Journal of Computational Physics 318 (2016) 305–326

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

The Plasma Simulation Code: A modern particle-in-cell code with patch-based load-balancing

Kai Germaschewski [a,*], William Fox [c], Stephen Abbott [d], Narges Ahmadi [a], Kristofor Maynard [a], Liang Wang [a], Hartmut Ruhl [b], Amitava Bhattacharjee [c]

[a] Space Science Center & Department of Physics, University of New Hampshire, Durham, NH, United States
[b] Faculty of Physics, Ludwig Maximilians University, München, Germany
[c] Princeton Plasma Physics Laboratory, Princeton, NJ, United States
[d] Oak Ridge National Laboratory, Oak Ridge, TN, United States

ARTICLE INFO

Article history:
Received 4 November 2015
Received in revised form 3 May 2016
Accepted 4 May 2016
Available online 9 May 2016

Keywords:
Particle-in-cell
Kinetic
Plasma
Load balancing

ABSTRACT

This work describes the Plasma Simulation Code (PSC), an explicit, electromagnetic particle-in-cell code with support for different order particle shape functions. We review the basic components of the particle-in-cell method as well as the computational architecture of the PSC code that allows support for modular algorithms and data structure in the code. We then describe and analyze in detail a distinguishing feature of PSC: patch-based load balancing using space-filling curves which is shown to lead to major efficiency gains over unbalanced methods and a previously used simpler balancing method.

1. Introduction

Rapidly advancing computer technology has enabled large first-principles plasma simulations in recent years. The kinetic description of plasmas, while computationally much more expensive, overcomes many limitations of fluid descriptions like magnetohydrodynamics (MHD) or extended MHD models. Fluid models describe plasma behavior at large scales very well, but approximations need to be made at small scales, which occur in magnetic reconnection and turbulence. For example, the one-fluid approximation breaks down at the ion skin depth scale d_i, electrons and ions decouple, and the magnetic field remains frozen to the electron flow. Reconnection requires breaking the frozen-in condition that occurs at electron scales, which can be represented in a fluid model in a generalized Ohm's Law that includes electron inertia and electron pressure tensor effects. Finding appropriate closures is still an area of active research, see e.g. [1]. The kinetic description of a plasma evolves the full 6-dimensional distribution function in time, either directly solving the Vlasov–Maxwell–Boltzmann equations (see e.g. [2]), or using the particle-in-cell method that this work focuses on. The particle-in-cell method is typically much less computationally intensive as it approximates the velocity part of the distribution function by quasi-particles which are evolved in 3-d configuration space, but also introduces complications like discrete particle noise, mapping between two spatial domains (a discrete mesh and continuous particle positions) and potential numerical instabilities. Kinetic models also allow investigation of problems entirely beyond the scope of fluid models, e.g. particle acceleration [3]. Particle-in-cell

* Corresponding author.
E-mail addresses: kai.germaschewski@unh.edu (K. Germaschewski), wfox@pppl.gov (W. Fox), abbottsr@ornl.gov (S. Abbott), narges.ahmadi@unh.edu (N. Ahmadi), k.maynard@unh.edu (K. Maynard), liang.wang@unh.edu (L. Wang), hartmut.ruhl@uni-muenchen.de (H. Ruhl), amitava@princeton.edu (A. Bhattacharjee).

http://dx.doi.org/10.1016/j.jcp.2016.05.013
0021-9991/

codes, while often run with modified physical parameters (e.g. reduced ion/electron mass ratio and speed of light), are now capable of simulating multi-scale problems spanning from electron through ion to global scales reaching 100's of d_i in two and even three dimensions. While efforts are underway to overcome some of the algorithmic limitations of explicit particle-in-cell methods (see, e.g., [4,5]), these methods scale efficiently to the largest supercomputers available today and are commonly used to address challenging science problems.

The Plasma Simulation Code (PSC) is an explicit, electromagnetic particle-in-cell code implementing similar methods as, e.g., VPIC [6], OSIRIS [7] and VORPAL [8]. PSC is based on H. Ruhl's original version [9], but has been rewritten as modular code that supports flexible algorithms and data structures. Beyond its origin in the field of laser-plasma interaction, PSC has been used in studies of laser-produced plasma bubbles [10–13], particle acceleration [14], and closure aspects in magnetic reconnection [15].

In this paper, we will review the main underlying particle-in-cell methods and the computational infrastructure of the PSC code. We will then introduce and analyze a distinguishing feature implemented in PSC: Patch-based dynamic load balancing addresses both performance and memory issues in simulations where many particles move between local domains.

2. Particle-in-cell method

2.1. Kinetic description of plasmas

The particle-in-cell method [16,17,5] solves equations of motion for particles and Maxwell's equations to find forces between those particles, which is very similar to the first-principle description of a plasma as a system of charged particles. It is, however, better understood as a numerical method to solve the Vlasov–Maxwell system of equations that describes the time evolution of the particle distribution function $f_s(\mathbf{x}, \mathbf{p}, t)$ where s indicates the species:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} + q_s(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f_s}{\partial \mathbf{p}} = 0 \tag{1}$$

The electromagnetic fields $\mathbf{E}$ and $\mathbf{B}$ are self-consistently evolved using Maxwell's equations:

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \tag{2}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{3}$$

$$\frac{\partial \mathbf{E}}{\partial t} = c^2 \nabla \times \mathbf{B} - \frac{\mathbf{j}}{\epsilon_0} \tag{4}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \tag{5}$$

where charge density ρ and current density $\mathbf{j}$ are obtained from the particle distribution functions:

$$\rho = \sum_s q_s \int f_s(\mathbf{x}, \mathbf{p}, t)\, d^3p \tag{6}$$

$$\mathbf{j} = \sum_s q_s \int \mathbf{v} f_s(\mathbf{x}, \mathbf{p}, t)\, d^3p \tag{7}$$

If the divergence equations (2), (3) in Maxwell's equations are satisfied at some initial time, it is easy to show from Ampére's Law (4) and Faraday's Law (5) that they will remain satisfied at all times provided that the charge continuity equation also holds:

$$\partial_t \rho + \nabla \cdot \mathbf{j} = 0. \tag{8}$$

This is important in the numerical implementation, where only Eqns. (4) and (5) are used to advance the electromagnetic fields in time. Even if Eqns. (2), (3) are discretely satisfied initially, large unphysical violations of Eqns. (2), (3) can accumulate, which is prevented here by using a compatible exact discrete implementation of Eqn. (8).

2.1.1. Particle-in-cell method

The particle-in-cell method approximates the distribution function f_s by representing it using quasi-particles with finite extent in configuration space:

$$f_s(\mathbf{x}, \mathbf{p}, t) = \sum_{i=1}^{N_s} N_i^s \phi(\mathbf{x} - \mathbf{x}_i^s(t))\, \delta^3(\mathbf{p} - \mathbf{p}_i^s(t)) \tag{9}$$

Using the δ-function in velocity space ensures that the spatial extent of each quasi-particle remains constant in time.