J. Plasma Physics (1999), *vol.* 61, *part* 3, *pp.* 367–389. Printed in the United Kingdom
© 1999 Cambridge University Press

Three-dimensional deformable-grid electromagnetic particle-in-cell for parallel computers

J. WANG,[1] D. KONDRASHOV,[1] P. C. LIEWER[1]
and S. R. KARMESIN[2]

[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, USA
[2]Los Alamos National Laboratory, Los Alamos, NM, USA

(Received 7 May 1998)

We describe a new parallel, non-orthogonal-grid, three-dimensional electromagnetic particle-in-cell (EMPIC) code based on a finite-volume formulation. This code uses a logically Cartesian grid of deformable hexahedral cells, a discrete surface integral (DSI) algorithm to calculate the electromagnetic field, and a hybrid logical–physical space algorithm to push particles. We investigate the numerical instability of the DSI algorithm for non-orthogonal grids, analyse the accuracy for EMPIC simulations on non-orthogonal grids, and present performance benchmarks of this code on a parallel supercomputer. While the hybrid particle push algorithm has a second-order accuracy in space, the accuracy of the DSI field solve algorithm is between first and second order for non-orthogonal grids. The parallel implementation of this code, which is almost identical to that of a Cartesian-grid EMPIC code using domain decomposition, achieved a high parallel efficiency of over 96 % for large-scale simulations.

1. Introduction

Plasma particle-in-cell (PIC) codes (Birdsall and Langdon 1985; Hockney and Eastwood 1981) have become standard research tools in plasma physics research. A PIC code models a plasma as many test particles and follows the evolution of the orbits of individual test particles in the self-consistent electromagnetic fields. Most existing electromagnetic PIC codes are based on the use of orthogonal grids (Cartesian or cylindrical). This restricts their applications to problems with simple geometries. Recently, a few studies have been published on electromagnetic (EM) and electromagnetic particle-in-cell (EMPIC) algorithms for non-orthogonal grids. Madsen (1995) has developed a discrete surface integral (DSI) method for an EM code using non-orthogonal unstructured grids. A similar algorithm has also been developed by Gedney and Lansing (1995) for an EM code. Eastwood et al. (1995) have developed the first non-orthogonal grid EMPIC code based on a body-fitted finite-element method.

In this paper, we describe a new parallel EMPIC code that uses non-orthogonal grid of hexahedral cells to handle complex geometries for large-scale simulations. The major features of this code include the following.

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

1. The computational grids are logically connected deformable hexahedral cells. Each grid point in the physical space is mapped one-to-one to a Cartesian mesh in the logical space.

2. The electromagnetic field update is based on the EM algorithms by Madsen (1995) and Gedney and Lansing (1995), specialized to the case of a structured grid of hexahedral cells. This is a discrete-volume generalization of the standard finite-difference time-domain (FDTD) algorithm (Yee 1966), and reduces identically to FDTD if the grid is orthogonal. We not only implement the original Madsen (1995) algorithm, but also present a modified version that uses a new weighting scheme.

3. The particle push is a 'hybrid' logical–physical space operation. Particle-position update, field interpolation and current deposit are performed in the Cartesian logical space, while the particle velocity is computed in physical space. This hybrid particle push is a unique feature compared with other non-orthogonal grid PIC codes.

4. The parallel code uses a domain decomposition of the grid and particles that is nearly identical to that of a Cartesian grid EMPIC code (Wang et al. 1995).

The code described here has some features similar to those of the code by Eastwood et al. (1995), such as the use of blocks of curvilinear hexahedral cells for the parallel decomposition of the computation domain and full charge conservation to avoid solving Poisson's equation for electrostatic potential correction. However, Eastwood et al. used a different approach to derive their numerical algorithm, based on a tensor formulation in general curvilinear coordinates. In Eastwood et al. (1995), the Maxwell equations and the equations of particle motion are integrated in curvilinear (or logical) space with a finite-element approximation to coordinate transformation. This is in contrast to our approach, which is based on finite volume and allows a relatively simpler formulation than the tensor formulation associated with finite elements. Moreover, we only update particle position and perform particle–grid interpolations in logical space; all other physical quantities are dealt with in physical space.

Our decisions to use deformable hexahedral cells rather than more sophisticated grids, such as tetrahedral cells or unstructured grids, and to use the hybrid logical–physical space particle push are primarily influenced by considerations of code performance on massively parallel supercomputers. The logically Cartesian connection of the hexahedral cells preserves the nearest-neighbour relationships and avoids the complications of indirect addressing needed for unstructured grids. The location in memory of quantities defined in neighbouring cells can be found trivially via indexing, which is in contrast to an unstructed grid, where the neighbours of a given cell must be found by lookups in a table or other methods requiring additional memory references. The use of the logically Cartesian grid, combined with updating particle positions in logical space, allows us to determine the locations of particles and cells almost as efficiently as in an orthogonal-grid PIC code. This is especially important for large-scale simulations, because a PIC code typically spents most of its computing time to push particles and perform particle–grid interpolations. Performing particle–grid interpolations in logical space avoids the complications of doing such tasks directly on non-orthogonal grids, and allows a direct utilization of the gather and scatter algorithms developed for Cartesian-grid

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

PIC codes. Finally, our approach allows an almost-identical parallel implementation of the code to that of the Cartesian-grid-based parallel EMPIC code (Wang et al. 1995). Interprocessor communication algorithms developed for Cartesian-grid-based parallel PIC codes may be applied with little modifications. Moreover, in a parallel code using a domain decomposition, code efficiency is influenced strongly by the time spent moving particles to the proper processor as they move out of the current processor's domain. For our logically Cartesian grid, the proper processor for a particle is determined trivially from its logical-space coordinates and the algorithms for exchanging particles among processors are nearly identical to and as efficient as those of the Cartesian-grid parallel EMPIC code (Wang et al. 1995).

In addition to the numerical algorithms for fields and particles, this paper also presents an analysis of the accuracy of the particle push on distorted grids, which has not been previously reported, and a stability and accuracy analysis for the finite-volume electromagnetic method, which extends the previous analysis by Madsen (1995) and investigates the weak numerical instability caused by grid distortion. Specifically, this instability is studied for three different weighting schemes for the electromagnetic field update and a procedure for determining which weighting scheme to use is suggested. This paper also presents performance benchmarks on parallel computers, which are compared with those of the Cartesian-grid parallel EMPIC code (Wang et al. 1995).

The numerical algorithm is described in Sec. 2. Numerical experiments of this code are discussed in Sec. 3, which includes accuracy of particle push, stability and accuracy of electromagnetic field update, and energy conservation of the entire EMPIC code. Parallel implementation and parallel benchmarking of the code is discussed in Sec. 4. Section 5 contains a summary and conclusions.

2. Algorithm

2.1. Governing equations

The governing equations in an electromagnetic PIC code are Newton's second law for individual particle trajectories (Birdsall and Langdon 1985; Hockney and Eastwood 1981)

$$\frac{d\gamma m\mathbf{v}}{dt} = \mathbf{F} = q\left(\mathbf{E} + \mathbf{v} \times \frac{\mathbf{B}}{c}\right), \tag{1}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v},$$

where $\gamma m = m/(1 - v^2/c^2)^{1/2}$ is the relativistic mass of the plasma particle, and the Maxwell equations for the macroscopic electromagnetic field:

$$\frac{\partial \mathbf{B}}{\partial t} = -c\boldsymbol{\nabla} \times \mathbf{E}, \tag{2a}$$

$$\frac{\partial \mathbf{E}}{\partial t} = c\boldsymbol{\nabla} \times \mathbf{B} + \mathbf{J}, \tag{2b}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0, \tag{2c}$$

$$\boldsymbol{\nabla} \cdot \mathbf{E} = \rho. \tag{2d}$$

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

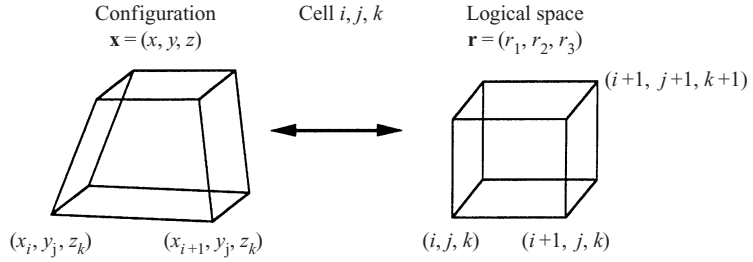


Figure 1. Transformation between logical space and physical space.

The charge density ρ and the current density $\mathbf{J}$ are derived from the motion of plasma particles, and satisfy the charge-continuity equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{J}. \tag{3}$$

Note that if this charge-conservation condition can be satisfied rigorously numerically, one can update the electromagnetic field using only the two curl Maxwell equations.

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

$$\partial_t \int_s \mathbf{B} \cdot d\mathbf{s} = -c \oint_l \mathbf{E} \cdot d\mathbf{l}, \tag{4a}$$

$$\partial_t \int_s \mathbf{E} \cdot d\mathbf{s} = c \oint_l \mathbf{B} \cdot d\mathbf{l} + \int_s \mathbf{J} \cdot d\mathbf{s}, \tag{4b}$$

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

$$\oint_s \mathbf{E} \cdot d\mathbf{s} = \int_v \rho \, dv, \tag{5a}$$

$$\oint_s \mathbf{B} \cdot d\mathbf{s} = 0, \tag{5b}$$

$$\partial_t \int_v \rho \, dv = \int_s \mathbf{J} \cdot d\mathbf{s}. \tag{5c}$$

2.2. Computational grid

We define a physical space Ω_P consisting of a structured grid of hexahedral cells that can be deformed in shape to body-fit complex geometries, and a logical space Ω_L consisting of a Cartesian mesh. Depending on the geometry of the physical domain and/or domain decomposition on parallel computers, the physical domain can be broken down into a set of non-overlapping patches, each with its own Cartesian coordinates in the logical space. As illustrated in Fig. 1, each cell in Ω_P is mapped one-to-one to a unit cube in Ω_L. The vertices of the cells in Ω_P are at the locations where the logical coordinates are all integers. The neighbours can be found by simply incrementing indices, and hence the complications of indirect addressing of unstructured grids are avoided.

Let $\mathbf{r} = (r, s, t)$ denote the logical-space coordinates and $\mathbf{x} = (x, y, z)$ the physical-space coordinates. A simple trilinear interpolation is used to map the logical coor-

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

dinates to the physical coordinates. For instance, the mapping to the x coordinate inside cell (i, j, k) is

$$x = x_0 + x_1 r + x_2 s + x_3 rs + x_4 t + x_5 rt + x_6 st + x_7 rst, \tag{6}$$

where

$$\begin{aligned}
x_0 &= x_{i,j,k},\\
x_1 &= x_{i+1,j,k} - x_{i,j,k},\\
x_2 &= x_{i,j+1,k} - x_{i,j,k},\\
x_4 &= x_{i,j,k+1} - x_{i,j,k},\\
x_3 &= x_{i+1,j+1,k} + x_{i,j,k} - x_{i,j+1,k} - x_{i+1,j,k},\\
x_5 &= x_{i+1,j,k+1} + x_{i,j,k} - x_{i,j,k+1} - x_{i+1,j,k},\\
x_6 &= x_{i,j+1,k+1} + x_{i,j,k} - x_{i,j,k+1} - x_{i,j+1,k},\\
x_7 &= -x_{i,j,k} + x_{i+1,j,k} + x_{i,j+1,k} + x_{i,j,k+1}\\
&\quad -x_{i+1,j+1,k} - x_{i+1,j,k+1} - x_{i,j+1,k+1}\\
&\quad +x_{i+1,j+1,k+1}.
\end{aligned}$$

The mapping to the y and z coordinates is similar.

2.3. Particle push

The particle push portion of the EMPIC code includes interpolating fields to particle positions (gather), updating particle positions and velocities (particle move), and depositing particle currents (scatter). In our code, the particle position is kept in logical space while the velocity is kept in physical space. 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. We choose to keep the particle location in logical space because this allows us to determine both a particle's cell and processor with the same efficiency of an orthogonal-grid parallel PIC code. This is in contrast to updating particle position in real space, which then requires a fairly complex scheme to determine a particle's new cell (Westermann 1992). Setting the particles' fundamental shape and position in logical space also simplifies both the force interpolation and the current deposit, because the interpolation weights are simple linear functions of each logical coordinate.

2.3.1. Gather and scatter. The field interpolation (gather) and current deposit (scatter) are done in logical space, which is Cartesian. Therefore the numerical schemes for field interpolation and current deposit are exactly the same as that in a flat-grid electromagnetic PIC code (see e.g. Wang et al. 1995). In particular, the current deposit is based on the charge conservation scheme discussed in Villasenor and Buneman (1992), which satisfies the discretized charge-conservation condition

$$\begin{aligned}
d_t \rho\, dV = {} & \mathbf{J} \cdot d\mathbf{s}_1 - \mathbf{J} \cdot d\mathbf{s}_{1,i+1} + \mathbf{J} \cdot d\mathbf{s}_2\\
& - \mathbf{J} \cdot d\mathbf{s}_{2,j+1} + \mathbf{J} \cdot d\mathbf{s}_3 - \mathbf{J} \cdot d\mathbf{s}_{3,k+1}
\end{aligned} \tag{7}$$

rigorously. Here $\rho\, dV$ is the total charge in a single cell of the **E** grid and $\mathbf{J} \cdot d\mathbf{s}$ is the current crossing a face of the **E** grid. The current deposit is done by calculating for each particle within a time step how much charge crosses each face of the **E** grid using the cells in logical space. Since current deposit is performed in logical space,

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

exchanging currents in the guard cells between neighbouring patches or neighbouring subdomains of parallel processors is identical to that in the orthogonal-grid parallel PIC code (Wang et al. 1995).

2.3.2. Particle move. The trajectory of each particle is integrated using a time-centring leapfrog scheme for (1). From the velocity in physical space $\mathbf{v}$, the logical space position $\mathbf{r}$ is readily obtained from

$$\frac{d\mathbf{r}}{dt} = \mathbf{R}(\mathbf{r}) \cdot \mathbf{v}, \tag{8}$$

where

$$\mathbf{R}(\mathbf{r}) = \left[\frac{\partial(x, y, z)}{\partial(r, s, t)}\right]^{-1}$$

is the rotation matrix. (The physical location of the particle can be easily calculated from $\mathbf{r}$ using the geometric information for each cell.) The rotation matrix $\mathbf{R}$ is calculated at each vertex once at the start of the computation. During the time loop, it is interpolated to the particle position in the particle update using linear interpolation in logical space.

The numerical scheme for particle move on a non-orthogonal grid, which requires a modification to the simple leapfrog update of $\mathbf{x}$ and $\mathbf{v}$ for an orthogonal grid, is as follows:

1. Identify particle location at step n in the logical space and interpolate the field value to the particle position: $\mathbf{E}(\mathbf{r}^n)$, $\mathbf{B}(\mathbf{r}^n)$
2. Update $\mathbf{v}$ in physical space: $\mathbf{u} = \gamma\mathbf{v}$,

$$\mathbf{u}^{n+1/2} - \mathbf{u}^{n-1/2} = dt\left(\frac{q}{m}\mathbf{E}^n + \frac{\mathbf{u}^{n+1/2} + \mathbf{u}^{n-1/2}}{2\gamma^n} \times \frac{q}{m}\gamma^n\frac{\mathbf{B}^n}{c}\right). \tag{9}$$

3. Advance $\mathbf{r}$ in the logical space with a predictor–corrector scheme:
   3.1. Calculate the rotation matrix at $\mathbf{r}^n$: $\mathbf{R}(\mathbf{r}^n)$.
   3.2. Predict $\mathbf{r}^{n+1/2}$:

$$\mathbf{r}^{n+1/2} = \mathbf{r}^n + \mathbf{R}(\mathbf{r}^n) \cdot \mathbf{v}^{n+1/2}\, dt/2. \tag{10}$$

   3.3. Calculate the rotation matrix at $\mathbf{r}^{n+1/2}$: $\mathbf{R}(\mathbf{r}^{n+1/2})$.
   3.4. Advance $\mathbf{r}$ a full time step:

$$\mathbf{r}^{n+1} = \mathbf{r}^n + \mathbf{R}(\mathbf{r}^{n+1/2}) \cdot \mathbf{v}^{n+1/2}\, dt. \tag{11}$$

4. Trade those particles that have moved to neighbouring patches or neighbouring subdomains of parallel processors.

2.4. Electromagnetic field solve

The electromagnetic field is updated using an explicit DSI (discrete surface integral) solution of the Maxwell equations, which is modified from the Gedney and Lansing (1995) algorithm and the Madsen (1995) algorithm for structured hexahedral cells. This algorithm uses a staggered grid system, as illustrated in Fig. 2. For each cell (primary cell), we define a dual cell with vertices all at half integers in the logical space. We shall call the primary cell the **B** grid and the dual cell the **E** grid. The fundamental variables used to update the electromagnetic field field in the code are

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

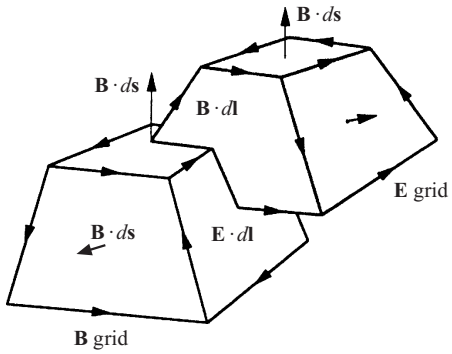


Figure 2. Primary and dual grid layout and location of electromagnetic field variables on the grids.

$\mathbf{B} \cdot d\mathbf{s}$ (on each face of the **B**-grid cells), $\mathbf{E} \cdot d\mathbf{s}$ (on each face of the **E** cells), $\mathbf{B} \cdot d\mathbf{l}$ (on the edges of each **E** cell and passing through the faces of the **B** grids), and $\mathbf{E} \cdot d\mathbf{l}$ (on edges of each **B** cell and pass through faces of the **E** grids). The current $\mathbf{J} \cdot d\mathbf{s}$ is co-located with $\mathbf{E} \cdot d\mathbf{s}$. When the grid is Cartesian, the edges and the face normals are colinear; and the grid system reduces to the familiar Yee lattice for Cartesian FDTD codes (Yee 1966).

We label each grid quantity with the logical coordinates of its centre, for example, a cell is labelled by the coordinates of its centre and a $\mathbf{B} \cdot d\mathbf{s}$ by the coordinates of the face centre. The vertices of the **B** grid and cells of the **E** grid are integer triplets, and all other quantities have one, two or three 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. To map these quantities to array locations, we simply drop any half-integers in the coordinates.

The electromagnetic field is updated from the discretized curl Maxwell equations using a leapfrog scheme:

$$(\mathbf{E} \cdot d\mathbf{s})^{n+1} = (\mathbf{E} \cdot d\mathbf{s})^{n} + dt \left(c \sum_{\text{edges}} \mathbf{B} \cdot d\mathbf{l} + \mathbf{J} \cdot d\mathbf{s} \right)^{n+1/2}, \tag{12a}$$

$$(\mathbf{B} \cdot d\mathbf{s})^{n+3/2} = (\mathbf{B} \cdot d\mathbf{s})^{n+1/2} + dt \left(c \sum_{\text{edges}} \mathbf{E} \cdot d\mathbf{l} \right)^{n+1}. \tag{12b}$$

By using the staggered grid, these equations will hold automatically for the discretized field divergence conditions (5). Therefore, if the particle push part of the code maintains the charge-conservation condition (3) rigorously, one can update the electromagnetic field only using (12) as long as the initial condition satisfies (5).

To close (12), we need to specify how the edge quantities, $\mathbf{E} \cdot d\mathbf{l}$ and $\mathbf{B} \cdot d\mathbf{l}$, are determined from $\mathbf{E} \cdot d\mathbf{s}$ and the $\mathbf{B} \cdot d\mathbf{s}$ respectively. For a uniform orthogonal grid, $d\mathbf{s}$ and $d\mathbf{l}$ are parallel, and the transformation from face quantities to edge quantities is trivial. For instance, we can calculate $\mathbf{B} \cdot d\mathbf{l} = \mathbf{B} \cdot d\mathbf{s} |d\mathbf{l}|/|d\mathbf{s}|$. Therefore there is no real distinction between face and edge quantities, and the algorithm reduces to the standard second-order FDTD algorithm. However, for non-orthogonal coordinates,

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

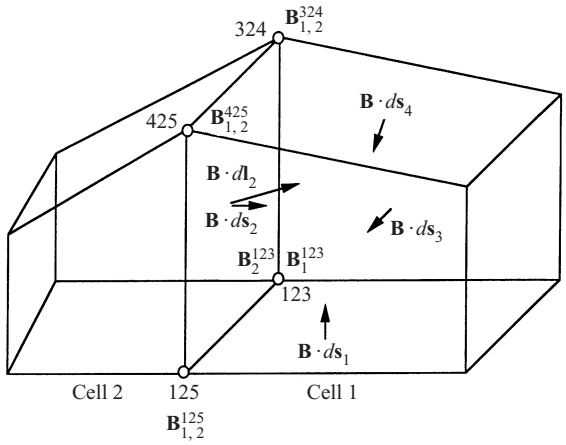


Figure 3. Illustration of face-to-edge transformation.

$d\mathbf{l}$ is not parallel to $d\mathbf{s}$, and converting face to edge quantities is more involved. The calculation of $\mathbf{B} \cdot d\mathbf{l}$ for a particular cell face involves two steps. Here we illustrate this calculation by considering face 2 in Fig. 3. The calculations of $\mathbf{B} \cdot d\mathbf{l}$ for other cell faces are performed in the same manner. We denote the two neighbouring cells separated by face 2 as cell 1 and cell 2. In each cell, there are four faces that are adjacent to face 2. We shall number them in each cell as faces 1, 3, 4 and 5.

The first step is to find the vertex values of $\mathbf{B}$ for the four vertices of face 2. A vertex value of $\mathbf{B}$ is obtained from the three cell faces sharing that vertex by solving a set of three coupled equations. For instance, the value of $\mathbf{B}$ at the vertex shared by faces 1, 2 and 3 of cell 1 is found by solving

$$\mathbf{B}^{123} \cdot \mathbf{n}_1 = \mathbf{B} \cdot d\mathbf{s}_1, \tag{13a}$$

$$\mathbf{B}^{123} \cdot \mathbf{n}_2 = \mathbf{B} \cdot d\mathbf{s}_2, \tag{13b}$$

$$\mathbf{B}^{123} \cdot \mathbf{n}_3 = \mathbf{B} \cdot d\mathbf{s}_3, \tag{13c}$$

where $\mathbf{n}_i$ is the unit normal to face i. We can perform the same calculation using faces 1, 2 and 3 of cell 2. Hence there are two values of $\mathbf{B}$ at each vertex associated with the two sides of the face. We denote the two vertex values of $\mathbf{B}$ calculated by considering face values at the two neighbouring cells separated by face 2 as $\mathbf{B}^{123}_1$ and $\mathbf{B}^{123}_2$. The vertex values at the other three vertices $\mathbf{B}^{324}_l$, $\mathbf{B}^{125}_l$ and $\mathbf{B}^{425}_l$ are calculated in the same manner, where subscript l refers to the two neighbouring cells.

The second step requires a weighted average of the eight vertex values dotted with the dual-edge vector to obtain $\mathbf{B} \cdot d\mathbf{l}$:

$$\mathbf{B} \cdot d\mathbf{l}_2 = \sum_{l=1}^{2} (\mathbf{B}^{123}_l w^{123}_l + \mathbf{B}^{324}_l w^{324}_l + \mathbf{B}^{125}_l w^{125}_l + \mathbf{B}^{425}_l w^{425}_l) \cdot d\mathbf{l}_2. \tag{14}$$

We have investigated three different weighting methods that may be used in (14):

Method A: Equal weights. Here

$$w^{i2k}_m = 0.125. \tag{15}$$

This method is identical to the simple vector sum average in Madsen (1995).

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

Method B: Two-sided volume weights. Here the weights are calculated as the volumes associated with each corner involved. The volume is calculated as a cross-product of edge vectors. Afterwards, the weights are averaged:

$$w_l^{i2k} = \frac{V_l^{i2k}}{\Sigma_{lik} V_l^{i2k}}. \tag{16}$$

This method is analogous to the full volume-weighted average by Madsen, but with weights defined slightly differently in our case.

Method C: One-sided volume weights. Here the vector $d\mathbf{l}_2$ is split into two vectors:

$$d\mathbf{l}_2 = d\mathbf{l}_{21} + d\mathbf{l}_{22},$$

where $d\mathbf{l}_1$ and $d\mathbf{l}_2$ are the vectors from the centres of cell 1 and cell 2 to the centre of face 2 respectively. We then rearrange (14) as

$$\mathbf{B} \cdot d\mathbf{l}_2 = \sum_{l=1}^{2} (\mathbf{B}_l^{123} w_l^{123} + \mathbf{B}_l^{324} w_l^{324} + \mathbf{B}_l^{125} w_l^{125} + \mathbf{B}_l^{425} w_l^{425}) \cdot d\mathbf{l}_{2l}. \tag{17}$$

The weights in (17) are one-sided volume averages:

$$w_m^{i2k} = \frac{V_l^{i2k}}{\Sigma_{ik} V_l^{i2k}}, \qquad l = 1, 2. \tag{18}$$

This method has not been investigated previously.

We have implemented all three of the weighting schemes in our code. These schemes will be analysed in Secs 3.2 and 3.3.

Finally, once all the $\mathbf{B} \cdot d\mathbf{s}$ and $\mathbf{E} \cdot d\mathbf{s}$ have been updated, we calculate $\mathbf{B}$ and $\mathbf{E}$ at the vertices of the primary grid, $\mathbf{B}_{\text{vert}}$ and $\mathbf{E}_{\text{vert}}$. $\mathbf{B}_{\text{vert}}$ and $\mathbf{E}_{\text{vert}}$ are the field vectors that are used to push particles. Each vertex of a primary grid is an intersection of twelve adjacent $\mathbf{B}$ primary faces (where $\mathbf{B} \cdot d\mathbf{s}$ is known) and is enclosed in a box by six dual $\mathbf{E}$ faces (where $\mathbf{E} \cdot d\mathbf{s}$ is known). Thus, for each primary-grid vertex, the calculation of $\mathbf{B}_{\text{vert}}$ involves twelve $\mathbf{B} \cdot d\mathbf{s}$ and that of $\mathbf{E}_{\text{vert}}$ involves six $\mathbf{E} \cdot d\mathbf{s}$. To compute $\mathbf{B}_{\text{vert}}$ at a vertex, we first compute the eight $\mathbf{B}$-vector values associated with the eight cells sharing this vertex. This is done by considering $\mathbf{B} \cdot d\mathbf{s}$ on three adjacent faces at a time and solving a system of three equations similar to (14), as discussed above. The eight $\mathbf{B}$-vector values are then averaged to obtain $\mathbf{B}_{\text{vert}}$. $\mathbf{E}_{\text{vert}}$ is obtained similarly.

3. Numerical experiments and algorithm analysis

3.1. Accuracy of the particle push algorithm

We first analyse the accuracy of the particle push part of the code for non-orthogonal grids. In an orthogonal-grid PIC code, the leapfrog scheme for particle update has second-order accuracy in time and space. In the present non-orthogonal-grid PIC code, the particle move algorithm involves transformation between the physical space and the logical space through a rotation matrix. Hence we first derive the numerical error introduced by the rotation matrix.

Let us characterize a smoothly distorted grid by α, representing the grid distortion magnitude, and λ, representing the wavelength of the distortion in the system. For simplicity, we consider the two-dimensional domain shown in Fig. 4. In this

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

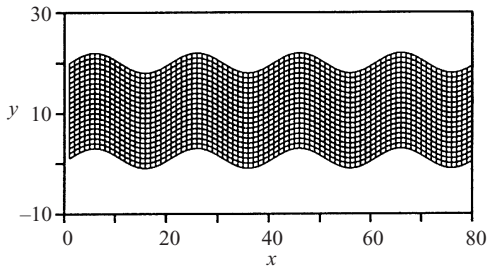


Figure 4. A distorted grid used in particle push test: $x(i,j) = ih$ and $y(i,j) = jh + \alpha \sin(ih/\lambda)$.

grid, the x coordinate is discretized with grid spacing $h = L_x/N_x$, where L_x is the domain length along the x direction and N_x the number of grid points. However, the y coordinate is deformed from the Cartesian location by $\delta y = \alpha \sin(x/\lambda)$. Hence the discrete grids are

$$x(i,j) = ih, \tag{19a}$$

$$y(i,j) = jh + \alpha \sin\left(i\frac{h}{\lambda}\right), \tag{19b}$$

where $\lambda = L_x/2\pi m$ ($m = 1,2,\ldots$). Note that h/λ represents the grid resolution for a given grid distortion.

From (6), the transformation between $\mathbf{\Omega}_p$ and $\mathbf{\Omega}_L$ within the (i,j) cell is

$$x = x(i,j) + rh, \tag{20a}$$

$$y = y(i,j) + \alpha\left\{\sin\left[(i+1)\frac{h}{\lambda}\right] - \sin\left(i\frac{h}{\lambda}\right)\right\} r + sh, \tag{20b}$$

where $r, s \in [0,1]$. Hence the rotation matrix is

$$\frac{dx}{dt} = h\frac{dr}{dt}, \tag{21a}$$

$$\begin{aligned}\frac{dy}{dt} &= \alpha\left\{\sin\left[(i+1)\frac{h}{\lambda}\right] - \sin\left(i\frac{h}{\lambda}\right)\right\}\frac{dr}{dt} + h\frac{ds}{dt} \\ &= \alpha\left[\frac{h}{\lambda}\cos\left(i\frac{h}{\lambda}\right) - \left(\frac{h}{\lambda}\right)^2\frac{1}{2}\sin\left(i\frac{h}{\lambda}\right) + o\left(\left(\frac{h}{\lambda}\right)^2\right)\right]\frac{dr}{dt} + h\frac{ds}{dt}.\end{aligned} \tag{21b}$$

In the limit of a continuous grid, $h \to 0$, the grids are given by

$$x = \zeta h,$$

$$y = \alpha \sin\left(\zeta\frac{h}{\lambda}\right) + \eta h,$$

where $\zeta, \eta \in (-\infty, \infty)$. Hence the rotation matrix is simply

$$\frac{dx}{dt} = h\frac{d\zeta}{dt}, \tag{22a}$$

$$\begin{aligned}\frac{dy}{dt} &= \alpha\frac{h}{\lambda}\cos\left(\zeta\frac{h}{\lambda}\right)\frac{d\zeta}{dt} + h\frac{d\eta}{dt} \\ &= \alpha\left[\frac{h}{\lambda}\cos\left(i\frac{h}{\lambda}\right) - \lambda^{-2}\delta\sin\left(i\frac{h}{\lambda}\right) + o(\lambda^{-2}h\delta)\right]\frac{d\zeta}{dt} + h\frac{d\eta}{dt},\end{aligned} \tag{22b}$$

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

where $\zeta = ih + \delta$. The difference between (22) and (21) is the numerical error introduced by the rotation matrix for the discrete grid of (19):

$$\text{Error}(\mathbf{R}) = \sin\left(i\frac{h}{\lambda}\right)\alpha\lambda^{-2}h\left(\frac{h}{2} - \delta\right) = O\left(\alpha\left(\frac{h}{\lambda}\right)^2\right). \tag{23}$$

This scaling of $O(\alpha h^2/\lambda^2)$ for error may be derived similarly for other simple distorted grids.

For a numerical test, we consider a particle moving in the absence of a force field, with an initial velocity v_{x0} along the x direction in the domain shown in Fig. 4. Figure 5 shows the error in the particle's trajectory as a function of its x position for different grid distortions α, grid resolutions mh and time step $v_{x0}\,dt/h$. Figure 6 shows the maximum spatial error as a function of α for three different grid resolutions: $h/\lambda = 0.0125(2\pi)$, $0.025(2\pi)$ and $0.05(2\pi)$. In agreement with the above derivation, we find that the error is linearly proportional to grid distortion, $\delta/h = O(\alpha)$, and second order in the grid resolution $\delta/h = O(h^2/\lambda^2)$. Not surprisingly, the spatial error δ/h in the trajectory is independent of the time step $v_{x0}\,dt/h$. We have done numerical testing on other smoothly distorted grids, and observed the same accuracy results.

Hence we conclude that the rotation matrix will introduce an additional numerical error to the leapfrog scheme that scales as $O(\alpha h^2/\lambda^2)$. The particle move scheme for non-orthogonal grid described in Sec. 2.4 is still second-order accurate in time and space, and is linearly proportional to the grid distortion magnitude:

$$\text{Error}_{\text{part}} = O(dt^2) + O(h^2) + O\left(\alpha\left(\frac{h}{\lambda}\right)^2\right). \tag{24}$$

3.2. Stability of the EM algorithm

In this and the next subsection, we analyse the electromagnetic field solve portion of the code alone. As discussed in Sec. 2, the field solve reduces to the standard FDTD scheme and has a second-order convergence rate when the grid is orthogonal. As we are unable to derive the accuracy of the field solve algorithm analytically for non-orthogonal grids, the stability and accuracy of the field solve will be studied by a series of numerical tests. This analysis includes three weighting schemes (i.e. Methods A, B and C) discussed in Sec. 2.3. It has been reported that the DSI algorithm is weakly unstable for certain non-orthogonal grids (Brandon and Rambo 1995; Langdon 1995). Hence we first examine the numerical stability of the field solve in this section.

The numerical test that we used was the propagation of a planar electromagnetic wave in a vacuum, periodic box with distorted grids. We first consider a rectangular box of size $L_x \times L_y \times L_z$ with $N_x \times N_y \times N_z$ cells. The grids are distorted in the x and y direction:

$$x(i,j,k) = ih + \alpha \sin\left(2\pi\frac{ih}{L_x}\right)\sin\left(2\pi\frac{jh}{L_y}\right), \tag{25a}$$

$$y(i,j,k) = jh + \alpha \sin\left(2\pi\frac{ih}{L_x}\right)\sin\left(2\pi\frac{jh}{L_y}\right), \tag{25b}$$

$$z(i,j,k) = kh, \tag{25c}$$

where $h = L_x/N_x$. A (x,y) cutting plane of such a domain is shown in Fig. 7 for two different grid distortion magnitude α. The domain is initially loaded with a plane

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

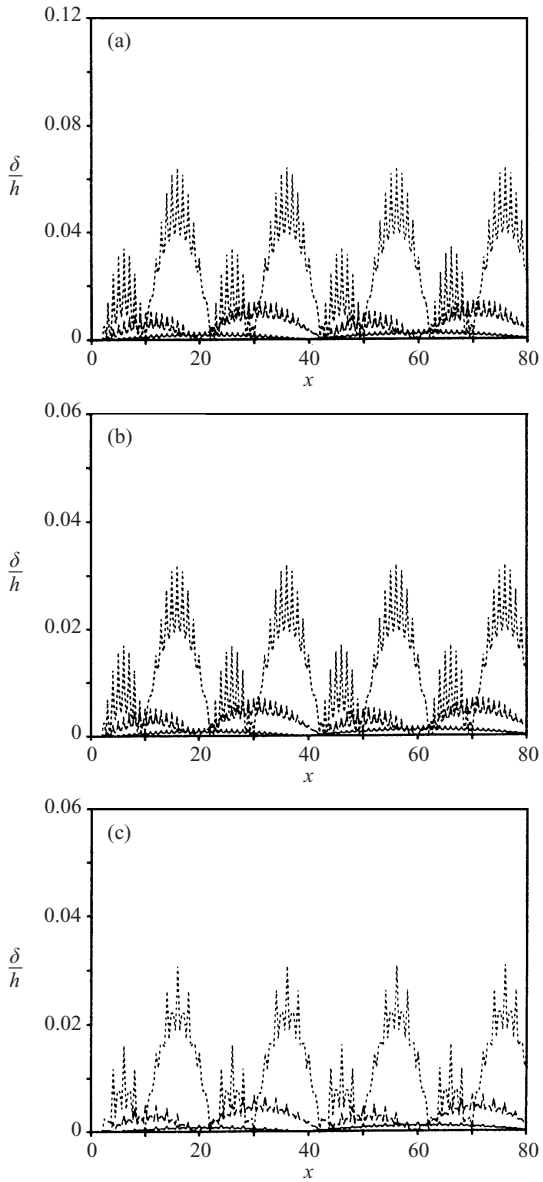


Figure 5. Numerical error in particle orbit. (a) $\alpha = 2$, $d\hat{t} = dt/(h/v_{x0}) = 0.1$; (b) $\alpha = 1$, $dt/(h/v_{x0}) = 0.1$; (c) $\alpha = 1$, $dt/(h/v_{x0}) = 0.4$. Grid resolutions $h/\lambda = 0.0125(2\pi)$ (solid line), $h/\lambda = 0.025(2\pi)$ (dashed line) and $h/\lambda = 0.05(2\pi)$ (dotted line).

EM wave propagating along the z direction with wavenumber $k_z = 2\pi/L_z$. The non-zero components of the EM field are E_x and B_y. We take the domain lengths to be $L_x = L_y = 10$ and $L_z = 20$ and the numbers of grid points to be $N_x = 16$, $N_y = 16$ and $N_z = 32$. We take the time step to be significantly below the Courant condition for the smallest grid cells. We take the speed of light $c = 1$. Hence the transit time for the wave through the box is $L_z/c = 20h/c$.

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

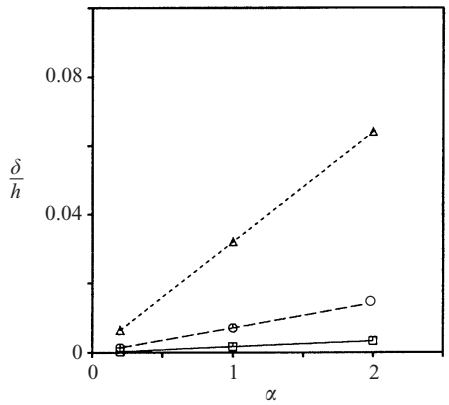


Figure 6. Maximum error in particle orbit. Grid resolutions $h/\lambda = 0.0125(2\pi)$ (solid line), $h/\lambda = 0.025(2\pi)$ (dashed line) and $h/\lambda = 0.05(2\pi)$ (dotted ine).

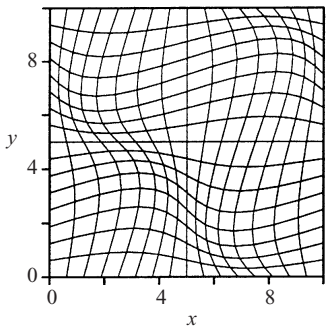


Figure 7. A distorted grid used in field solve test and PIC code test.

We use the total electromagnetic energy E_{fld}, and the z component of $\mathbf{E} \cdot d\mathbf{s}$, $\tilde{E}_z = (\mathbf{E} \cdot d\mathbf{s})_z$, as a measure of the numerical error. For the analytical solution, the total energy is a constant, $E_{\text{fld}} = 1000$, and the z component of the $\mathbf{E}$ field is always zero. Time history of the total energy E_{fld} and the maximum value of $(\mathbf{E} \cdot d\mathbf{s})_z$ over the computational domain are shown in Fig. 8 for a distorted grid with $\alpha = 0.8$. We find that the electromagnetic algorithm is numerically unstable for all three weighting schemes, even though the Courant condition is well satisfied in all grid cells. For the distorted grid used here, Method A (equal weight) has the smallest growth rate, while Method C (one-sided volume weight) has the largest growth rate (Fig. 8a). Note also from Fig. 8(a) that the run using Method A takes about 750 wave transit times to become unstable, while the run using Method C takes only about 50 wave transit times. From Fig. 8(b), one can observe that the numerical error as measured by $(\mathbf{E} \cdot d\mathbf{s})_z$ is directly related to the numerical stability. Once the numerical error becomes comparable to the non-zero components of the exact solution, the instability results in an exponential growth of the total energy. In Fig. 9, we plot the time history of the wave energy for several different grid distortions α, using Methods A and C. From Fig. 9, it can be seen that the instability growth rate increases with the grid distortion α. For comparison, we also considered a non-orthogonal uniformly skewed grid, shown in Fig. 10(a). The time history of the non-physical field component $(\mathbf{E} \cdot d\mathbf{s})_z$ is shown in Fig. 10(b). In contrast to the

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

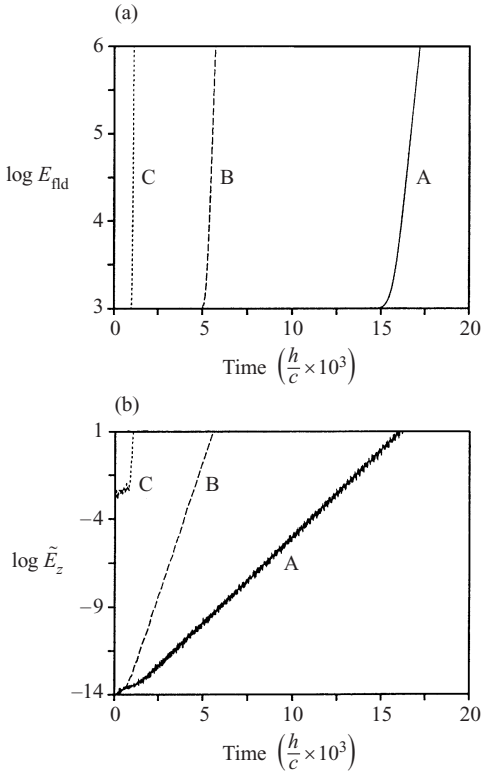


Figure 8. Numerical stability test of the EM algorithm. (a) Time history of the total energy. (b) Time history of the non-physical component $(\mathbf{E}\cdot d\mathbf{s})_z$. Simulations are performed using Method A (solid line), Method B (dashed line) and Method C (dotted line). Grid distortion $\alpha = 0.8$.

results for the grid shown in Fig. 7, we find that the solution for this grid is always numerically stable.

Madsen (1995) did not observe this weak numerical instability associated with grid distortion, and concluded that the DSI algorithm is stable as long as the Courant condition is satisfied. It is not clear whether this was due to his choice of grid or to not running the computation long enough. Based on the Courant condition, Madsen finds that the simple vector sum weighting (Method A) allows the smallest time step (Courant number) for the solution to be numerically stable, and thus is inferior to other weighting schemes. We also observe that the solution become unstable quickly if the critical Courant number is exceeded. However, the weak numerical instability discussed here occurs even though the Courant number is well below the critical value. This distorted-grid instability grows much more slowly than that from the violation of the Courant condition, and its growth rate does not depend upon the value of the time step.

Brandon and Rambo (1995) have studied the dispersion characteristics of the DSI method using a 'Chervon' grid, and have shown that the DSI algorithm supports slowly growing electromagnetic oscillations. These oscillations have complex frequencies with non-zero growth rate for a vanishing time step. Langdon (1995) has considered the two-dimensional DSI algorithm, and has suggested that the

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

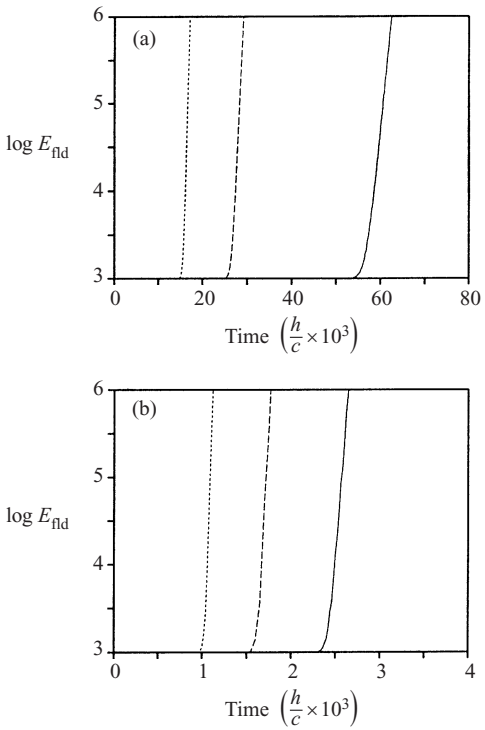


Figure 9. Numerical stability test of the EM algorithm: time history of the total energy. (a) Simulations using Method A. (b) Simulations using Method C. Grid distortion $\alpha = 0.4$ (solid line), $\alpha = 0.6$ (dashed line), and $\alpha = 0.8$ (dotted line).

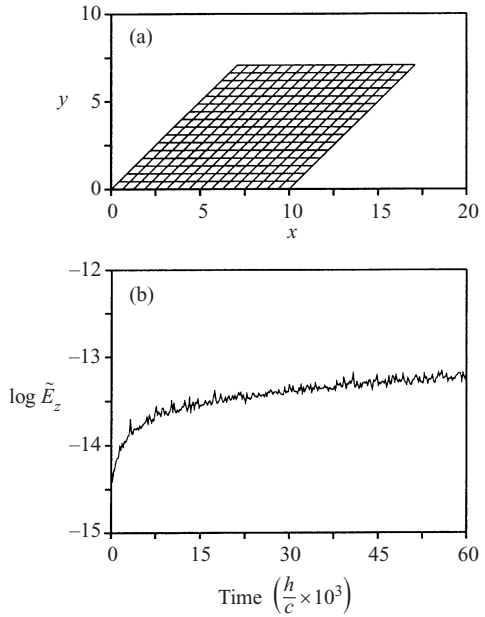


Figure 10. Numerical stability test of the EM algorithm. for a uniformly skewed grid. (a) The skewed grid. (b) Time history of the non-physical component $(\mathbf{E}\cdot d\mathbf{s})_z$.

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

condition for numerical stability is controlled by the properties of the face-to-edge transformation matrix, which in turn depends on the weighting scheme and the grid geometry. Here we present a framework for analysis of the three-dimensional DSI algorithm.

Let E_{DF}, B_{PF}, B_{DE} and E_{PE} denote vectors of $\mathbf{E}\cdot d\mathbf{s}$, $\mathbf{B}\cdot d\mathbf{s}$, $\mathbf{B}\cdot d\mathbf{l}$ and $\mathbf{E}\cdot d\mathbf{l}$ respectively. The spatially discretized Maxwell equations in integral form can be written in the vector form:

$$\partial_t E_{DF} = S_E \cdot B_{DE}, \tag{26a}$$

$$\partial_t B_{DF} = S_B \cdot E_{PE}, \tag{26b}$$

where the matrices S_E and S_B are related by $S_E = -S_B^{\mathrm{T}}$. The face-to-edge transformation can be specified as

$$B_{DE} = T_B \cdot B_{PF},$$

$$E_{PE} = T_E \cdot E_{DF}.$$

Taking the time derivative of (26), we have

$$\partial_{tt} E_{DF} = -M_E \cdot E_{DF}, \tag{27a}$$

$$\partial_{tt} B_{PF} = -M_B \cdot B_{PF}, \tag{27b}$$

where

$$M_E = S_E T_B S_E^{\mathrm{T}} T_E, \qquad M_B = S_E^{\mathrm{T}} T_E S_E T_B.$$

If the matrices M_B and M_E are symmetric and positive-definite then all of their eigenvalues are positive. The largest eigenvalue will impose a restriction on the maximal time step allowed: the Courant limit. The computations will be numerically stable under this limit. The matrices M_B and M_E will be symmetric if

$$S_E T_B = T_E^{\mathrm{T}} S_E, \tag{28}$$

and they will be positive-definite if the boundary conditions are set properly.

For the uniform Cartesian mesh, the face-to-edge transformation is a diagonal matrix Λ: $T_B = T_E = \Lambda$. Obviously, (28) is satisfied in this case, and the solution is always stable, which is known from the properties of the Yee algorithm. For the non-orthogonal uniformly skewed grid shown in Fig. 9(a), the face-to-edge transformation is symmetric:

$$T_B = T_B^{\mathrm{T}}, \qquad T_E = T_E^{\mathrm{T}}.$$

We find that (28) also holds in this case. There, the solution is numerically stable, which is confirmed by our results (Fig. 10b). On the other hand, for a general non-orthogonal grid such as that shown in Fig. 7, the condition (28) does not hold, and thus some of the eigenvalues of the matrices M_E and M_B are complex numbers. In this case, the computations will be numerically unstable even with a time step below the Courant limit. One can speculate that since different weighting schemes result in different complex eigenvalues, different weighting schemes will lead to different numerical instability growth rates for the same grid. For the particular grid shown in Fig. 7, we speculate that the weighting scheme of Method A provides relatively more symmetry for M_E and M_B, and is thus relatively more stable.

We find that the symmetric face-to-edge transformation condition proposed by Langdon (1995) is a necessary but not sufficient condition for the DSI method to be

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

stable. The symmetric transformation condition due to Langdon ensures stability of the DSI method for a special 2D case, where T_B is diagonal, and $M_B = S_E^{\mathrm{T}} T_E S_E$. In such a case, T_E must indeed be symmetric in order for M_B to be symmetric and positive-definite. However, for a general 3D case, a symmetric face-to-edge transformation does not guarantee that the matrices M_B and M_E will always be symmetric. Therefore, once a grid has been chosen, we suggest that one should test all three weighting schemes to choose the most stable one for this particular grid and to determine the instability onset time. The fact that the DSI method is only weakly unstable and has typically a long instability onset time means that it can still be applied to many problems.

3.3. Accuracy of the EM algorithm

We next examine the accuracy of the electromagnetic field solve. We consider the same test problem used in Sec. 3.2, i.e. a planar electromagnetic wave in a rectangular box with distorted grids. In addition to the 2D distorted grid used in Sec. 3.2, (25), we also consider a grid distorted in all three directions,

$$x(i,j,k) = ih + \alpha \sin\left(2\pi \frac{ih}{L_x}\right) \sin\left(2\pi \frac{jh}{L_y}\right) \sin\left(2\pi \frac{kh}{L_z}\right), \tag{29a}$$

$$y(i,j,k) = jh + \alpha \sin\left(2\pi \frac{ih}{L_x}\right) \sin\left(2\pi \frac{jh}{L_y}\right) \sin\left(2\pi \frac{kh}{L_z}\right), \tag{29b}$$

$$z(i,j,k) = kh + \alpha \sin\left(2\pi \frac{ih}{L_x}\right) \sin\left(2\pi \frac{jh}{L_y}\right) \sin\left(2\pi \frac{kh}{L_z}\right), \tag{29c}$$

and a grid distorted in only one direction,

$$x(i,j,k) = ih + \alpha \sin\left(2\pi \frac{ih}{L_x}\right), \tag{30a}$$

$$y(i,j,k) = jh, \tag{30b}$$

$$z(i,j,k) = kh. \tag{30c}$$

The domain size is again taken to be $L_x = L_y = 10$ and $L_z = 20$. The results in Sec. 3.2 have shown that Method A has the lowest numerical instability growth rate for this type of grid, with an instability onset time of about 750 wave transit times. Therefore the numerical tests here all use Method A. The simulations are run for 10 wave transit times ($200h/c$).

We perform numerical tests for different grid resolutions h and grid distortions α, and compare numerical solutions with the analytical one. In the numerical tests presented here, the following four sets of grid numbers ($N_x \times N_y \times N_z$) are used: $8\times8\times16$ ($h = 1.25$), $16\times16\times32$ ($h = 0.625$), $32\times32\times64$ ($h = 0.3125$) and $64\times64\times128$ ($h = 0.1563$). For each grid resolution, we consider three grid distortions: $\alpha = 0.4$, 0.8 and 1.2.

In Fig. 11, we plot the result for the 3D distorted grid, (29a). We take the grid distortion to be $\alpha = 0.8$ and compare the total field energy for the four different grid resolutions as a function of time. Note that the total energy for the analytical solution is $E_{\mathrm{fld}} = 1000$. For the $8 \times 8 \times 16$ grids, the error in the field energy is $\delta E_{\mathrm{fld}}/E_{\mathrm{fld}} \approx 0.02$. This error is reduced significantly as one improves the grid resolution. For the $64 \times 64 \times 128$ grid, we find $\delta E/E_{\mathrm{fld}} \approx 10^{-4}$.

Since the *particle push* utilizes the field vectors at the vertices of the primary grid, $\mathbf{E}_{\mathrm{vert}}$ and $\mathbf{B}_{\mathrm{vert}}$, to push particles, we next evaluate the numerical error of the

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

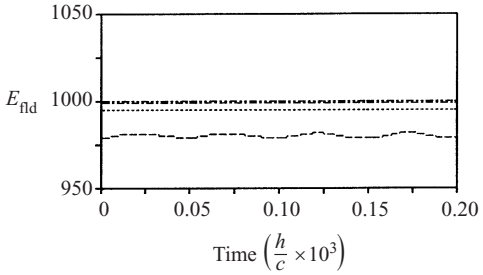


Figure 11. Accuracy test of the EM algorithm. Total electromagnetic field energy. 3D distorted grid with $\alpha = 0.8$. The number of grid points used are $8 \times 8 \times 16$ (dashed line), $16 \times 16 \times 32$ (dotted line), $32 \times 32 \times 64$ (dot-dashed line) and $64 \times 64 \times 128$ (dot-dot-dashed line).

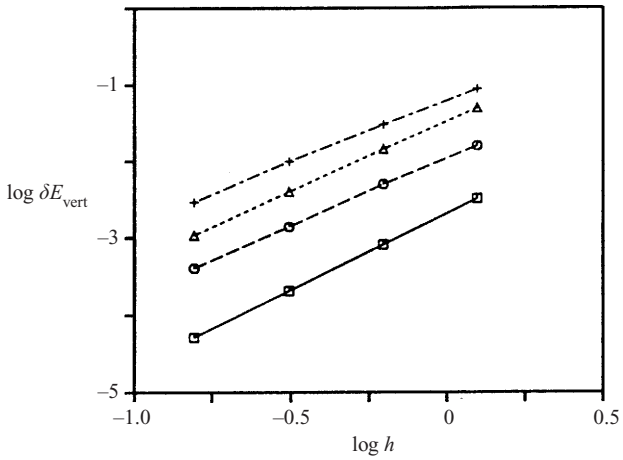


Figure 12. Error in the vertex E field, E_{vert}, at the location of maximum grid distortion. Grid distortions $\alpha = 0$ (solid line), $\alpha = 0.4$ (dashed line), $\alpha = 0.8$ (solid line) and $\alpha = 1.2$ (dot-dashed line).

field component at the vertices of the primary grid. We use the difference between the x component of the numerical $\mathbf{E}_{\text{vert}}$ and the analytical one at the location with maximum grid distortion, δE_{vert}, as a measure of the numerical error for the field solve. In Fig. 12, we plot δE_{vert} as a function of the grid resolution h for all three grid distortions as well as the orthogonal grid ($\alpha = 0$). The error shows a second-order convergence ($\delta E_{\text{vert}} = O(h^2)$) for the orthogonal grid, as expected. For the three non-orthogonal grids, our numerical tests show a convergence rate of $\delta E_{\text{vert}} = O(h^\beta)$ with $1.6 \leqslant \beta \leqslant 1.85$.

In Fig. 13, we compare δE_{vert} as a function of h for the 1D, 2D and 3D distorted grids, and the Cartesian grid. The grid distortion is taken to be $\alpha = 0.8$. For the 1D and 2D distorted grid, the results yield much better accuracy than for the 3D distortion case. For the 1D distorted grid, the accuracy is almost the same as that of for the Cartesian grid. This is because, for this particular test problem, the grid along the direction of spatial dependence of the solution, i.e. z, is not distorted in the 1D and 2D distorted grids. This suggests that, if possible, one may improve the overall accuracy by minimizing the distortion of the grid in the direction of spatial dependence of the solution.

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

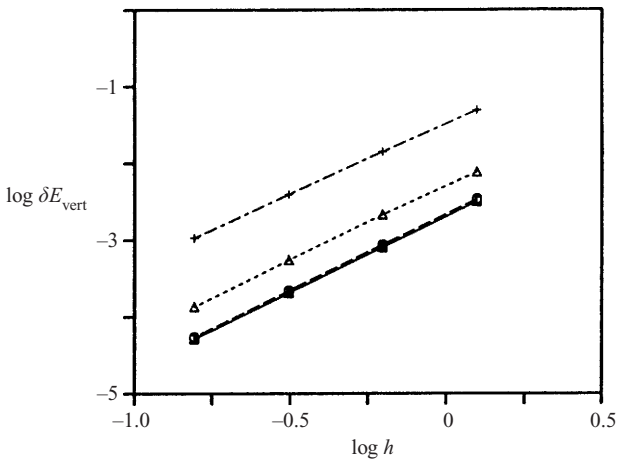


Figure 13. Error in E_{vert}: no distortion (solid line); 1D grid distortion (dashed line); 2D grid distortion (dotted line); 3D grid distortion (dot-dashed line). For all distorted grids, $\alpha = 0.8$.

3.4. Energy conservation of the EMPIC code

Finally we test the complete (field plus particle) non-orthogonal grid EMPIC code using energy conservation as a measure of code accuracy. For this test, we consider a relativistic counterstreaming beam system: two equal electron beams are set to counterstream in the x direction with drifting velocities $v_d = \pm 0.4c$. The electrons within each beam follow a Maxwellian distribution with thermal velocity $v_t = 0.05c$. The ions are considered as a fixed background. This counterstreaming system generates the well-known two-stream instability, which for the high drift velocity used here is electromagnetic in nature. The drifting speed is close to the speed of light, and the unstable wave generated has comparable electric and magnetic fields. The instability grows at the expense of the kinetic energy of the beams. However, the total energy of the system should stay as constant. Hence the two-stream instability provides a good test case for energy conservation of the code.

We consider a rectangular box domain with grids distorted in all three directions as described by (29a). For this test, the box size is $L_x = L_y = 10$ and $L_z = 20$; the number of grid points used is $12 \times 12 \times 24$; and the number of particles used is about 10^5. In Figs. 14(a) and (b), we plot the electromagnetic field energy E_{fld}, the particle kinetic energy E_{part}, and the total energy in the system E_{tot} as a function of time for a distorted grid $\alpha = 0.8$ and for the Cartesian grid $\alpha = 0$ respectively. Both results show similar behaviour of the energy transfer between the particles and the fields as a result of the instability excitation and saturation, and a constant total energy of the system. The numerical instability from the EM field solve discussed in Sec. 3.2 has no effect on these results, because the calculation only takes several wave transit times. In Fig. 14(c), we compare E_{tot}, for different grid distortions $\alpha = 0$, 0.2, 0.4 and 0.8. We find that the error in total energy conservation does not exceed 2 %, even for a highly distorted grid $\alpha = 0.8$.

4. Parallel implementation and performance benchmarks

As the algorithm is based on the use of logically connected hexahedral cells and the particles are moved in logical space, the parallel implementation of this code is

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

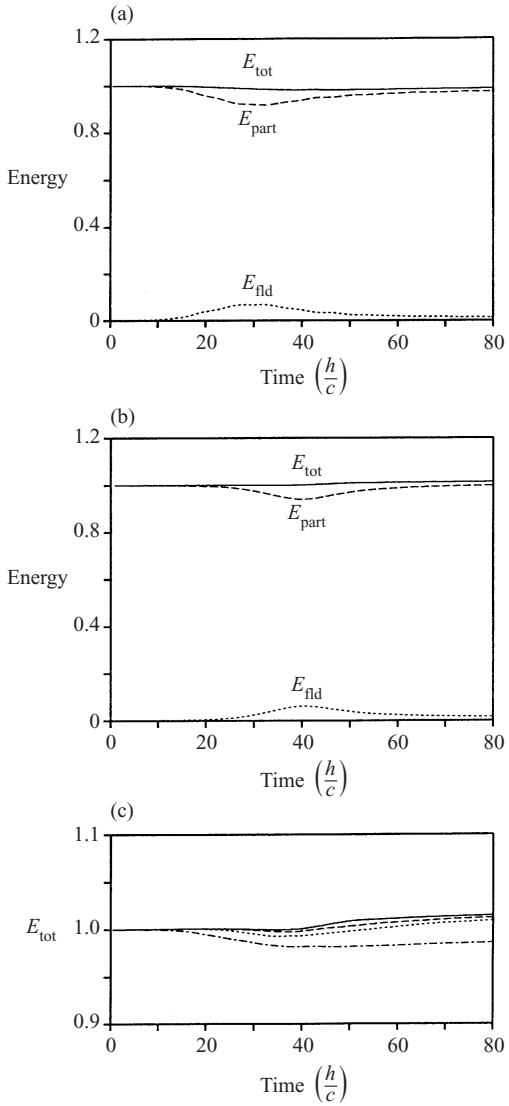


Figure 14. PIC code test: two-stream instability. Total energy, field energy and particle energy. (a) Grid distortion $\alpha = 0.8$. (b) Cartesian grid $\alpha = 0$. (c) Comparison of the total energy for a Cartesian grid (solid line), distortion with $\alpha = 0.2$ (dashed line) distortion with $\alpha = 0.4$ (dashed line), and distortion with $\alpha = 0.8$ (dotted line).

identical to that of the Cartesian-grid-based PIC code of Wang et al. (1995), which is implemented using the general concurrent PIC (GCPIC) algorithm (Liewer and Decyk 1989). The algorithm uses spatial decompositions of the physical domain to divide the computation among parallel processors. Each processor is assigned a subdomain and all the particles and grid points in it. When a particle moves from one subdomain to another, it must be passed to the appropriate processors, which requires interprocessor communication. However, the interpolations between the particles and the grids (gather/scatter) are done locally via the use of guard cells. Interprocessor communication is also necessary to exchange guard cell information

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

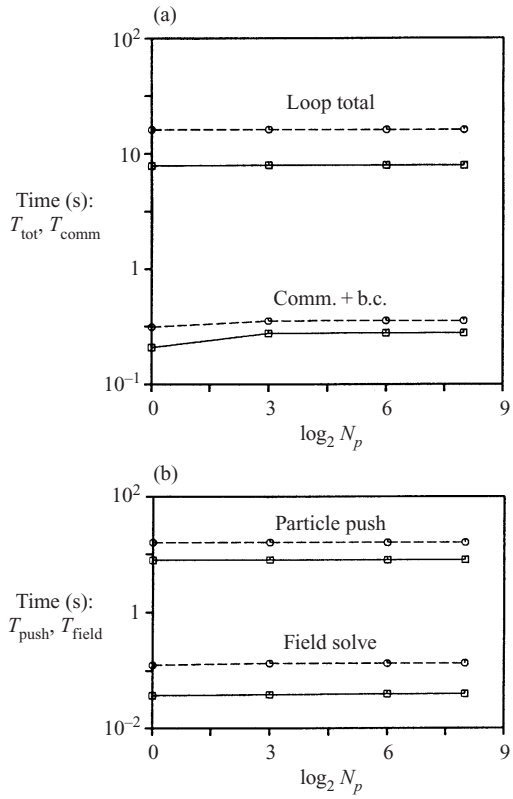


Figure 15. Run times of a Cartesian-grid EMPIC code (solid line) and the non-orthogonal grid EMPIC code (dashed line) on a 256-processor Cray T3D. (a) Total loop time T_{tot} and communication/boundary condition time T_{comm}. (b) Particle push time T_{push} and field solve time T_{field}.

for current deposit and field solve. In Wang et al. (1995), three communication algorithms for parallel EMPIC codes are described: a *Particle Trade* to trade particles between processors; a *Guard Cell Exchange* to update the **E** and **B** fields at processor boundary cells, and a *Guard Cell Summation* to add the currents deposited in guard cells to the proper cells in the neighbouring processors. The same communication algorithms are utilized here. The readers are referred to Wang et al. (1995) for detailed discussions on the parallel implementation of a EMPIC code.

To benchmark the performance of this code, we present timing result from running this code on the CRAY T3D parallel supercomputer at JPL. The CRAY T3D at JPL has 256 numerical nodes, each with a memory of 64 Mbytes (8 Mwords) and a peak speed of 150 Mflops. Hence the total memory size is 16.384 Gbytes (2.048 Gwords) and the total peak speed 38.4 Gflops. We shall use the two-stream instability test case described in the last section. We measure the total loop time per time step as well as the times spent by each major part of the code. Since each processor runs the code with slightly different times, the times measured are the maximum processor times on a parallel computer.

We perform a scaled problem size analysis, i.e. keeping the problem size on each processor fixed while increasing the number of processors used. We load the following problem: each processor has a cubic subdomain of $16 \times 16 \times 16$ cells and

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

3.16×10^5 particles (about 77 particles/cell). The problem size is scaled up in three dimensions. When this problem is loaded on all 256 nodes, the total problem size is $64 \times 128 \times 128$ (1.05 million) cells and about 80.8 million particles. The run times of this code as a function of the number of processors used, N_p, are shown in Fig. 15. For comparison, we also plot the results from running the Cartesian grid EMPIC code (Wang et al. 1995) in the same figure.

Figure 15(a) shows the total loop time $T_{\rm tot}$, and the time spent by the subroutines to trade particle and guard cell information between processors, $T_{\rm comm}$. Since the global conditions are applied within the same subroutines for interprocessor communications, $T_{\rm comm}$ is the sum of the time spent by the code on communications and global boundary conditions. When the number of processors used is much larger then one, $T_{\rm comm}$ is dominated by the interprocessor communication time, and hence it is a good measure of the parallel communication cost.

We find that, similarly to the Cartesian grid EMPIC code, $T_{\rm tot}$ for the non-orthogonal-grid PIC code stay almost constant as the number of processors is increased, indicating a high parallel efficiency. $T_{\rm comm}$ only occupies a negligible portion of $T_{\rm tot}$. The parallel efficiency $\epsilon \approx 1 - T_{\rm comm}/T_{\rm tot}$ is about 0.96 and 0.98 for the Cartesian and non-orthogonal grid codes, respectively.

Figure 15(b) shows the times spent by the two stages of the EMPIC codes. We define the particle push time $T_{\rm push}$ as the sum of the times for moving particles, depositing currents, applying boundary conditions, and related interprocessor communications, and the field solve time $T_{\rm field}$ as the sum of the times for updating the E and B fields, applying boundary conditions, and related interprocessor communications. Not surprisingly the new, non-orthogonal grid PIC code runs slower than the Cartesian grid PIC code because it involves significantly more computations. For the Cartesian grid code, we have $T_{\rm push} \simeq 7.97$s and $T_{\rm field} \simeq 0.039$s. While for the non-orthogonal grid code, $T_{\rm push} \simeq 16$s and $T_{\rm field} \simeq 0.135$s. The particle push time per particle per time step $T_{\rm push}/part$ is 98.6ns for the Cartesian grid code and 198ns for the non-orthogonal grid code. Hence, the overall computing speed of the non-orthogonal grid PIC code is about half of that of the Cartesian grid PIC code.

5. Summary and conclusions

We have developed a three-dimensional non-orthogonal grid electromagnetic PIC code for parallel supercomputers. The numerical formulation is based on a finite-volume approach. The computation grids are hexahedral cells that are logically connected cubic cells but distorted to body-fit complex geometries in physical space. The *particle push* algorithm is a hybrid logical–physical space operation that calculates particle velocity in physical space but performs particle position update and particle–grid interpolations in logical space. The discrete surface integral algorithm for *field solve* is an extension of the classical staggered mesh finite-difference time domain (Yee 1996) to non-uniform meshes (Madsen 1995; Gedney and Lansing 1995), and reduces to the standard FDTD algorithm when the grid is Cartesian. The parallel implementation of the code is almost identical to that of the Cartesian-grid EMPIC code (Wang et al. 1995). We show that the accuracy of the new hybrid particle push algorithm is second-order in time and space (as is the standard leapfrog scheme), and is linearly proportional to the grid distortion. While the classical FDTD algorithm has a second-order accuracy in space, we find that the accuracy of the DSI field solve algorithm is between first- and second-order for

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press

non-orthogonal grids. In addition, we find that the DSI algorithm is weakly unstable for certain non-orthogonal grids, even when the Courant condition is well satisfied. The instability growth rate does not depend on the value of the time step, but rather is controlled by the properties of the face-to-edge transformation matrix, which in turn depends upon the grid geometry and the weighting scheme. However, since the instability onset may take several hundreds of wave transit time even for large grid distortions, we find that the DSI method can still be applied to many problems. Numerical tests of this EMPIC code show that the error in energy conservation does not exceed about 2 % even when highly distorted grids are used. The performance of the code is evaluated on a 256-processor CRAY T3D. We find that the code runs with a high parallel efficiency $\epsilon > 96\,\%$. However, since the non-orthogonal-grid code involves significantly more calculations, the speed of the non-orthogonal-grid code is only about a half of that of the Cartesian-grid code of Wang et al. (1995).

Acknowledgements

We should like to acknowledge useful discussions with J. U. Brackbill (LANL), V. Decyk (UCLA), E. Huang (JPL), R. J. Kaves (LANL), A. B. Langdon (LLNL) and P. Rambo (LLNL). This work was supported by the AFOSR Computational Mathematics Program under Grant F49620-94-1-0336, and was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA. Access to the JPL/Caltech CRAY T3D supercomputer used in this study was provided by funding from NASA Offices of Mission to Planet Earth, Aeronautics, and Space Science.

References

Birdsall, C. K. and Langdon, A. B. 1985 *Plasma Physics via Computer Simulation*. McGraw-Hill, New York.

Brandon, S. T. and Rambo, P. W. 1995 Stability of the DSI electromagnetic update algorithm on a Chevron grid. 22nd IEEE International Conference on Plasma Science, June 1995.

Eastwood, J., Arter, W., Brealey, N. and Hockney, R. 1995 Body-fitted electromagnetic PIC software for use on parallel computers. *Comput. Phys. Commun.* **87**, 157–178.

Gedney, S. and Lansing, F. 1995 A generalized Yee-algorithm for the analysis of microwave circuit devices. Submitted to *IEEE Trans. Microwave Theory Techniques*.

Hockney, R. W. and Eastwood, J. W. 1981 *Computer Simulation Using Particles*. McGraw-Hill, New York.

Langdon, A. B. 1995 2D electromagnetic and PIC on a quadrilateral mesh. Memorandum UCB/ERL M95/96, University of California, Berkeley.

Liewer, P. C. and Decyk, V. K. 1989 A general concurrent algorithm for plasma particle-in-cell simulation codes. *J. Comput. Phys.* **85**, 302–322.

Madsen, N. K. 1995 Divergence preserving discrete surface integral methods for Maxwell's curl equations using non-orthogonal unstructured grid. *J. Comput. Phys.* **119**, 34–45.

Villasenor, J. and Buneman, O. 1992 Rigorous charge conservation for local electromagnetic field solvers. *Comput. Phys. Commun.* **69**, 306–316.

Wang, J., Liewer, P. and Decyk, V. 1995 3D electromagnetic plasma particle simulations on a MIMD parallel computer. *Comput. Phys Commun.* **87**, 35–53.

Westermann, T. 1992 Localization schemes in 2D boundary-fitted grids. *J. Comput. Phys.* **101**, 307–313.

Yee, K. S. 1966 Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Antennas Propag.* **14**, 302–307.

https://doi.org/10.1017/S0022377899007552 Published online by Cambridge University Press