Journal of Computational Physics 237 (2013) 56–60

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Short note

Note on the use of Yee-lattices in (semi-) implicit particle-in-cell codes

Andreas Kempf*, Urs Ganse, Patrick Kilian, Felix Spanier

Lehrstuhl für Astronomie, Universität Würzburg, Emil-Fischer-Straße 31, D-97074 Würzburg, Germany

ARTICLE INFO

Article history:
Received 14 June 2012
Received in revised form 8 November 2012
Accepted 26 November 2012
Available online 13 December 2012

Keywords:
Particle in cell
Divergence
Yee
Implicit timestep

ABSTRACT

A modification of the implicit algorithm for particle-in-cell simulations proposed by Petrov and Davis (2011) [1] is presented. The original lattice arrangement is not inherently divergence-free, possibly leading to unphysical results. This arrangement is replaced by a staggered mesh resulting in a reduction of the divergence of the magnetic field by several orders of magnitude.

1. Introduction

In order to correctly reproduce physical processes in a particle-in-cell code, Maxwell's equations need to be solved consistently. However, the requirement of the magnetic field being divergence free is often violated by numerical algorithms leading to unphysical results [2]. Consequently, several divergence-cleaning schemes have been proposed, providing a way to remove magnetic source terms after the fact. Another possibility to correctly incorporate Gauss' law for magnetism into a PiC-code is to use the staggered mesh first proposed by Yee [3] in 1966. The special arrangement of electric and magnetic fields inherently conserves a zero-valued divergence [4], provided that $\nabla \cdot \vec{B} = 0$ at $t = 0$. A comparison by Balsara and Kim [5] identifies several problems of divergence-cleaning methods in MHD and notes their absence when using a staggered mesh.

Petrov and Davis [1] proposed an implicit particle-in-cell algorithm forgoing the staggered mesh approach. Continuing previous work by Kilian et al. [6] we intend to use this algorithm to study particle acceleration in astrophysical plasmas while keeping unphysical effects to a minimum. In this paper we therefore modify the scheme, incorporating the Yee lattice and effecting a reduction of $\nabla \cdot \vec{B}$ by several orders of magnitude.

2. Definitions

The quantities from [1] that are relevant to this paper are

$$\hat{S}_\alpha^{n+1/2} = \frac{n_\alpha q_\alpha}{4\varepsilon_0 m_\alpha \gamma_\alpha^{n+1/2}} \hat{T}_\alpha^{n+1/2} \tag{1}$$

* Corresponding author.
E-mail address: akempf@astro.uni-wuerzburg.de (A. Kempf).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2012.11.045

and

$$\delta\vec{j}_\alpha^{n+1/2} = \frac{n_\alpha q_\alpha}{2m_\alpha \gamma_\alpha^{n+1/2}} \left(\vec{p}_\alpha^{\,n} + \hat{T}_\alpha^{n+1/2}\left(\vec{p}_\alpha^{\,n} \times \Delta\vec{\Omega}_\alpha^{n+1/2}\right)\right). \tag{2}$$

The tensor $\hat{T}$ (with indices suppressed for brevity) is defined as

$$\hat{T} = \frac{1}{1+|\Delta\vec{\Omega}|^2} \begin{bmatrix} 1+\Delta\Omega_x^2 & \Delta\Omega_x\Delta\Omega_y + \Delta\Omega_z & \Delta\Omega_x\Delta\Omega_z - \Delta\Omega_y \\ \Delta\Omega_x\Delta\Omega_y - \Delta\Omega_z & 1+\Delta\Omega_y^2 & \Delta\Omega_y\Delta\Omega_z + \Delta\Omega_x \\ \Delta\Omega_x\Delta\Omega_z + \Delta\Omega_y & \Delta\Omega_y\Delta\Omega_z - \Delta\Omega_x & 1+\Delta\Omega_z^2 \end{bmatrix}, \tag{3}$$

with

$$\Delta\vec{\Omega}_\alpha^{n+1/2} = \frac{q_\alpha \vec{B}_\alpha^{n+1/2}}{m_\alpha \gamma_\alpha^{n+1/2}} \frac{\Delta t}{2}. \tag{4}$$

The quantities $q_\alpha,\ m_\alpha,\ n_\alpha$ are the charge, mass, and number density of (computational) particle α. $\gamma_\alpha^{n+1/2}$ is the particle's relativistic gamma factor and $\vec{B}_\alpha^{n+1/2}$ its local magnetic field at time $n+1/2$. $\vec{p}_\alpha^{\,n}$ is the momentum of particle α at time n.

The deposition of $\hat{S}$ and $\delta\vec{j}$ on the grid and the interpolation of $\vec{E}$ and $\vec{B}$ to the particle position is achieved via a standard weighting function. Our algorithm makes use of the triangular shaped cloud (TSC) scheme.

3. The modified lattice arrangement

The original algorithm by Petrov and Davis [1] stores electric fields on grid nodes and magnetic fields in the cell center. The vector quantity $\delta\vec{j}$ and the tensor quantity $\hat{S}$ are deposited on grid nodes, as well. Since the electric field is updated according to

$$\left(\hat{I} + \hat{S}^{n+1/2}\right)\vec{E}^{n+1} = \left(\hat{I} - \hat{S}^{n+1/2}\right)\vec{E}^{n} + \frac{\Delta t}{\varepsilon_0}\left(\vec{\nabla}\times\vec{H}^{n+1/2} - \delta\vec{j}^{n+1/2}\right) \tag{5}$$

and all required quantities are defined on grid nodes, this equation can be solved locally for $\vec{E}^{n+1}$.

Our approach keeps the original field layout by Yee [3] with the components of $\delta\vec{j}$ stored like the corresponding components of the electric field. $\hat{S}$ is stored on grid nodes and interpolated linearly for each component of the electric field to be calculated. When calculating the new value for $E_x^{i+1/2,j,k}$, $\hat{S}$ is taken to be $(\hat{S}^{i,j,k} + \hat{S}^{i+1,j,k})/2$, for $E_y^{i,j+1/2,k}$ it is $(\hat{S}^{i,j,k} + \hat{S}^{i,j+1,k})/2$ and for $E_z^{i,j,k+1/2}$ it is $(\hat{S}^{i,j,k} + \hat{S}^{i,j,k+1})/2$.

Since $\hat{S}$ is not a diagonal tensor, all the components of $\delta\vec{j}$, $\nabla\times\vec{B}$ and $\vec{E}^n$ need to be known at the same point as the component of $\vec{E}^{n+1}$ to be calculated, as well. These three quantities can be interpolated the same way.

For E_x^{n+1}:

$$A_x^{i+1/2,j,k} = A_x^{i+1/2,j,k}, \tag{6}$$

$$A_y^{i+1/2,j,k} = \left(A_y^{i,j+1/2,k} + A_y^{i,j-1/2,k} + A_y^{i+1,j+1/2,k} + A_y^{i+1,j-1/2,k}\right)\Big/4, \tag{7}$$

$$A_z^{i+1/2,j,k} = \left(A_z^{i,j,k+1/2} + A_z^{i,j,k-1/2} + A_z^{i+1,j,k+1/2} + A_z^{i+1,j,k-1/2}\right)\Big/4. \tag{8}$$

For E_y^{n+1}:

$$A_x^{i,j+1/2,k} = \left(A_x^{i+1/2,j,k} + A_x^{i+1/2,j+1,k} + A_x^{i-1/2,j,k} + A_x^{i-1/2,j+1,k}\right)\Big/4, \tag{9}$$

$$A_y^{i,j+1/2,k} = A_y^{i,j+1/2,k}, \tag{10}$$

$$A_z^{i,j+1/2,k} = \left(A_z^{i,j,k+1/2} + A_z^{i,j+1,k+1/2} + A_z^{1,j,k-1/2} + A_z^{i,j+1,k-1/2}\right)\Big/4. \tag{11}$$

For E_z^{n+1}:

$$A_x^{i,j,k+1/2} = \left(A_x^{i+1/2,j,k} + A_x^{i+1/2,j,k+1} + A_x^{i-1/2,j,k} + A_x^{i-1/2,j,k+1}\right)\Big/4, \tag{12}$$

$$A_y^{i,j,k+1/2} = \left(A_y^{i,j+1/2,k} + A_y^{i,j+1/2,k+1} + A_y^{ij-1/2,k} + A_y^{i,j-1/2,k+1}\right)\Big/4, \tag{13}$$

$$A_z^{i,j,k+1/2} = A_z^{i,j,k+1/2}. \tag{14}$$