7].

To give unique labels to the various grid quantities, we label each with the logical coordinates of its center, e.g. a cell is labeled by the coordinates of its center and a $\mathbf{B} \cdot \mathbf{ds}$ by the coordinates of the face center. Vertices of the $\mathbf{B}$ grid and cells of the $\mathbf{E}$ grid are integer triplets (since we chose to have unit spacing in logical space) and all other quantities have 1, 2 or 3 half integer coordinates. Faces and face variables must have a subscript 1, 2 or 3 for the logical coordinate direction of the normal (replacing the x, y, z subscripts in a Cartesian grid). Likewise, edges have a subscript of 1, 2, or 3 for the logical coordinate direction of the edge vector. Thus the $\mathbf{B} \cdot \mathbf{ds}$ component in direction 1 located on the face whose center is at $\mathbf{r} = (i, j + 1/2, k + 1/2)$ is labeled $\mathbf{B} \cdot \mathbf{ds}_{1,i,j+1/2,k+1/2}$.

To map these quantities to array locations we simply drop any half integers in the coordinates. For simplicity, below we suppress the subscript on quantities that are at location (i, j, k), and for quantities at nearby gridpoints we include only the coordinates that are offset. For example we write $\mathbf{E} \cdot \mathbf{dl}_{3,i,j+1,k}$ as $\mathbf{E} \cdot \mathbf{dl}_{3,j+1}$. With this notation, the spatially discretized Maxwell's equations are then:

$$\begin{aligned}
d_t\mathbf{B} \cdot \mathbf{ds}_1 &= \mathbf{E} \cdot \mathbf{dl}_{2,k+1} - \mathbf{E} \cdot \mathbf{dl}_2 + \mathbf{E} \cdot \mathbf{dl}_3 - \mathbf{E} \cdot \mathbf{dl}_{3,j+1} \\
d_t\mathbf{B} \cdot \mathbf{ds}_2 &= \mathbf{E} \cdot \mathbf{dl}_{3,i+1} - \mathbf{E} \cdot \mathbf{dl}_3 + \mathbf{E} \cdot \mathbf{dl}_1 - \mathbf{E} \cdot \mathbf{dl}_{1,k+1} \\
d_t\mathbf{B} \cdot \mathbf{ds}_3 &= \mathbf{E} \cdot \mathbf{dl}_{1,j+1} - \mathbf{E} \cdot \mathbf{dl}_1 + \mathbf{E} \cdot \mathbf{dl}_2 - \mathbf{E} \cdot \mathbf{dl}_{1,i+1} \\
d_t\mathbf{E} \cdot \mathbf{ds}_1 &= \mathbf{B} \cdot \mathbf{dl}_{2,k-1} - \mathbf{B} \cdot \mathbf{dl}_2 + \mathbf{B} \cdot \mathbf{dl}_3 - \mathbf{B} \cdot \mathbf{dl}_{3,j-1} \\
d_t\mathbf{E} \cdot \mathbf{ds}_2 &= \mathbf{B} \cdot \mathbf{dl}_{3,i-1} - \mathbf{B} \cdot \mathbf{dl}_3 + \mathbf{B} \cdot \mathbf{dl}_1 - \mathbf{B} \cdot \mathbf{dl}_{1,k-1} \\
d_t\mathbf{E} \cdot \mathbf{ds}_3 &= \mathbf{B} \cdot \mathbf{dl}_{1,j-1} - \mathbf{B} \cdot \mathbf{dl}_1 + \mathbf{B} \cdot \mathbf{dl}_2 - \mathbf{B} \cdot \mathbf{dl}_{1,i-1}.
\end{aligned} \tag{5}$$

Note that this set of equation in not complete until until we specify how the edge quantities, the $\mathbf{E} \cdot \mathbf{dl}$ and the $\mathbf{B} \cdot \mathbf{dl}$, are determined from the $\mathbf{E} \cdot \mathbf{ds}$ and the $\mathbf{B} \cdot \mathbf{ds}$ respectively.

The spatially discretized constraint equations are

$$\begin{aligned}
0 &= \mathbf{B} \cdot \mathbf{ds}_1 - \mathbf{B} \cdot \mathbf{ds}_{1,i+1} + \mathbf{B} \cdot \mathbf{ds}_2 - \mathbf{B} \cdot \mathbf{ds}_{2,j+1} + \mathbf{B} \cdot \mathbf{ds}_3 - \mathbf{B} \cdot \mathbf{ds}_{3,k+1} \\
0 &= \mathbf{E} \cdot \mathbf{ds}_1 - \mathbf{E} \cdot \mathbf{ds}_{1,i-1} + \mathbf{E} \cdot \mathbf{ds}_2 - \mathbf{E} \cdot \mathbf{ds}_{2,j-1} + \mathbf{E} \cdot \mathbf{ds}_3 - \mathbf{E} \cdot \mathbf{ds}_{3,k-1}
\end{aligned} \tag{6}$$

If we substitute the discretized Maxwell's equations into these equations we find that they hold automatically, and this is the reason for staggering the $\mathbf{E}$ and $\mathbf{B}$ grids. If the initial condition satisfies the constraints, the evolved solution will also.

Note that the constraints hold no matter how edge values ($\mathbf{E} \cdot \mathbf{dl}$ and $\mathbf{B} \cdot \mathbf{dl}$) are calculated from face values ($\mathbf{E} \cdot \mathbf{ds}$ and $\mathbf{B} \cdot \mathbf{ds}$). Various choices are possible for moving quantities from faces to edges. The accuracy of the computation is undoubtedly very sensitive to the choices here. We plan to investigate other choices in the future.

The system is then time discretized by staggering $\mathbf{E}$ and $\mathbf{B}$ by $dt/2$ and leapfrogging $\mathbf{E}$ and $\mathbf{B}$ over each other with time step dt (as in the standard FDTD scheme), which clearly preserves the constraint relations. (Specifically, we define the electric field and current on whole time steps and the magnetic field on the half time steps.) Leapfrog time discretization is used for second order convergence in time.

3.3 Obtaining $\mathbf{B} \cdot \mathbf{dl}$ from $\mathbf{B} \cdot \mathbf{ds}$

To close the field equations we must define how to calculate $\mathbf{B} \cdot \mathbf{dl}$ from $\mathbf{B} \cdot \mathbf{ds}$. Calculating $\mathbf{E} \cdot \mathbf{dl}$ from $\mathbf{E} \cdot \mathbf{ds}$ done in exactly the same way, so the discussion will focus on $\mathbf{B}$. For a uniform orthogonal grid, $\mathbf{ds}$ and $\mathbf{dl}$ are parallel and we can calculate $\mathbf{B} \cdot \mathbf{dl} = \mathbf{B} \cdot \mathbf{ds}|\mathbf{dl}|/|\mathbf{ds}|$, making Eqs. 2 a closed system. This forms the same expressions as discretizing Eqs. 1 with centered differences, as in the FDTD algorithm, and in this limit, the method gives second order accuracy in space. There is no real distinction between face and edge quantities.

For nonorthogonal coordinates, $\mathbf{dl}$ is not parallel to $\mathbf{ds}$, and converting face to edge quantities is more involved. In our code, we have used the technique due to Gedney and

Figure 4: The steps to find $\mathbf{B} \cdot \mathbf{dl}$ from $\mathbf{B} \cdot \mathbf{ds}$. First, the vertex vector $\mathbf{B}^{123}$ is obtained from the $\mathbf{B} \cdot \mathbf{ds}$ of the faces labeled 1, 2 and 3 by the method described in the text and similarly for the other vertices of the face. In the second step, the $\mathbf{B} \cdot \mathbf{dl}$ for the line segment from face 2 to the center of the cell is obtained from the vectors associated with the four vertices of face 2.

Lansing [1] specialized to the case of hexahedral cells. This technique is similar, but not identical, to the Discrete Surface Integral method of Madsen [2].

Figure 4 shows the two steps in obtaining a single $\mathbf{B} \cdot \mathbf{dl}$ from a cell center to a cell face from the $\mathbf{B} \cdot \mathbf{ds}$ for that face and 4 others in the same cell. (In the figure, the quantities are labeled by cell face, not logical coordinate.) The first step in obtaining a $\mathbf{B} \cdot \mathbf{dl}$ for a face is to find vertex values for $\mathbf{B}$ for the four vertices of this face. A cell vertex value is obtained from the 3 cell faces sharing this vertex by solving a set of 3 coupled equations. In the figure $\mathbf{B}^{123}$, the cell vertex value at the vertex shared by faces 1, 2 and 3, is found by solving

$$\begin{aligned} \mathbf{B}^{123} \cdot \mathbf{n}_1 &= \mathbf{B} \cdot \mathbf{ds}_1 \\ \mathbf{B}^{123} \cdot \mathbf{n}_2 &= \mathbf{B} \cdot \mathbf{ds}_2 \\ \mathbf{B}^{123} \cdot \mathbf{n}_3 &= \mathbf{B} \cdot \mathbf{ds}_3 \end{aligned} \tag{7}$$

where $\mathbf{n}_i$ is the unit normal to face i. Once these cell vertex values are found, there are many choices of how to proceed to obtain the dual edge quantity $\mathbf{B} \cdot \mathbf{dl}$. We choose to follow Gedney and Lansing [1] and this is illustrated in the second step in Figure 4. After the above computation is done for all vertices of the cell, we average four vertex values to give a $\mathbf{B}$ at the face center using a volume weighted average and this value is dotted with the dual edge vector from the face center to the cell center ($\mathbf{dl}_2$ for face 2),

$$\mathbf{B} \cdot \mathbf{dl} = \mathbf{B}^{123} w^{123} \cdot \mathbf{dl}_2 + \mathbf{B}^{234} w^{234} \cdot \mathbf{dl}_2 + \mathbf{B}^{125} w^{125} \cdot \mathbf{dl}_2 + \mathbf{B}^{245} w^{245} \cdot \mathbf{dl}_2. \tag{8}$$

The weights are calculated from the volume associated with each corner (calculated from the cross product of the edge vectors). Note that $\mathbf{B} \cdot \mathbf{ds}$ from 5 faces of the cell were used to compute one $\mathbf{B} \cdot \mathbf{dl}$. At this stage, we have only the portion of the path integral $\mathbf{B} \cdot \mathbf{dl}$ from the cell center to the face center. We now add to this the $\mathbf{B} \cdot \mathbf{dl}$ computed from the neighboring cell on the other side of this face (from the face center to the cell center of the neighboring cell). This sum of the $\mathbf{B} \cdot \mathbf{dl}$ along the two line segments from cell center to cell

face is an approximation to a $\mathbf{B} \cdot \mathbf{dl}$ computed along the $\mathbf{E}$ grid edge connecting the two cells centers. By computing it in this way, the $\mathbf{E} \cdot \mathbf{dl}$ can be computed for each cell locally and no interprocessor communication is necessary. Alternatives to this approximation will be investigated in the future.

In practice, the calculation of the $\mathbf{B} \cdot \mathbf{dl}$'s is done in the code by precalculating the coefficients that connect all the $\mathbf{B} \cdot \mathbf{ds}$ values with each $\mathbf{B} \cdot \mathbf{dl}$ (9 faces in 3D) and performing a single sparse matrix vector product to obtain all the $\mathbf{B} \cdot \mathbf{dl}$'s from the $\mathbf{B} \cdot \mathbf{ds}$'s.

This update is formally only first order because information from different face locations has been combined to obtain a single vertex value and convergence tests presented below indicate this. Since the algorithm is known to be second order accurate on an orthogonal grid, it is important to minimize the regions of the grid that are strongly nonorthogonal.

3.4 Conducting Boundary conditions

In the interior of the simulation the $\mathbf{B}$ and $\mathbf{E}$ grids are dual to each other. We choose to align them so that the faces of the $\mathbf{B}$ grid lie on the boundaries of Ω to simplify the boundary conditions. The conditions at the surface of a perfect conductor are that $\mathbf{B} \cdot \mathbf{n} = 0$ and $\mathbf{E} \times \mathbf{n} = 0$. The first is easily applied by simply setting $\mathbf{B} \cdot \mathbf{ds} = 0$ for faces on a conductor, and the second by setting $\mathbf{E} \cdot \mathbf{dl} = 0$ for the edges on the conductor. If the $\mathbf{E}$ grid is orthogonal at the boundary this may be accomplished by setting $\mathbf{E} \cdot \mathbf{ds} = 0$ for the $\mathbf{E}$ faces that penetrate the conductor. If the grid is not orthogonal at the boundary, it is generally necessary to solve a matrix equation over the boundary surface to determine the set of $\mathbf{E} \cdot \mathbf{ds}$ that will give $\mathbf{E} \cdot \mathbf{dl} = 0$.

3.5 Currents

Currents can be included in the discretization by including the current on the right hand side of Eqs. 5 to give a modified update for the $\mathbf{E}$ field:

$$\begin{aligned} d_t \mathbf{E} \cdot \mathbf{ds}_1 &= \mathbf{B} \cdot \mathbf{dl}_{2,k-1} - \mathbf{B} \cdot \mathbf{dl}_2 + \mathbf{B} \cdot \mathbf{dl}_3 - \mathbf{B} \cdot \mathbf{dl}_{3,j-1} + \mathbf{J} \cdot \mathbf{ds}_1 \\ d_t \mathbf{E} \cdot \mathbf{ds}_2 &= \mathbf{B} \cdot \mathbf{dl}_{3,i-1} - \mathbf{B} \cdot \mathbf{dl}_3 + \mathbf{B} \cdot \mathbf{dl}_1 - \mathbf{B} \cdot \mathbf{dl}_{1,k-1} + \mathbf{J} \cdot \mathbf{ds}_2 \\ d_t \mathbf{E} \cdot \mathbf{ds}_3 &= \mathbf{B} \cdot \mathbf{dl}_{1,j-1} - \mathbf{B} \cdot \mathbf{dl}_1 + \mathbf{B} \cdot \mathbf{dl}_2 - \mathbf{B} \cdot \mathbf{dl}_{1,i-1} + \mathbf{J} \cdot \mathbf{ds}_3. \end{aligned} \tag{9}$$

The discretized constraint equation on $\mathbf{J}$ is

$$d_t \rho dV = \mathbf{J} \cdot \mathbf{ds}_1 - \mathbf{J} \cdot \mathbf{ds}_{1,i+1} + \mathbf{J} \cdot \mathbf{ds}_2 - \mathbf{J} \cdot \mathbf{ds}_{2,j+1} + \mathbf{J} \cdot \mathbf{ds}_3 - \mathbf{J} \cdot \mathbf{ds}_{3,k+1}. \tag{10}$$

Here ρdV is the total charge in a single cell of the $\mathbf{E}$ grid and $\mathbf{J} \cdot \mathbf{ds}$ is the current crossing a face of the $\mathbf{E}$ grid. As long as $\mathbf{J} \cdot \mathbf{ds}$ and ρdV are related in this way we will maintain the discrete version of the divergence equation for the electric field

$$\rho dV = \mathbf{E} \cdot \mathbf{ds}_1 - \mathbf{E} \cdot \mathbf{ds}_{1,i+1} + \mathbf{E} \cdot \mathbf{ds}_2 - \mathbf{E} \cdot \mathbf{ds}_{2,j+1} + \mathbf{E} \cdot \mathbf{ds}_3 - \mathbf{E} \cdot \mathbf{ds}_{3,j+1} \tag{11}$$

For our EMPIC simulations, ρ is defined by the locations of the particles, and the current $\mathbf{J}$ is calculated from the particle motion.

3.6 Particle Update for Non-Orthogonal Grid

Particles moving through a non-uniform grid are handled similarly to particles in a uniform grid – they move according to Eq. 4 and they deposit current onto the faces of the **E** grid in such a way as to maintain Eq. 3. We briefly describe here the particle time step, elaborating on the parts that pertain to non-uniform geometries. The method is an extension of the scheme used in the Cartesian 3D EMPIC code of Wang et. al [4] and a more detailed treatment can be found there.

In the code, the trajectory of each particle is integrated using a standard time-centering scheme discussed in Birdsall and Langdon [6] and as used in [4] with the particle position defined on integer time steps and the velocity defined on half integer time steps. However, in our non-orthogonal grid code, the particle position is kept in logical space while the velocity is kept in physical space. Setting the particles' fundamental shape and position in logical space simplifies both the force interpolation and the current deposit because the interpolation weights are simple linear functions of each logical coordinate. If the physical location of the particle is required it is easily calculated using the geometry information for each cell. We keep the particle velocity in physical space because it is needed for use in the calculation of the Lorentz force and the current deposit, and this requires a modification to the simple Cartesian leapfrog update of $\mathbf{x}$ and $\mathbf{v}$.

To update the velocity, we solve

$$\frac{\mathbf{u}^{n+1/2} - \mathbf{u}^{n-1/2}}{\Delta t} = \frac{q}{m}(\mathbf{E}^n + \frac{1}{c}\frac{\mathbf{u}^{n+1/2} + \mathbf{u}^{n-1/2}}{2\gamma^n} \times \mathbf{B}^n) \tag{12}$$

where $\mathbf{u} = \gamma\mathbf{v}$.

Note that the primary variables in the field computation (Eqs. 5 and 9) are $\mathbf{B} \cdot \mathbf{ds}$ and $\mathbf{E} \cdot \mathbf{ds}$, whereas we need vector values for **E** and **B**, interpolated to the particle positions, for updating the particle velocities (Eq. 12). In a flat grid EMPIC simulation, the **E** and **B** fields are averaged from their positions on faces to the vertices of the **B** grid. In a distorted grid, this is done by using Eqs. 7 for calculating a cell's **B** at a vertex from $\mathbf{B} \cdot \mathbf{ds}$ and then computing an averaged **B** from averaging the vertex **B**'s from all cells sharing this vertex. **E** on the vertices of the **E** mesh is similarly calculated from $\mathbf{E} \cdot \mathbf{ds}$ and then averaged to the vertices of the **B** mesh. The values of **B** and **E** are then interpolated to particle positions in logical space using linear interpolation.

To update the logical particle positions, we introduce the transformation matrix $M(r) = d\mathbf{r}/d\mathbf{x}(\mathbf{r})$, where $\mathbf{r}$ is the logical position of a particle and $\mathbf{x}$ is its physical position, and it is evaluated at $\mathbf{r}$. We then update $\mathbf{r}$ with predictor corrector scheme

$$\begin{aligned} \mathbf{r}^{n+1/2} &= \mathbf{r}^n + M(\mathbf{r}^n) \cdot \mathbf{v}\Delta t/2 \\ \mathbf{r}^{n+1} &= \mathbf{r}^n + M(\mathbf{r}^{n+1/2}) \cdot \mathbf{v}\Delta t \end{aligned} \tag{13}$$

There are several possible ways to calculate M. In our code, M is calculated at each vertex of the **B** grid once at the start of the computation. It is then interpolated to the particle position in the particle update using linear interpolation in logical space.

An alternative technique would be to calculate $d\mathbf{r}/d\mathbf{x}$ directly at the particle position when needed, but special processing is required for particles that cross a cell boundary during a time step. This method would be significantly more accurate for nonsmooth grids, but those sorts of grids are not our present focus. The tests of particle motion below use the interpolation technique. Presently, the accuracy of the EMPIC code is limited by the accuracy of the EM field algorithm.

3.7 Current Deposit

The last ingredient in the particle time step is a technique for depositing the current. Here, we use the method of Villasenor and Buneman [3] extended to non-orthogonal grids. The current deposit is done by calculating for each particle how much charge crosses each face of the **E** grid as described in [3], which we briefly summarize here. In our non-orthogonal grid code, exactly the same calculation is done, but it is now in logical space using the cells of the **E** grid.

We make the current deposit tractable by defining each particle to be a cube with uniform charge distribution and unit edge length in logical space. The current is deposited to the same location as the **E** field, the centers of **E** grid faces. We deposit the physical current, but do the interpolations in logical space. In a time step, a particle will deposit current to any face which its shape function moves across. It is obvious that a particle "cube" can touch up to eight **E** cells at any given time. Following Villasenor and Buneman[3], if during a time step a particle stays within the same **B** cell, then the same eight **E** cells will be occupied by the particle and the particle will deposit current to the twelve faces that separate those eight **E** cells. If during a time step a particle crosses from one **B** cell to another the path is broken into segments each of which is entirely within a single **B** cell, and the current is deposited in each one. Below we describe the deposit procedure in each **B** cell. Having the time step broken in this manner allows this method to be used without modification when particle are being injected and absorbed at boundaries.

The particle is taken to move uniformly from the logical position $\mathbf{r}^0$ to $\mathbf{r}^1$ along the path $\mathbf{r} = \mathbf{r}^0 + (\mathbf{r}^1 - \mathbf{r}^0)(t - t^0)/dt$. Since the particle is within a single cell we may take each component of $\mathbf{r}^0$ and $\mathbf{r}^1$ to be in $[0, 1)$, and because the motion is linear in t the average current crossing each face can be found from the currents at $t = t^0 + dt/2$. The current deposited is then found from the particle position $\bar{\mathbf{r}} = (\mathbf{r}^0 + \mathbf{r}^1)/2$.

4 Parallel Implementation and Performance

The issues involved with parallelizing an EMPIC code are similar to those involved with any PIC code. Our algorithms are primarily designed for distributed memory parallel machines in which each processor has memory that it can access much faster than any other processor can.

The particles and the grid have to be distributed around the processors so that the computational load is approximately balanced, and the communication between processors

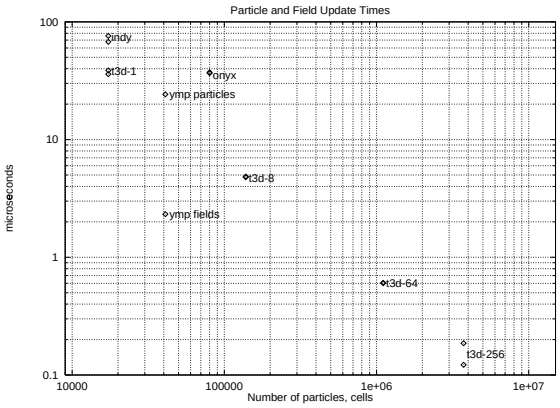

Figure 5: Times in microseconds for field and particle updates for various computers. The field update is per cell, and the particle update is per particle. The horizontal scale is the total number of cells in the simulation, and each run was done with eight particles per cell. Times on the T3D include time for communication between processors. The linear relationship between the time and the number of processors on the T3D shows the code's high parallel efficiency.

is minimized. We distribute the particles and the grid using a domain decomposition of both grid and particles in much the same way as Liewer and Decyk [8] and as used by Wang et. al [4]. The patches of the grid described above are distributed onto the processors and the particles are distributed to be with the section of grid that they will reference. All of the memory references and calculations in the inner loops are then the same as they would be on a serial computer and no interprocessor communication is needed for the particle updates. Between time steps the guard cells of the grids on each processor are exchanged and particles that have moved from the domain of one processor to another are exchanged.

In Fig. 5 we present results from running this code on a variety of machines from a workstation to a massively parallel supercomputer. The figure shows the times in microseconds for field and particle updates for several computers: Silicon Graphics Indy, Silicon Graphics Onyx, Cray YMP, and Cray T3D using 1, 8, 64 and 256 processors. The problem size was chosen to be the largest problem that would run with eight particles per cell in that system. The size of the system run is chosen to scale up with the computer in order to compare vastly different systems.

This problem scales very well on parallel architectures like the T3D, where communication to update guard cells and swap particles between processors required only about 3% of the time for a time step. From the fact that the T3D speed decreases linearly as the number of processors increases, we can see that the parallel efficiency is very high (about 97%). Here, the domain was partitioned with equal numbers of grid points in each processor domain. The parallel efficiency is this high because the particles are uniformly distributed and thus both the particle computation and the field computation are load balanced. Other schemes better suited to highly non-uniform particle distributions are discussed in the proposed work section.

For microprocessor based computers the time per cell and the time per particle are nearly

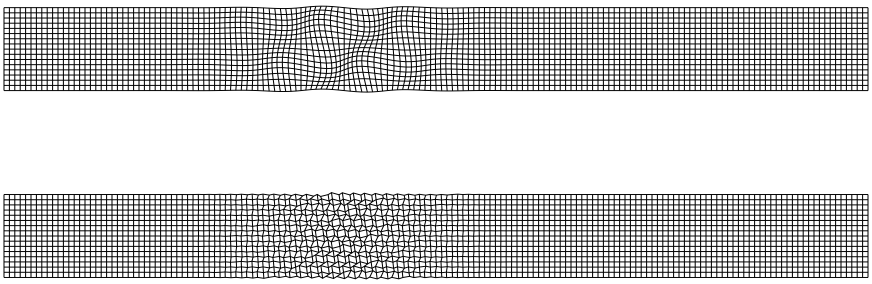

Figure 6: Smoothly and randomly distorted meshes for testing the convergence of the numerical techniques. The distortions are exaggerated here to be more easily visible.

the same. Because the field update vectorizes while the particle update does not, the particle update is much slower on the Cray YMP than the field update (Fig. 5). Because runs are typically done with eight or more particles per cell, the total time is dominated by the particle update.

4.1 Convergence Tests of the Numerical Algorithms

We describe here tests of the convergence rate of the electromagnetic discretization and the particle time step. On a uniform mesh the field and particle updates are well known to converge with second order accuracy. We consider here how they converge for non-uniform meshes.

To test the convergence of the electromagnetic update and particle motion on non-uniform grids we consider the grids shown in Figure 6. The physical size of the domain is of length $L_x = 10$, and is of unit length and periodic in y and z. For each test we consider a series of runs with increasing resolution (increasing number of grid points with system size fixed).

We consider two kinds of distortions to the uniform grid.

- Smooth distortion. Each component of the physical space coordinate $\mathbf{x}$ is moved by an amount $\delta = \alpha \sin(2\pi x) \sin(2\pi y)\sin(2\pi z)$ for $\alpha = 0.025$. For this grid, the distortion is fixed at a constant physical space length scale as the resolution increases.

- Random distortion. Each grid point is moved a random distance δ in each dimension given by $\delta = \alpha d(2r - 1)$ where d is the grid spacing, r is a random number uniformly distributed between 0 and 1, and $\alpha = 0.1$. The distortion is constant in logical space, but not physical space.

Each distortion is then localized to a region in the x direction about a position x_0 with a gaussian factor $\exp(-(x - x0)^2)$ where $x0 = 0.4$. Electromagnetic waves and test particles are injected at $x = 0$, well outside of the distorted region, allowed to propagate through it and then sampled well outside of it on the other side.

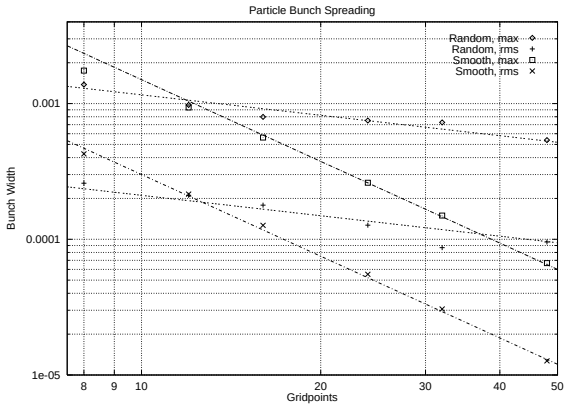

Figure 7: Convergence plots for pushing particles through distorted meshes. 10,000 particles were injected in a plane on the left hand side, and we plot the rms and maximum width of the bunch after passing through smoothly and randomly distorted meshes as a function of the number of grid point $N = L_x/\Delta x$. The data points are shown with lines of slope N^{-2} for the smooth distortions and with lines of slope $N^{-1/2}$ for the random distortions.

4.2 Particle Update Convergence Rate

We examine the convergence of the predictor corrector algorithm in Eqs. 13 for the same grids by moving particles through the grids with no electromagnetic fields. We inject 10,000 particles in a plane at x=0 and allow them to propagate through the distorted region and into the flat region on the right, where we examine the spread of the particle bunch (how much they deviate from the expected straight line orbits). The spreading (bunch width) is plotted in Fig. 7 vs. the number of grid points in the fixed length system. The curves labeled "max" plot the maximum deviation or spreading and the "rms" curves plot the root-mean-square deviation.

In the smoothly distorted grid, in which increasing the number of grid points N better resolves the distortion ($\Delta x = L_x/N$), the convergence is seen to be with the second power of the grid spacing and thus the algorithm is 2nd order (error $\propto \Delta x^2$).

For the random grid, increasing the grid points does not better resolve the distortion and here convergence is with the square root (error $\propto \sqrt{\Delta x}$). The square root convergence for the randomly distorted grid results from the particles receiving a random "kick" from each distorted cell ($\propto N$) ; the total deviation is the random walk error $\propto \sqrt{N}/N = 1/\sqrt{N} \propto \sqrt{\Delta x}$.

4.3 Electromagnetic Field Convergence Rate

To test the convergence of the electromagnetic update we inject plane polarized electromagnetic waves at $x = 0$ and allow them to pass through the distorted region centered at $x = 0.4$ in grids with resolution from $\Delta x = 1/8$ through $\Delta x = 1/800$. In Figure 8 we plot the maximum difference between the signal propagated through an undistorted and a distorted

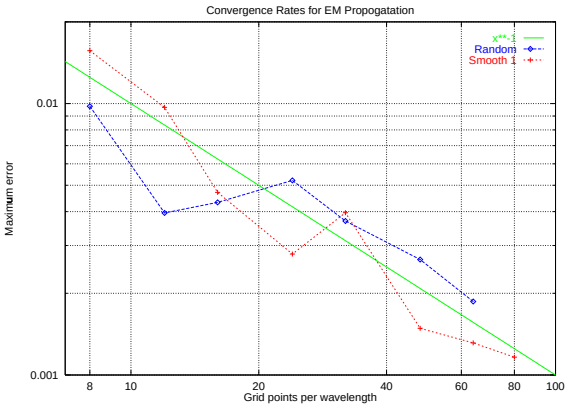

Figure 8: Convergence plots for propagating electromagnetic fields through the smooth and random grids. Plane polarized waves are injected at the left edge, passed through the distorted region, and sampled in the smooth region on the right. Both converge roughly with the first power of the mesh spacing.

mesh in the region $7 < x < 9$. The maximum is the maximum error in any of the 6 field components.

We note that for both cases the convergence is roughly with the first power of the grid spacing (error $\propto \Delta x$), confirming the first order accuracy of the spatial discretization discussed in Sec. II.A.1. This limitation of the algorithm is discussed in the proposed work section.

5 Future Directions

In this paper, we have presented the algorithm used in our three-dimensional non-orthogonal grid relativistic electromagnetic particle-in-cell code. The code has been tested by comparison with results from our Cartesian grid 3D EMPIC code for an electron two-stream instability test case. The electromagnetic field portion has been tested on both orthogonal and non-orthogonal grids using rectangular wave guides test cases with good results. A primary outstanding area of research is in improving the accuracy of the discretization of the electromagnetic update on distorted meshes. At present with first order accuracy, regions of distortion should be minimized to improve the overall accuracy of a simulation. Relaxing that constraint will greatly ease simulations of complex geometries.

6 Acknowledgements

We would like to acknowledge many useful conversations with J. U. Brackbill, LANL and Viktor Decyk, UCLA. We would also like to thank Edith Huang, JPL and Prof. D. Meiron, Caltech for their help and support. This work was supported by the AFOSR Computational

Mathematics Program under Grant #F49620-94-1-0336 to the California Institute of Technology. A portion of this work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA. The JPL/Caltech Cray YMP and T3D used in this investigation was supported by NASA. Visualization support was provided by the JPL Supercomputing Project.

7 References

1. S.D. Gedney and F. Lansing, A generalized Yee-algorithm for the analysis of microwave circuit devices, *IEEE Trans. Microwave Theory and Techniques*, (submitted for publication, 1995).
2. N. K. Madsen, Divergence preserving discrete surface integral methods for Maxwell's curl equations using non-orthogonal unstructured grids, *J. Comput. Phys.*,**119**, 34 (1995).
3. J. Villasenor and O. Buneman, Rigorous charge conservation for local electromagnetic field solves, *Comput. Phys. Comm.* **69**, 306 (1992).
4. J. Wang, P. Liewer, V. Decyk, 3D electromagnetic plasma particle simulations on a MIMD parallel computer, *Comput. Phys. Comm.* **87**, 35 (1995).
5. J. W. Eastwood, W. Arter, N. J. Brealey, R. W. Hockney, Body-fitted electromagnetic PIC software for use on parallel computers, *Comput. Phys. Comm.* **87**, 155 (1995).
6. C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation* (McGraw-Hill, New York, 1985).
7. K. S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Antennas Propagat.* **AP-14(3)**, 302 (1966).
8. P. C. Liewer and V. K. Decyk, A general concurrent algorithm for plasma particle-in-cell simulations codes, *J. Comput. Phys.* **85**, 302 (1989).