Progress towards Physics-Based Space Weather Forecasting with Exascale Computing

Maria Elena Innocenti[a], Alec Johnson[a], Stefano Markidis[b], Jorge Amaya,[a], Jan Deca[c], Vyacheslav Olshevsky[a], Giovanni Lapenta[a]

[a] *Center for Mathematical Plasma Astrophysics, Departement Wiskunde, KULeuven, University of Leuven, Celestijnenlaan 200B - bus 2400, B-3001 Heverlee, Belgium.*
[b] *Department of Computational Science and Technologies, KTH Royal Institute of Technology, Stockholm, Sweden*
[c] *Laboratory for Atmospheric and Space Physics (LASP), University of Colorado, 1234 Innovation Drive, Boulder, CO 80303-7814, USA*

Abstract

Space weather is a rapidly growing field of science which studies processes occurring in the area of space between the Sun and the Earth. The development of space weather forecasting capabilities is a task of great societal relevance: space weather effects may damage a number of technological assets, among which power and communication lines, transformers, pipelines and the telecommunication infrastructure. Exascale computing is a fundamental ingredient for space weather forecasting tools based on physical, rather than statistical, models. We describe here our recent progresses towards a a physics-based space weather forecasting tool with exascale computing. We select the semi-implicit, Particle In Cell, Implicit Moment Method implemented in the parallel, object-oriented, C++ iPic3D code as a promising starting point. We analyse the structure and the performances of the current version of the iPic3D code. We describe three algorithmic developments, the fully implicit method, the Multi-Level Multi-Domain method, and the fluid-kinetic method, which can help addressing the multiple spatial and temporal scales present in space weather simulations. We then examine, in a co-design approach, which requirements - vectorisation, extreme parallelism and reduced communication - an application has to satisfy to fully exploit architectures such as GPUs and Xeon Phi's. We address how to modify the iPic3D code to better satisfy these requirements. We then describe how to port the iPic3D code to the DEEP architecture currently under construction. The FP7 project DEEP (www.deep-project.eu) aims at building an exascale-ready machine composed of a cluster of Xeon nodes and of a collection of Xeon Phi coprocessors, used as boosters. The aim of the DEEP project is to enable exascale performance for codes, such as iPic3D, composed of parts which exhibit different potential for extreme scalability. Finally, we provide examples of simulations of space weather processes done with the current version of the iPic3D code.

Keywords: Space Weather, Particle-In-Cell, Adaptive, Implicit, Exascale, High Performance Computing

1. Introduction

Space weather [1, 2, 3, 4] is the fast-growing area of science that focuses on the conditions in the space amidst the Sun, the Earth and the other planets of our solar system. The Sun is an extremely dynamic source of energy. Besides the electromagnetic radiation (including of course light) that makes life possible, the Sun is a giant dynamo that produces a strongly magnetized plasma. The plasma flows away from the Sun at supersonic speeds, arriving at the Earth with velocities ranging from about 200 to 800 Km/s. This wind is very dynamic and varies greatly. Just like Earth winds, it also carries storms. The most powerful space storms are generated irregularly in singular events of mass and energy released by the Sun, called coronal mass ejections. The intensity and frequency of these storms oscillates over the course of a solar cycle, with a period of about 11 years. Peak activity is called solar maximum (solar maxima occurred in 2000 and in 2013) and is separated by periods of very low activity at solar minimum (2008-2009). Solar activity also

changes on much longer scales. Generations of scientists have monitored the solar activity since the 17th century. The solar cycles of the space age, following the first space trip of Yuri Gagarin, have happened to be particularly active. However, a period of prolonged quietness lasting several decades happened in the end of the 17th century. It is called the Maunder minimum and corresponds to a period of exceptionally cold winters and insufficient crop production in Europe. Since this period, at least one exceedingly strong event, the Carrington event, happened in 1859, causing severe telegraph disruptions back in those early industrial revolution days. A similar event now would wreak havoc on our space technology and on our ground infrastructure, such as, as examples, power and communication lines, transformers, pipelines and telecommunication infrastructure. A Carrington-sized event now could not only render multiple satellites permanently inoperative but could damage beyond repair the largest transformers on the highest voltage long distance power lines. These transformers and satellites would require several months to replace. One can clearly see that a blackout lasting months would greatly disrupt the economy and social tranquility. One has also to take into account that past blackouts caused by space weather events involved especially the northeast of the American continent, because of the geographical location of the magnetic north pole. The area includes some of the world's most active financial trading and data centers. This brief summary explains why space weather has become a key area of scientific computing. Predicting space weather is a top priority to defend key societal infrastructures and to expand the human presence in space. All major industrialized and developing countries are developing a space weather forecasting capability, with both military and civil agencies contributing to the effort at national and international levels. All these activities center on one key ingredient: the reliance on physics-based high performance computing codes that can follow a space storms from its origin on the Sun to its consequences at Earth. These models are very complex and must include many processes and aspects, including the sheer size of the vastness of the space involved and the great variety of conditions encountered from the hot solar corona to the cold depths of deep space. Density and temperature change over many orders of magnitude. In space, matter is in the form of plasma. Atoms are ionized and electrons are freed from their bond to the nuclei of atoms. Electrons, being much lighter than the nuclei (for hydrogen the mass ratio between nuclei and electrons is 1836), respond on much smaller scales. The physics-based description of space (see, for example, [5] for an alternative, statistical description of the solar wind propagation from the Sun to Earth) requires handling multiple scales in space and time and multiple physical processes. Dealing with multi-scale and multi-physics systems is the key challenge in modern high performance computing (HPC), common to many other engineering and science applications. Even the largest supercomputers can only cover a limited range of scales and physics, posing a great challenge to co-design: how to design new computer architectures and new algorithms so that the gaps are filled and a complete answer can be found for multi-scale and multi-physics challenges.

Here we detail the approach followed in our co-design effort for HPC applied to space weather modeling. The paper is organised as follows. First, we describe our target application: kinetic modelling of space weather, using the Particle In Cell (PIC) approach. Section 2 describes the basics of the Particle In Cell method, i.e. space discretisation of field values at grid points and representation of ions and electrons as computational particles moving in the continuum. In Section 2.1 we describe an implicit temporal discretisation of the equations for the temporal evolution of fields and particles. The implicit discretisation has the advantage of making PIC simulations computationally cheaper with respect to explicit formulations. It relaxes the constraints in terms of spatial and temporal resolution for the stability of the simulation. It also has the algorithmic disadvantage of coupling the equations for field and particle time advancement: to advance in time the electric and magnetic fields, we now need information on particle position and velocity at the future time step, and vice versa. We describe in Section 2.3 the Implicit Moment Method (IMM), a method to remove the coupling between the field and particle sets of equations. The IMM has been implemented in the parallel, object-oriented, C++ code iPic3D [6], described in Section 2.4. In Section 2.5, the iPic3D code is broken into four logical blocks. The relative weight of these blocks in terms of execution time and communication requirements is investigated on different clusters and with different problem sizes. The Scalasca performance analysis tool is used in the investigation. Section 3 and Section 4 are the core of this paper. There, we analyse the progress that we recently achieved and that will enable an effective and efficient use of PIC methods for space weather forecasting. The progress reported follows two intertwined lines: development of new algorithms and identification of co-design issues.

On the algorithmic line (Section 3), we report model development designed to handle multiple spatial and temporal scales. The temporal discretization has been improved to reach exact energy conservation (Section 3.1), a crucial aspect to guarantee that simulations faithfully reproduce reality. The discretization capabilities have been expanded: it is now possible to use multiple levels of spatial and temporal resolution to describe different levels of physical refinement in different regions. The resulting method, the Multi-Level Multi-Domain method, is described in Section 3.2. Finally, a new hybrid approach that extends the ability to include multiple physical descriptions based on kinetic or fluid methods has been developed (Section 3.3) Co-design aims at establishing continuous and effective feedback between the code development and machine design stages. On one side, algorithms must be deviced as to exploit at best the characteristics of exascale-oriented machines: parallelisation must be increased, communication reduced. On the other end, the requirements coming from cutting-edge scientific fields, such as space weather, have to be considered in the design of next generation HPC machines. In Section 4 we describe how the issues of vectorisation, communication and parallelisation (Section 4.1) apply to the code iPic3D and to fully kinetic simulations of space weather, and in which directions we plan on developing iPic3D to better comply with the programming models required to exploit at best GPU/ Xeon Phi architectures (Section 4.2). We also report our experience (as of 2014) within the DEEP project, which aims at building an exa-scale enabling machine constituted by a traditional cluster of Xeon nodes plus a collection of Xeon Phi coprocessors (Section 4.3). Finally, in Section 5, we conclude with some examples of the capabilities of the IMM and of the iPic3D code of simulating space weather relevant processes: transfer of magnetic energy to particles in simulations of null points dynamics in 3D (Section 5.1) and solar wind/ magnetosphere interaction (Section 5.2).

2. Target Application: Kinetic modeling of space weather

The fundamental agents of a space weather simulation are the electric and magnetic fields that permeate space and the particles moving through them.

The field evolution is described by the equations of Maxwell

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t}, \\ \nabla \times \mathbf{B} &= \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \mathbf{J}, \end{aligned} \tag{1}$$

for the electric field $\mathbf{E}$ and the magnetic field $\mathbf{B}$, with ϵ_0 and μ_0 the permittivity and permeability of free space respectively. The particle evolution is described by the equations of Newton,

$$\begin{aligned} \frac{d\mathbf{x}_p}{dt} &= \mathbf{v}_p, \\ \frac{d\mathbf{v}_p}{dt} &= \frac{q_s}{m_s}(\mathbf{E}_p + \mathbf{v}_p \times \mathbf{B}_p), \end{aligned} \tag{2}$$

for the position $\mathbf{x}_p$ and velocity $\mathbf{v}_p$ of each particle labelled by p. Particles are organized in species (e.g. electrons, ions) labelled by s and sharing common properties such as the ratio of charge to mass, $q_s/m_s = q_p/m_p$. Particle motion feeds back into field evolution by means of the *current* $\mathbf{J}(\mathbf{x}) := \sum_p q_p \mathbf{v}_p \delta(\mathbf{x} - \mathbf{x}_p)$ (i.e. the flux of charge per unit area per time); here δ is the unit-volume impulse function of Dirac. Equations (1) maintain the *divergence conditions*

$$\begin{aligned} \epsilon_0 \nabla \cdot \mathbf{E} &= \rho, \\ \nabla \cdot \mathbf{B} &= 0, \end{aligned} \tag{3}$$

where $\rho(\mathbf{x}) := \sum_p q_p \delta(\mathbf{x} - \mathbf{x}_p)$ is the *charge density*.

In some instances in space weather, relativistic effects need to be considered. Their inclusion does not introduce particular complexities, but for simplicity of presentation they are neglected here.

To solve this interacting system, where the sources of the Maxwell's equations (the charge density ρ and current $\mathbf{J}$) are determined by the particles and the forces on the particles are determined by the fields, the most commonly used method is the Particle In Cell (PIC) approach. Time is discretized in fixed intervals of length Δt. A grid is introduced to discretize the field equations. The knowledge of the system is then provided by the position and velocity of all particles at the discrete time levels $t^n = n\Delta t$ and of the fields at prescribed discrete grid locations and at prescribed times: $\mathbf{E}_g^n$, $\mathbf{B}_g^n$. Here the superscript n is used to label time, and the subscript g labels the grid points. A single label is used regardless of the dimensionality of the grid.

The PIC approach is shown in Fig. 1.

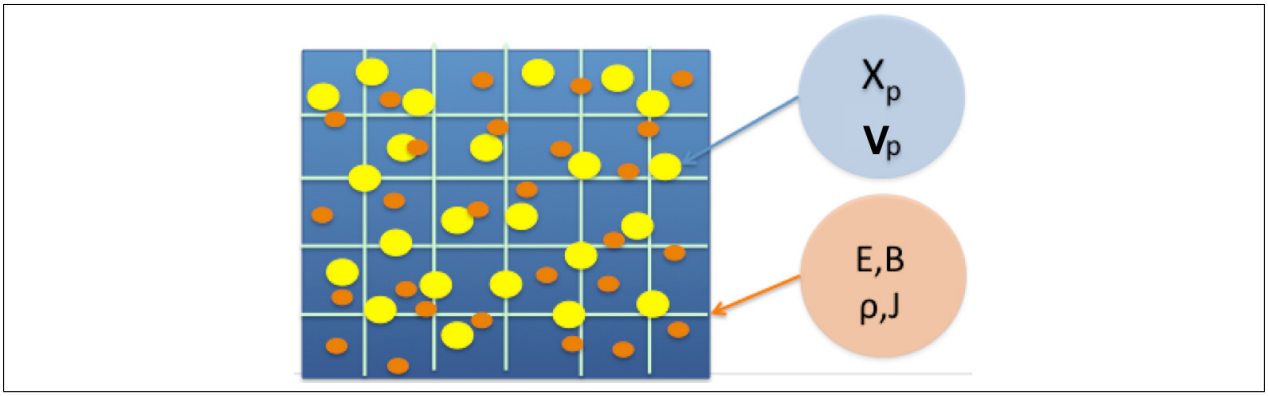


Figure 1: Conceptual representation of the PIC method. The fields are computed on a grid (shown here in 2D for visual simplicity but 3D in the code used here). The particles move in the continuum space. To guide the eye, only the center of the particles is shown, but each particle has a finite square (cubic in 3D) shape around the particle center. The electrons are shown in orange (darker color in the black and white version) while the ions in yellow (lighter color). To remind the reader that the ions are 1836 times more massive than the electrons, their dot is bigger. In a real simulation there are on average hundreds of particles per cell, but for clarity the figure only shows a few.

A first fundamental feature of the PIC approach is evident: the particle positions are defined in the continuum of the 3D space and their location changes in time. The fields are defined at precise, immutable locations. In terms of data structures, this means that there is no a priori knowledge of where the particles will be relative to the grid.

A second key aspect of the method is that information regarding these two types of entities needs to be exchanged. Since the particles and the grid are not co-located, interpolations are needed. The sources of the Maxwell equations are defined by

$$\begin{aligned} \rho(\mathbf{x}) &= \sum_p q_p S(\mathbf{x} - \mathbf{x}_p), \\ \mathbf{J}(\mathbf{x}) &= \sum_p q_p \mathbf{v}_p S(\mathbf{x} - \mathbf{x}_p), \end{aligned} \tag{4}$$

where the *shape function* S has unit volume and is typically a "top-hat" S_0 proportional to the indicator function of a mesh cell centered at zero or a smoothed version S_ℓ, defined by $S_\ell := S_0 * S_{\ell-1}$, where $*$ denotes convolution product. The *interpolation function* $W := S * S_0$ is then used to interpolate values via

$$\begin{aligned} \rho_g^n &= \frac{1}{V_g} \sum_p q_p W(\mathbf{x}_g - \mathbf{x}_p^n), \\ \mathbf{J}_g^n &= \frac{1}{V_g} \sum_p q_p \mathbf{v}_p^n W(\mathbf{x}_g - \mathbf{x}_p^n), \end{aligned} \tag{5}$$

where $\mathbf{x}_g$ is the position of each node of the grid and V_g is the volume of each cell of the grid. We assume here uniform grids with grid spacing Δx, Δy and Δz (thereby $V_g = \Delta x \Delta y \Delta z$).

Similarly, the electromagnetic field information is interpolated to the continuum particle location from the discrete nodes of the grid as:

$$\begin{aligned} \mathbf{E}_p &= \sum_g \mathbf{E}_g W(\mathbf{x}_g - \mathbf{x}_p), \\ \mathbf{B}_p &= \sum_g \mathbf{B}_g W(\mathbf{x}_g - \mathbf{x}_p). \end{aligned} \tag{6}$$

In iPic3D we choose $S = S_0$ and thus $W = S_1$. To be explicit,

$S_\ell(\mathbf{x}) = b_\ell(x/\Delta x) b_\ell(y/\Delta y) b_\ell(z/\Delta z)$, where b_ℓ is the B-spline of order ℓ and in particular

$$b_0(x) = \begin{cases} 1 & \text{if } |x| < \frac{1}{2}, \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad b_1(x) = \begin{cases} 1 - |x| & \text{if } |x| < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The evolution equations for the fields and the particles need to be discretized in time. The simplest approach is to solve the two sets of equations in sequence. This is called the explicit approach: the field equations are advanced over one time step Δt using the previous values of the particle properties. Then the particles are advanced by one time step using the fields previously computed. The explicit method is widely popular for its simplicity, allowing an efficient implementation in parallel architectures (e.g. see [7]).

The explicit approach, however, is not viable for space weather simulations. Due to the sequential nature of the approach, each component is temporarily frozen while the other is advancing: the particles stay fixed while the fields evolve, and conversely the particles advance in frozen fields. Needless to say, this is not how nature works. This distortion of the real behavior leads to a limited numerical stability.

In-depth analysis of the explicit scheme [8, 9] shows that the method is stable if three conditions are satisfied. First, the time step must resolve the fastest temporal scale; in space weather problems, this corresponds to the condition $\omega_{pe}\Delta t < 2$, where $\omega_{pe} = (4\pi n_e e^2/m_e)^{1/2}$ is the plasma frequency determined by the electron density, mass and charge. Second, the grid spacing is anchored to the smallest spatial scales. In space weather problems, this is determined by the electron thermal speed and electron plasma frequency as $\Delta x \omega_{pe}/v_{\mathrm{th},e} < \xi$. The parameter ξ is of order unity, but its precise value depends upon the details of the method used. For cloud-in-cell interpolation, its value is approximately 3 [9]. Finally, the explicit solution of Maxwell's equations requires the motion of light waves to be resolved in accordance with a Courant condition: $c\Delta t/\Delta x < 1$. This latter condition is of lesser importance in space weather simulation since the first two are already more limiting.

Let us briefly consider the actual numbers typical of the Earth-space environment. Figure 2 shows typical values of the temporal and spatial scales encountered. The smallest scales are of the order of hundreds of meters and tens of microseconds. Simulating the whole Earth environment (the typical simulation uses a box of 100 Earth radii on a side, about 600,000 km) for hours or days is clearly not feasible using explicit PIC codes. This limitation is not caused by the intrinsic need to resolve those small scales but is caused only by the numerical limitations of the explicit method due to the three stability constraints listed above. The stability constraints of the explicit method are shown in Fig. 2 by the light blue area.

A space weather simulation would be exceedingly accurate, including all the physics processes of interest, if it were to limit itself to much larger scales. In Fig. 2, the smallest scales of actual interest are indicated by the yellow area.

To avoid the prohibitively large number of time steps and the exceedingly high resolution needed to run realistic problems with explicit particle methods, implicit methods must be preferred [10, 11]

2.1. Selected algorithm: Implicit PIC

The present study takes as a starting point the implicit approach deployed in the code iPic3D.

The implicit approach used in iPic3D retains the coupling between the Maxwell and Newton equations. The particles move in a field that is based on an average over the time step, and similarly the fields are advanced using an average position of the particles during the time step. Of course neither is known before

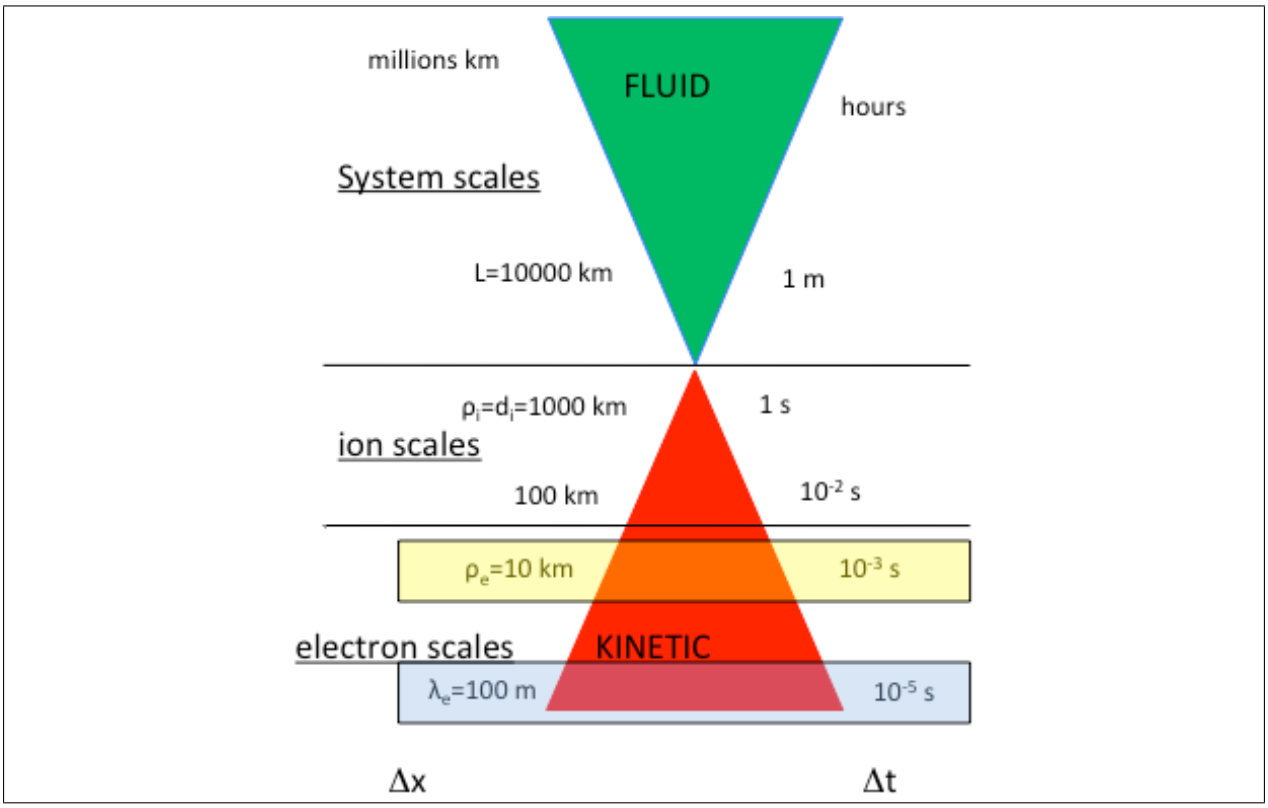


Figure 2: Visual representation of the typical scales observed in the Earth environment (conditions common in the magnetotail). The stability limit of the explicit PIC method forces the use of the resolution at the level indicated by the light blue box (darker box in the black and white version). The desired resolution, instead, is shown by the yellow box (lighter box).

the other is computed, and therefore it becomes impossible to solve one without the other. An iterative procedure becomes necessary.

The starting point of implicit particle methods [12, 13, 14, 15] replaces the explicit time differencing of the equations of motion with an implicit scheme. In our approach, the so-called θ scheme is used [16, 17]:

$$\begin{aligned} \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \mathbf{v}_p^{n+1/2}\Delta t, \\ \mathbf{v}_p^{n+1} &= \mathbf{v}_p^n + \frac{q_s \Delta t}{m_s}\left(\mathbf{E}_p^{n+\theta}(\mathbf{x}_p^{n+1/2}) + \mathbf{v}_p^{n+1/2} \times \mathbf{B}_p^n(\mathbf{x}_p^{n+1/2})\right), \end{aligned} \tag{8}$$

where the flexibility of defining a decentering parameter θ is used to vary the properties of the scheme. The quantities at time level $n+\theta$ are computed as weighted averages of the values at the old and new time level, $\Psi^{n+\theta} = \Psi^n(1-\theta) + \Psi^{n+1}\theta$. Note that the magnetic field does not introduce stability limits (because the magnetic field does no work) and therefore for simplicity can be considered at the old time level n.

The second of the equations (8) can be reformulated more efficiently from a computational point of view by a clever decomposition of the velocity. The velocity equation is rewritten as:

$$\mathbf{v}_p^{n+1/2} = \widehat{\mathbf{v}}_p + \beta_s \widehat{\mathbf{E}}_p^{n+\theta}(\mathbf{x}_p^{n+1/2}), \tag{9}$$

where $\beta_s = q_p \Delta t / 2m_p$ (independent of the particle weight and unique to a given species s, electrons or ions). For convenience, we have introduced hatted quantities obtained by explicit transformation of quantities

known from the previous computational cycle:

$$\begin{aligned} \widehat{\mathbf{v}}_p &= \boldsymbol{\alpha}_p^n \cdot \mathbf{v}_p^n, \\ \widehat{\mathbf{E}}_p^{n+\theta} &= \boldsymbol{\alpha}_p^n \cdot \mathbf{E}_p^{n+\theta}, \end{aligned} \tag{10}$$

The transformation tensor operators $\boldsymbol{\alpha}_p^n$ are defined as:

$$\boldsymbol{\alpha}_p^n = \frac{1}{1+(\beta_s B_p^n)^2}\left(\boldsymbol{I} - \beta_s \boldsymbol{I} \times \mathbf{B}_p^n + \beta_s^2 \mathbf{B}_p^n \mathbf{B}_p^n\right), \tag{11}$$

and represent a scaling and rotation of the velocity vector.

Equations (8) are implicit, because the new fields are needed before the equations for the particles can be solved. If the electric and magnetic fields were given, the equations of motion could be solved independently. Note that even this task is not trivial, because to solve for the particle velocity one needs to compute the electric and magnetic field at the particle position. Moreover, even if the value of the fields at the advanced time were given, the equations for particle velocity and position would still form a coupled set. Given that a B-spline interpolation is used, the equations are nonlinear, and the task of solving them is not trivial. Nevertheless, each particle can be treated as a system of 6 coupled nonlinear equations to be solved with any of the iterative schemes widely used. Newton's method can be used [18], but in iPic3D this is accomplished by a simpler fixed number of iterations of the predictor-corrector scheme [6]. Three iterations are used in the present work. The real challenge of the implicit method is that the electric and magnetic fields are not given. Unlike the explicit method, the particle equations of motion need time-advanced fields that are not known yet. The implicit method reintroduces the coupling of the particle equations with the field equations. Implicit methods solve Maxwell's equations also implicitly in time. Indicating with index g the spatial discretisation of the operators, the discretised Maxwell's equations can be written as:

$$\begin{aligned} \mathbf{B}_g^{n+1} - \mathbf{B}_g^n &= -\Delta t \nabla \times \mathbf{E}_g^{n+\theta}, \\ \mathbf{E}_g^{n+1} - \mathbf{E}_g^n &= \frac{\Delta t}{\mu_0 \epsilon_0}\left(\nabla \times \mathbf{B}_g^{n+\theta} - \mu_0 \mathbf{J}_g^{n+\frac{1}{2}}\right), \\ \epsilon_0 \nabla \cdot \mathbf{E}_g^{n+\theta} &= \rho_g^{n+\theta}, \\ \nabla \cdot \mathbf{B}_g^{n+1} &= 0. \end{aligned} \tag{12}$$

Many spatial discretization schemes can be used, but their detail is beyond the scope of the present discussion. More details on the finite volume method used by iPic3D are presented elsewhere [6]. The point of crucial interest here is the temporal discretization.

Two issues need to be considered to reach an effective implementation of the implicit method: the divergence conditions and the coupling between the Maxwell and Newton equations.

2.2. The divergence conditions

There are in fact eight equations for only six unknowns. The electric field is determined by the three curl equations and also by the divergence equation, Gauss's law). Similarly, the magnetic field is governed by three curl equations plus the zero-divergence condition. The latter is of no concern in PIC methods because the numerical discretization used automatically satisfies it: the magnetic field is expressed as the curl of another vector, thereby imposing its zero divergence. However, Gauss's law requires careful consideration.

In the continuum, before a spatial discretization is introduced, the theory of electromagnetism ensures that any system governed by the two sets of curl equations and satisfying the two divergence conditions at the initial time will automatically preserve them at all times, provided that charge is conserved in the system, meaning that

$$\frac{\partial \rho}{\partial t} = \nabla \cdot \mathbf{J}. \tag{13}$$

The divergence conditions become then just a constraint on the initial state.

In a discretized system, however, the charge conservation equation is not satisfied. Of course, the total charge in the system is on average maintained, as no charge is created or destroyed by the discretization, but the detailed balance in each cell implied by Eq. (13) may not be satisfied. Two types of approaches are commonly followed to overcome this difficulty.

The so-called charge conserving schemes can be used. The interpolation of charge and density is chosen in a way to conserve charge locally, ensuring that the discretised version of Eq. (13) is valid [9]. The approach has two serious drawbacks: to enforce this property, one is forced to sacrifice the conservation of momentum (see below) and, reportedly, the simulations display an increased level of noise, requiring many more particles per cell (see Ref. [9] at page 360 and Ref. [19]). The advantage is that no elliptic solver is needed, as both divergence equations are automatically enforced. This feature leads to excellent parallel scaling when implemented on massively parallel computers [7].

Alternatively, divergence cleaning approaches can be used by simply correcting the electric field for its electrostatic part to enforce Poisson's equation:

$$\nabla^2 \delta\varphi = \nabla \cdot \mathbf{E} - \rho/\epsilon_0, \tag{14}$$

where the corrected field is $\mathbf{E}' = \mathbf{E} - \nabla\delta\varphi$. Clearly, this achieves the goal by adding a correction, but at the cost of solving an elliptic problem, a CPU-intensive operation. To circumvent this difficulty, inexact versions have been proposed [20, 21] where the elliptic equation is not solved to convergence but is in fact in a sense converged over multiple time steps. The computational burden is reduced at the cost of allowing a controlled but non-zero error in the divergence equation. Also, this method can be made efficient by choosing scalable modern solvers [22, 23].

In iPic3D, fortunately, the situation is simplified by the choice of discretization that ensures that any divergence error is damped in time and eliminated automatically [24]. For this reason, it is typically not necessary to apply either of the two techniques.

2.3. The Implicit Moment Method - IMM

The second challenge is handling the coupling between the Maxwell and Newton equations. In iPic3D, this task is accomplished with the IMM. The algorithm of implicit PIC requires dealing with the coupled equations (12) and (8). The first step is to consider the true nature of the coupling between the two sets. The time-discretized Maxwell equations can be conveniently rewritten in their second-order form by taking the curl of the first equation in system (12), solving for $\nabla \times \mathbf{B}_g^{n+1}$, and substituting in the second equation to eliminate $\mathbf{B}_g^{n+\theta} := \theta\mathbf{B}_g^{n+1} + (1-\theta)\mathbf{B}_g^n$ [24]:

$$\mathbf{E}^{n+\theta} = \mathbf{E}^n + c^2\theta\Delta t\left(\nabla \times \mathbf{B}^n - \theta\Delta t \nabla \times \nabla \times \mathbf{E}^{n+\theta} - \mu_0 \mathbf{J}^{n+1/2}\right). \tag{15}$$

The present discussion focuses on the temporal discretization, which is the defining aspect of the IMM.

For stability of the numerical scheme [24], the double curl is best expressed using vector identities and Gauss's law as:

$$\nabla \times \nabla \times \mathbf{E}^{n+\theta} = \frac{1}{\epsilon_0}\nabla\rho^{n+\theta} - \nabla^2\mathbf{E}^{n+\theta}. \tag{16}$$

With this choice, the system for the field equations for the implicit PIC method becomes:

$$\mathbf{E}^{n+\theta} - (c\theta\Delta t)^2\nabla^2\mathbf{E}^{n+\theta} = \mathbf{E}^n + c^2\theta\Delta t\Big(\nabla \times \mathbf{B}^n - \frac{\theta\Delta t}{\epsilon_0}\nabla\rho^{n+\theta} - \mu_0\mathbf{J}^{n+1/2}\Big), \tag{17}$$

where it is now most evident that the new value of the electric field requires two sources: the density and the current computed at intermediate levels during the time step. These values are not available from the previous time step and require an iteration with the equations of motion. The current and densities are in fact defined by Eq. (5), which in turn requires to know the position and velocity of all particles at the advanced time level t^{n+1}.

Two approaches are possible: a full non-linear iterative solver of the coupled particle and field equations or a semi-implicit solver that linearizes this iteration procedure. Recently, the first approach finally was attempted as reported in Section 3 below, but the current production version of iPic3D uses the latter, in the specific implementation called the IMM [14].

The idea of the IMM is that the two sources in the field equation (17), $\rho^{n+\theta}$ and $\mathbf{J}^{n+1/2}$, are provided by an average over all the particles in accordance with Eq. (4). Therefore, an approximation of these averages can be sufficient without having to move all the particles self-consistently. In practice, an approximation procedure can be followed where the expressions in Eq. (4) are expanded in Taylor series $S(\mathbf{x}-\mathbf{x}^{n+\theta}) = S(\mathbf{x}-\mathbf{x}^n) - (\theta\Delta t)\mathbf{v}^{n+\theta}\cdot\nabla S(\mathbf{x}-\mathbf{x}^n)+\cdots$ to obtain approximations in terms of initial quantities

$$\begin{aligned}\mathbf{J}^{n+1/2} &\approx \widehat{\mathbf{J}}^n + \frac{\boldsymbol{\mu}^n}{\theta\Delta t}\cdot\mathbf{E}^{n+\theta},\\ \rho^{n+\theta} &\approx \rho^n - \theta\Delta t\nabla\cdot\mathbf{J}^{n+1/2}\end{aligned} \tag{18}$$

that can then be substituted in Eq. (17). In accordance with Eq. (9), hatted quantities approximate average values that would result in the absence of the electric field:

$$\begin{aligned}\widehat{\mathbf{J}}^n(\mathbf{x}) &:= \sum_p q_p\widehat{\mathbf{v}}_p^n S(\mathbf{x}-\mathbf{x}_p^n) - \frac{\Delta t}{2}\nabla\cdot\sum_p q_p\widehat{\mathbf{v}}_p^n\widehat{\mathbf{v}}_p^n S(\mathbf{x}-\mathbf{x}_p^n),\\ \widehat{\rho}^n(\mathbf{x}) &:= \sum_p q_p S(\mathbf{x}-\mathbf{x}_p^n) - \theta\Delta t\nabla\cdot\widehat{\mathbf{J}}^n(\mathbf{x}).\end{aligned} \tag{19}$$

The response of the plasma to the changing electric field is summarized in the plasma response tensor:

$$\boldsymbol{\mu}^n = \sum_s \frac{q_s\rho_s^n}{m_s}\frac{\left(\boldsymbol{I}-\beta_s\boldsymbol{I}\times\mathbf{B}^n+\beta_s^2\mathbf{B}^n\mathbf{B}^n\right)}{1+(\beta_s B^n)^2}. \tag{20}$$

Assuming that spatial gradients of the magnetic field are small, Eq. (19) yields an approximation that gives implicit moments per species in terms of initial moments:

$$\begin{aligned}\widehat{\mathbf{J}}_s^n(\mathbf{x}) &\approx \boldsymbol{\alpha}_s^n(\mathbf{x})\cdot\left(\mathbf{J}_s^n(\mathbf{x}) - \frac{\Delta t}{2}\nabla\cdot\sum_{p\in s} q_s\mathbf{v}_p\mathbf{v}_p S(\mathbf{x}-\mathbf{x}_p^n)\right),\\ \widehat{\mathbf{J}}^n(\mathbf{x}) &:= \sum_s \widehat{\mathbf{J}}_s^n(\mathbf{x}).\end{aligned}$$

This approximation is used in iPic3D.

Substituting in Eq. (17), the final form of the field solver for the IMM follows as

$$\begin{aligned}&(c\theta\Delta t)^2\left[-\nabla^2\mathbf{E}^{n+\theta} - \nabla\nabla\cdot\left(\boldsymbol{\mu}^n\cdot\mathbf{E}^{n+\theta}\right)\right] + \boldsymbol{\epsilon}^n\cdot\mathbf{E}^{n+\theta}\\ &= \mathbf{E}^n + (c\theta\Delta t)\left(\nabla\times\mathbf{B}^n - \mu_0\hat{\mathbf{J}}^n\right) - \frac{(c\theta\Delta t)^2}{\epsilon_0}\nabla\hat{\rho}^n,\end{aligned} \tag{21}$$

where $\boldsymbol{\epsilon}^n = \mathbf{I}+\boldsymbol{\mu}^n$ and $\mathbf{I}$ is the identity tensor.

The equation (21) derived above is the field solver of the IMM; from it the new electric field can be computed, and consequently, from the first of Eq. (12), the new magnetic field can be solved. And this without yet needing to move the particles. The information about the particle response to the changing fields during the time step is summarized by Eq. (20), obtained with a linearization procedure.

The computational advantage is great when compared with the fully implicit method, because the field solver needs only a linear iteration without involving the particles other than to compute the auxiliary quantities $\widehat{\rho}^n$ and $\widehat{\mathbf{J}}^n$ once at the beginning of the time step. Compared with the fully implicit method, the semi-implicit method does not require any iteration between particles and fields and does not require any nonlinear solver. In iPic3D, Eq. (21) is solved with GMRES [6]. Of course, as discussed in Section 4, this

simplification comes at a cost: the IMM does not conserve energy exactly, while the fully implicit method does.

Provided that the condition that the particles on average do not travel more than one time cell per time step is satisfied, however, the semi-implicit method provides accurate results [14]. For a typical maxwellian plasma, this accuracy condition can be expressed as:

$$v_{\mathrm{th},e}\Delta t/\Delta x < 1. \tag{22}$$

Returning to the typical scales of space weather, reported in Fig. 2, this condition can be satisfied by arbitrarily large values of Δt and Δx, provided their ratio is constrained by Eq. (22). In practice, the constraint means that the temporal and spatial resolutions can both be increased in the hourglass representation of Fig. 2, as long as the yellow bar is kept horizontal. Δt and Δx can be increased together by the same factor from the lowest values imposed by the explicit scheme (in light blue) to the desired higher value (in light yellow). The resolution bar must remain horizontal in semi-implicit methods, but this is exactly the position it is needed to have for space weather problems, causing no practical limitation to the user. Fully implicit methods do not have this limitation, and Δt and Δx can be chosen independently, a property that can be of value to other applications beyond space weather [25].

The typical computational cycle of the IMM implemented in iPic3D is reported in Fig. 3.

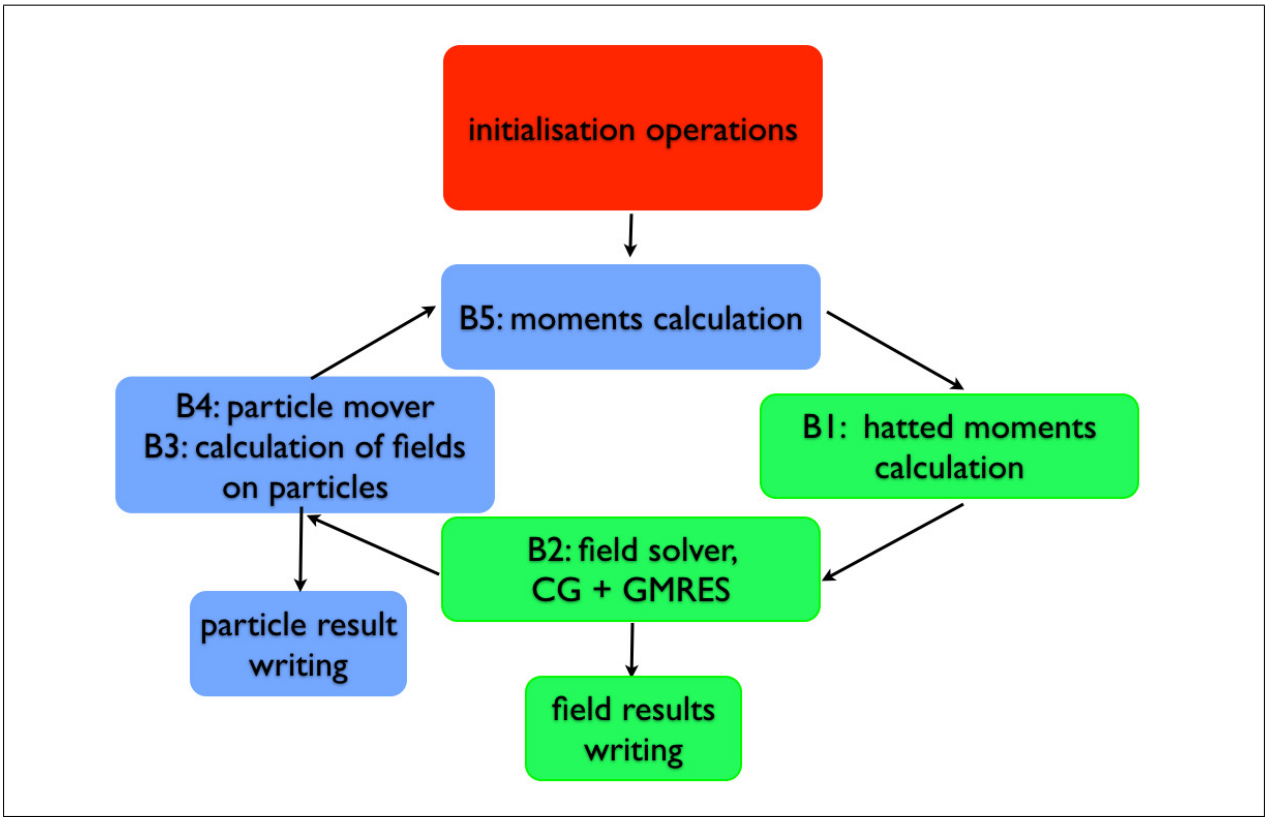


Figure 3: Schematic representation of an IMM PIC code. After initialization (red block, darkest color in the black and white version), the different logical blocks described in Section 2.1 are executed in a cycle. The blocks depicted in blue (darker color) and green (lightest color) are marked as "particle" and "grid" blocks respectively according to whether the number of operations in the block scales with the number of particles N_p or with the number of grid points N_g.

2.4. Algorithm deployment on distributed memory computers: iPic3D

The IMM PIC method has been implemented in a parallel software package, called iPic3D, written entirely in C++ [6]. An Object-Oriented (OO) design has been followed in writing iPic3D using the so-called lite OO approach presented in a previous work [26]. The variables related to particles are organized as arrays in Particle objects and divided depending on the species (electrons, ions, ...). The electromagnetic field constitutes a whole object that consists of the electromagnetic field and field sources variables. OO paradigms, such as class inheritance and polymorphism, are used to make it easy for developers to add new code to iPic3D without modifying the existing code.

It is crucial that both field-solving and particle-moving are parallelized efficiently. On distributed memory systems, the domain decomposition technique is used to divide the computational workload among the computing units. The simulation box is divided among processors using a generic cartesian virtual topology. Particles are distributed among processors based on their location and communicated to adjacent processors if exiting from the processor domain. In fact, an important aspect of efficiency is the need to retain the particles and cells belonging to a subdomain on the same processor. Large amounts of information are exchanged between grid and particles residing in the same physical domain because of the interpolation step, and therefore it is important to avoid that this information exchange results in inter-processor communication.

The parallelization of the code is based on MPI libraries and on blocking `MPI_Send`/`MPI_Receive` calls. Parallel communication is involved in all the stages of the IMM PIC code. At each mover iteration, the data for particles leaving the domain is collected in temporary arrays. Once the particle data is updated, the temporary arrays are exchanged among contiguous domains. Ghost cell data is communicated during each solver iteration and during the calculation of the densities in the interpolation from particles to the grid. The use of non-blocking MPI subroutines in combination with the OmpSs programming model is under development within the DEEP project.

2.5. Performance analysis of iPic3D

The very good scaling properties of iPic3D on as many as 16384 CPUs have already been demonstrated in, for example, [27]. In this section, following [28], the focus is on a more detailed analysis of the different logical blocks of Fig. 3, which are repeated every computational cycle. It is reminded that the logical blocks under investigation are the following: B5 (moments calculation), B1 (hatted moment calculation), B2 (field solver), and B4 (particle mover). The logical block B3 (calculation of fields on particles) is included in B4. A deep understanding of the characteristics of each of them is fundamental to address issues of co-design and of code porting to different architectures.

Figure 4 shows the percentage execution time (panel a) and the percentage bytes transferred (panel b) for the different logical blocks. The measures are obtained with Scalasca, an open-source toolset for scalable performance analysis of large-scale parallel applications [29]. Manual instrumentation is inserted in the source code around the entry points of the different logical blocks to reduce the overhead in the analysis coming from automatic instrumentation. The Scalasca overhead, provided as a metric, is bound to reasonable values. This confirms that the significance of the measures is not hindered.

Different test cases are examined in Fig. 4. The aim is to check if the behavior registered is dependent on the problem size, the number of processors used and the hardware used.

All cases are 3D simulation of magnetic reconnection [30]. The test cases labeled as "judge" refer to quite small simulations performed on the cluster JUDGE [31], an IBM iDataPlex cluster hosted at Forschungszentrum Jülich. The test case "juqueen" is a rather bigger simulation performed on JUQUEEN [32], an IBM BlueGene/Q cluster and currently Europe's fastest computer according to the most recent TOP500 list at the time of first submission [33]. The information on the number of MPI processes used in each direction follows the name of the cluster. For example, "judge - 4x2x2" refers to the test case performed on judge, with 4, 2 and 2 processes used in the x, y and z direction respectively. Hence, 16 processes are used in total. The input parameters for the two test cases are listed in Table 1, second and third column. The number of processors used in the three simulated directions is listed in the legend of Fig. 4 for the different test cases. The number of processes equals the number of cores.

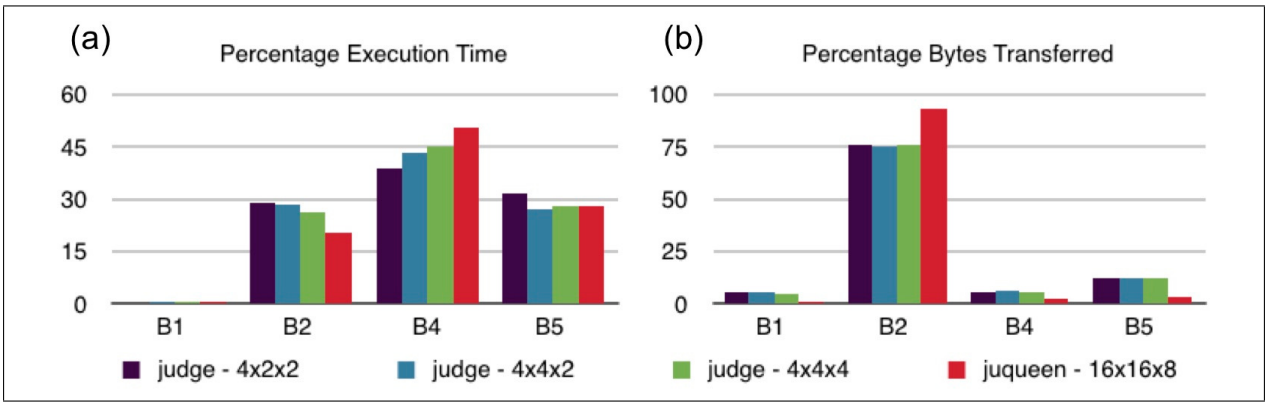


Figure 4: Scalasca measurements of magnetic reconnection simulations performed with different input parameters and with a varying number of processors (i.e. cores) on the clusters JUDGE and JUQUEEN. Panel (a) shows the percentage execution time and panel (b) the percentage bytes transferred for the four main logical blocks which constitute iPic3D.

Table 1: Input parameters for the "judge" and "juqueen" test cases of Fig. 4. n_x, n_y and n_z are the cell number in the three directions, n_{ppc} the number of computational particles per cell, N_s the number of particle species (electrons, ions, background electrons and background ions), $L_x/d_i \times L_y/d_i \times L_z/d_i$ the physical dimensions of the domain in the three directions normalized to the ion skin depth d_i.

	JUDGE	JUQUEEN
$n_x \times n_y \times n_z$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
n_{ppc}	27	27
N_s	4	4
$L_x/d_i \times L_y/d_i \times L_z/d_i$	$10 \times 10 \times 10$	$20 \times 20 \times 20$

Figure 4 shows that the results obtained both for the percentage execution time and the percentage bytes transferred are rather independent of the problem size and of the number of processors used. The block B4 (particle mover) is consistently the most time-consuming, due to the need to iterate over all particles and perform a computationally expensive implicit solve on each particle.

Another very time-consuming block is B5 (moments calculation), which cycles over all particles to extract particle moments (density, currents, pressures) to deposit on the grid. Note that the percentage execution times of these two blocks increases with the number of particles per cell used in the simulation; the value used here, 27, the same as in the sample application of Section 5.1 but is still rather low when compared to the number of particles often employed in production simulations, typically $n_{ppc} \approx 100$. Using a higher number of particles helps to control numerical noise, but of course increases the computational cost of the simulation. Blocks B4 and B5 can be collectively defined as "particle blocks," since they are dominated by computation proportional to the number of particles. Blocks B1 (hatted moments calculation) and B2 (solver), instead, operate on vectors sized according to the computational grids and are therefore "grid blocks."

Fig. 4 (b) shows that block B2 (solver) is always responsible for most of the communication. In fact, the block solves Equation (21) and (14) with two iterative solvers, the Generalized Minimal RESidual method (GMRES) and the Conjugate Gradient (CG) [34]. At each iteration of the two solvers, Peer-to-peer (P2P) communication is performed to update the values of the grid ghost nodes of each process, and global communication is done to calculate vector moduli and scalar products.

The only other block performing global communication in iPic3D is B4 (mover), as part of a routine which communicates particles residing in a process to another one: if at the end of the particle move the center of a particle residing in one process is located in the grid area mapped to another process, the particle

is moved. However, most of the global communication is done in B2. This is confirmed by Fig. 5, which shows how much of the bytes transferred on the network during the test cases described are part of global communications (blue bars) and how much of these global communications are initiated in B2 (red bars). The measures are again done with Scalasca. Observe that the percentage of bytes on the network transferred with collective communication significantly increases, as it can be expected, when the number of processors increases. It is also consistently evident that B2 is primarily responsible for global communication.

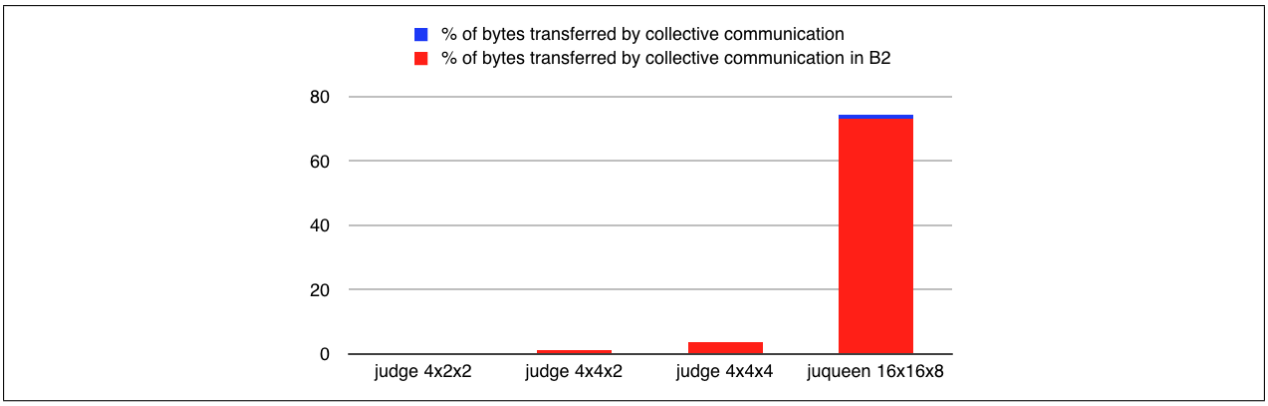


Figure 5: Scalasca measurements of the communication footprints of the test case simulations described before. The blue (darker color in the black and white version) bars are the percentage of the total bytes transferred which are part of global communication. The red (lighter color) bars show how much of this global communication is initiated within B2.

3. Algorithmic developments

3.1. Fully implicit PIC

The fully implicit PIC method requires the concurrent solution of the equation of motion for each particle and of the field equations for the electric and magnetic fields at each grid point. The solution of the implicit numerical equations of PIC method implies the computation of a nonlinear system. Nonlinearity arises from the coupling between particles and field variables through the interpolation functions of the PIC method. In the fully implicit PIC scheme, the nonlinear equations are solved by a Newton-Krylov solver [35]. Despite the belief that the iterative solution of such equations could hardly converge, it has been proven that such PIC methods are convergent [36]. In addition, the fully implicit PIC scheme requires the solution of a very large matrix whose rank is of the order of the number of particles. The number of particles is considerably higher than the number of grid points in typical PIC simulations. For this reason, implementations of fully implicit PIC methods are based on matrix-free Jacobian-free solvers to avoid the storage of the matrix and Jacobian coefficients [36].

A fully implicit method can be formulated so that the PIC method conserves the total energy of the system exactly [37, 38, 25]. This quantity is not conserved in most PIC formulations. Energy conservation is of crucial importance when acceleration mechanisms need to be described correctly. Compared with the IMM, the fully implicit PIC has a much simpler formulation and results in an easier implementation.

The main disadvantage of the fully implicit PIC method is that it requires the solution of a very large system. The size of the system increases with the number of particles. The number of particles is easily more than a million in a typical PIC simulation, leading to a matrix to be inverted whose rank size is of the order of million. To reduce the size of this matrix, it is possible to use a technique called *kinetic enslavement* [37, 38, 25]. In this method, the only unknowns of the problem are the value of the electric magnetic field at the grid points at the new time level, and the Jacobian-Free Newton-Krylov (JFNK) solver computes only the field equations. The particle equations of motion are calculated by an ODE Newton-Raphson method and are embedded in the field solver as function evaluations.

3.2. Multi Level Multi Domain (MLMD)

The aim of the IMM Multi Level Multi Domain (MLMD) algorithm is to reduce the computational costs of simulations algorithmically, rather than by relying on more powerful or innovative hardware. The algorithm is of particular relevance in the field of space weather, since, as recalled in Section 2 and Fig. 2, the problem is intrinsically multi-scale. If different scales are of relevance in different sub-domains of the simulation, notable computational resources can be saved if the different subdomains are resolved with the *local* resolution of interest, rather than with the highest resolution needed in the domain. Examples of such *adaptive* PIC codes are Adaptive Mesh Refinement (AMR) codes [39, 40] and the already cited MLMD technique [41, 42, 43, 44, 45].

The MLMD represents the domain as a collection of subdomains resolved with increasing resolution. If smaller spatial scales are of interest in a subsection of the grid, that area is resolved both with the initial resolution and with increased resolution as a new level of the MLMD system. The two levels are labeled as coarse and refined respectively. All levels are self-similar; that is, they are fully simulated in fields and particles. Contrary to standard AMR implementations, coarse-level particles are present in the coarse grid area even where it is also resolved with a refined grid and refined particles. This is done for two reasons. First, under the numerical point of view, this technique allows to avoid a number of issues related to the change of the shape function of computational particles which cross boundaries between resolution levels (refer to [41] for a discussion). Second, the self-similarity between the levels results in an easier transition from a standard, single-level IMM PIC code to its MLMD version. Since the same particle and grid operations are performed at each level, the levels can be represented as instances of the same class. The difference lies, of course, in the physical dimension of the subdomains and in the resolutions used.

Fig. 6 shows how the MLMD method modifies the sequence of logical operation of Fig. 3, which refers to a single-level PIC IMM code. For simplicity, from now on just one coarse (level ℓ) and one refined (level $\ell + 1$) grids are assumed.

The operations underlined need to be performed additionally in a MLMD system:

- MLMD initialization: most of the MLMD initialisation operations consist in the construction of the "communication maps" between the levels. Each process assigned to the refined level calculates from the input data the rank of the coarse level processes it interacts with for the MLMD operations described below. This information is then broadcast to all coarse-level processes. Each coarse-level process involved in MLMD operations then receives extra information (e.g.: the number of grid points to send/receive) through P2P communication.

  Notice that the same number of processes is assigned to the coarse and the refined grid and that a fraction $L_x/RF \times L_y/RF$ of the coarse grid is simulated with increased resolution and the same number of grid points and computational particles as the coarse grid. L_x and L_y are the dimensions of the domain in the x and y directions. RF is the Refinement Factor between the grids, i.e. the jump in spatial resolution. Thus, the number of coarse grid processes interacting with the refined grid becomes smaller and smaller with increasing RFs. Conversely, most of MLMD operations are performed only by the processes located at the boundary of the refined grid. This is a source of computational imbalance between the processes.

- Boundary Condition (BC) interpolation: Before the refined grid starts the computation of block B2 (field solver), boundary conditions for the electric and magnetic field at time $n + 1$ must be received from the coarse grid. This introduces a bottleneck in the execution, since the refined grid has to wait idle for the coarse grid solution to be computed before starting its own block B2.

  Notice however, from Section 2.5, that block B2 is a grid-related block, that is not one of the most time consuming ones. Additionally, coping strategies (e.g.: using as BC for the refined grid an intermediate solution of the iterative GMRES solver, rather than the fully converged result) can be devised to reduce the waiting time of the refined grid.

- Field Projection: After completion of block B2 on the refined grid, the increased-resolution solution is projected to the coarse grid. This introduces a second bottleneck, since this time the coarse grid has to wait for the refined grid solution.

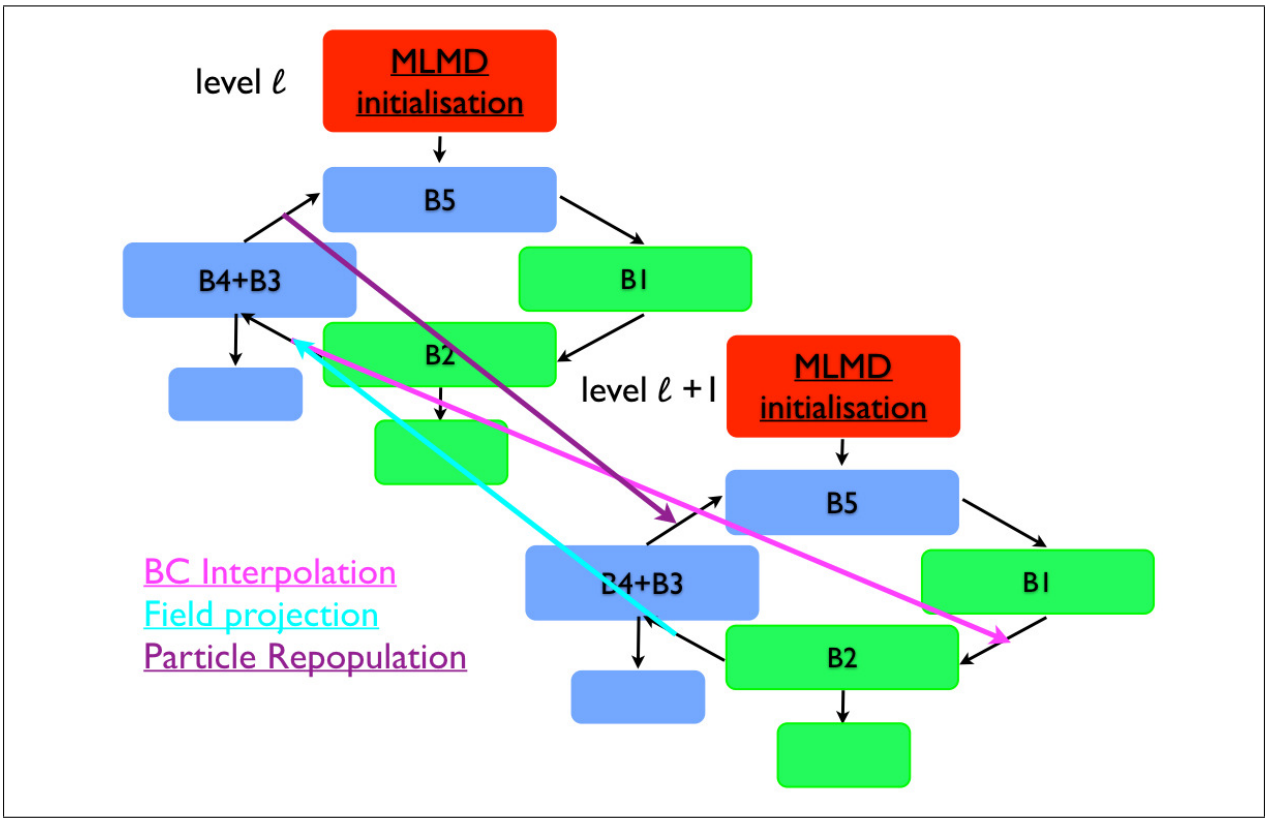


Figure 6: Communication operations between the levels simulated with different resolution in a MLMD system. Level ℓ is the coarse grid, level $\ell + 1$ the refined grid. The extra operations performed in a MLMD system with respect to a single-level system (Fig. 3) are underlined. The endpoints of the arrows give the direction of the communication, from the coarse to the refined grid or vice versa.

- Particle Repopulation: To provide optimal particle boundary conditions for the refined grid, the native refined grid particles sitting in a small area around the grid boundary, indicated as Particle Repopulation Area (PRA), are deleted after the last iteration of the particle mover. New refined particles are created from the corresponding coarse-grid particles with the splitting algorithm described in [46]. This operation is performed after both grids have completed block B4 (particle mover).

  The operation adds a minimum overhead on the coarse-grid processes overlapping the PRA area, which store on the fly, while moving particles, the particle information to be sent to the refined grid. Higher overhead befalls the refined grid processes, which do the particle splitting and communicate the split particles to the appropriate refined process, according to the position of the center of the new particle.

Notwithstanding the node imbalance and bottleneck issues just underlined, the performances of the MLMD code are very promising due to two facts. First, the IMM method is used as a baseline algorithm, while all AMR codes in use rely on explicit PIC algorithms. This means that the MLMD implementation proposed benefits from the laxer stability constraints of implicit methods, described in Section 2. To that, the benefits coming from the MLMD method itself have to be added.
The IMM allows to have very high RF jumps between the levels, while still being in the stability range of the method. Such high refinement jumps are also possible due to the particle repopulation method used. With other particle boundary conditions, the RF jumps between the levels are limited also by the necessity to keep the errors in the update of the particle positions and velocities at the boundaries between the levels bounded to reasonable values (see [41] for a discussion). Such necessity is removed with the splitting algorithm used.
Fig. 7, from [42], shows performance measures. The two lines refer to the execution times in seconds

Table 2: Execution time in seconds for two grid MLMD Maxwellian plasma simulations with varying RF. From [42].

RF	Exec Time [s]
2	41.25
6	43.98
10	45.87
14	46.97

for the simulation of the same problem, a Maxwellian plasma relaxation over a relatively small domain, $L_x = L_y = 21.0\ d_e$, where d_e is the electron skin depth [47]. All simulations are performed using 128 processes for the same number of cycles. The red line depicts the execution times for a two level MLMD system with increasing RF between the two grids. The blue line shows the execution times for the simulation of the entire domain with the resolution of the refined grid of the corresponding MLMD case.

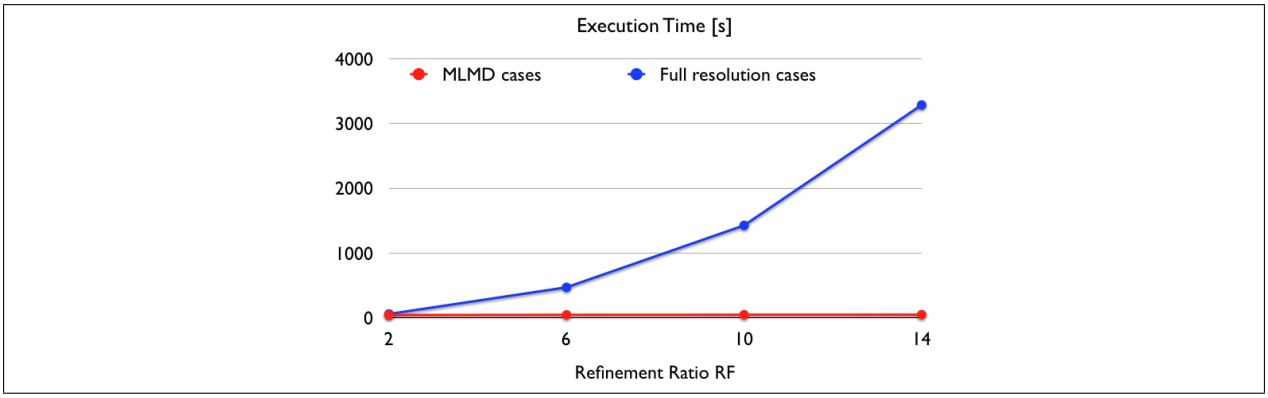


Figure 7: Comparison of the execution times of two-level MLMD simulations (red line, lighter color in the black and white version) and one-level simulations (blue line, darker color) performed with the same resolution of the MLMD refined grid as a function of the Refinement Factor RF. From [42].

Notice that the saving in computing time achieved with the MLMD method is remarkable, if a fraction $L_x/RF \times L_y/RF$ of the total domain needs to be simulated with a resolution per side RF times higher than the coarse grid. In the case with $RF = 14$, the execution times in the two cases are 46.97 versus 3287.46 seconds! $RF = 14$ is chosen as final value in the series not due to a performance breakdown, but because higher factors would violate the IMM accuracy constraint on the refined grid. The execution times of the MLMD simulations with increasing RFs can be compared in Table 2.

The timings are remarkably similar, because the same number of computational particles is present in all the simulations, notwithstanding the RF. The slight increase with the RF is traceable to the increasing cost of the MLMD operations with increasing refinement. In particular, the particle repopulation operations are expected to scale with RF^2. However, this overhead is bounded to very reasonable values and does not constitute a major issue at this stage of the code development, when performance issues have not been addressed yet.

3.3. Fluid-Kinetic PIC

A new fluid-kinetic method that solves concurrently the multi-fluid and Maxwell's equations has been recently developed [48]. The governing equations for the fluid-kinetic PIC solver are the multi-fluid equations, Maxwell's equations and particle equations of motion. The multi-fluid equations and Maxwell's equations are solved on the grid, while the equation of motion are solved for each computational particle.

The fluid continuity and momentum equations for the species s are:

$$\begin{cases} \partial\rho_s/\partial t + \nabla\cdot\mathbf{J}_s = 0, \\ \partial\mathbf{J}_s/\partial t = (q/m)_s(\rho_s\mathbf{E} + (\mathbf{J}_s\times\mathbf{B})/c - \nabla\mathcal{T}_s); \end{cases} \tag{23}$$

$\mathcal{T}_s = \mathbf{J}_s\mathbf{J}_s/\rho_s + p_s$ and p_s are respectively the stress and pressure tensors. Different closure equations can be provided to calculate the stress tensor $\mathcal{T}_s$ in Eq. 23. In the fluid-kinetic method, we intend to retain the kinetic effects by calculating the stress tensor using the computational particles of the PIC method by interpolation. Therefore, the closure equation for the stress tensor is provided by particles as:

$$(\mathcal{T}_s)_g = \sum_p^{N_p} q_s\mathbf{v}_p\mathbf{v}_p W(\mathbf{x}_g - \mathbf{x}_p). \tag{24}$$

The fluid-kinetic PIC solver includes both fluid and kinetic models in the same framework, making the fluid-kinetic PIC solver an ideal candidate for coupling fluid-kinetic models. It shows that fluid-Maxwell solvers for plasma simulations that neglect kinetic effects can be equipped with computational particles to introduce kinetic effects in the simulation. Like other implicit PIC methods, the fluid-kinetic PIC solver allows to use large time steps, thus enabling kinetic simulations over time scales typical of the fluid simulations. Finally, it simplifies the formulation of IMM PIC models, making the implementation of such schemes less prone to errors.

4. Adapting iPic3D to GPU and Xeon Phi architectures

Co-design aims at establishing continuous and effective feedback between the code development and machine design stages. On one side, algorithms must be devised as to exploit at best the characteristics of exascale-oriented machines: vectorisation and parallelisation must be extreme, communication and synchronisation must be reduced. On the other end, the requirements coming from cutting-edge scientific fields, such as space weather, have to be considered in the design of next generation HPC machines.
We here briefly recall the requirements that codes must satisfy in order to efficiently exploit architectures such as Graphical Processing Units - GPUs [49] and Xeon Phi's [50]: vectorisation, massive parallelisation, smart handling of communication and reduction of synchronisation. We then highlight directions of future developments for iPic3D, so that these requirements could be better met in the future. Finally, we report our experience and future work within the DEEP project, which aims at building a cluster of Xeon nodes (the "Cluster") and Xeon Phi coprocessors (the "Booster"). The final objective of the DEEP project is to enable, though this innovative architecture, exascale performance for codes composed of parts which exhibit different scalability properties.

4.1. Requirements for efficient exploitation of GPUs and Xeon Phi's architectures

Vectorization promises to accelerate the compute-intensive parallelizable parts of our algorithm.
Xeon Phi and GPU are two competing technologies that offer alternative mechanisms for vectorization. Both Xeon Phi and GPU offer a platform involving at least one coprocessor (or "accelerator") attached to a host processor. Efficient use of the coprocessor for *massively parallel processing* requires respecting constraints on the organization and movement of data.

A Xeon Phi coprocessor has a large number of cores (e.g. 50 or 60) each of which supports the full x86 instruction set and each of which has a vector unit capable of operating on eight double-precision numbers.

The GPU allows to create a large team of lightweight threads that operate on parallel data and that all simultaneously execute the same stream of instructions taken from a limited instruction set. The smallest unit in which these teams of threads are organized is a *warp*, a team of typically 32 GPU "threads" executing identical instructions. Memory access is efficient on the GPU if the data accessed by each warp is aligned and in sequence as if for a conventional vector processor with 32 elements in each vector. In contrast to vector units, such perfect coalescence of memory is not a requirement of the GPU, but the time required

to access memory can increase by as much as a factor of 32 if restrictions on localization, alignment, and sequence of data are violated [51]; more advanced GPU architectures are less restrictive on the alignment and sequencing required for fast access [52, 53].

On massively parallel architectures such as GPU clusters (a cluster of CPU hosts each with one (or more) GPU(s) connected to it over the PCI Express), the available bandwidth is still low compared to the computing capabilities of the GPU. It is not surprising then that one of the main limiting factors for multi-GPU architectures is the latency of transfer between CPU and GPU memory [54].

On GPUs, current state-of-the-art MPI libraries can only access host memory, implying that if data needs to be transferred between GPUs on different nodes then one needs first to transfer it explicitly from the device to the host. There has been much effort to overcome this problem by explicitly pipelining the data movement and kernel execution at the application level [55, 56, 57, 58, 59, 60]. Since collective communication routines are internal to MPI, there is only so much one can do, and ultimately efficient collective communication will require MPI implementations that can access directly the device memory. Two such examples that provide the capability to communicate data in between GPUs through a distributed-memory parallel architecture and using an MPI-like message passing interface are cudaMPI and glMPI [61]. Note, these libraries only circumvent the two-step communication for the user by providing one single function taking care of both steps. They do not eliminate the necessity to physically pass through the CPU host(s). Collective communication is less of an issue on Xeon Phi, on the other hand, as the coprocessor runs independently of the host processor and can issue its own MPI calls.

4.2. Future directions for the development of iPic3D to satisfy GPU / Xeon Phi requirements

In this Section, we highlight how the different logical blocks of iPic3D can be adapted and modified to better exploit GPU and Xeon Phi architectures. These modifications will be implemented as future work.

All parts of iPic3D have the potential to benefit from parallelization.

The part that has the greatest potential for vectorization is moving particles (block B4), since each particle moves independently of the others. The main potential bottleneck to computation is that calculating particle movement requires random-access interpolation of field data at the location of each particle. This means that particles should be sorted by location into "neighborhoods" so that the field data for each neighborhood fits in cache. Each particle needs to be transferred if necessary to the appropriate neighborhood after it is moved, which requires communicating particles between processors responsible for adjacent neighborhoods or subdomains. Because of the large number of particles, it is beneficial for particle data to remain on the coprocessor.

Computing moments (block B5) requires random write-access to a grid. If multiple threads simultaneously accumulate particle moments on the same mesh, then one must use atomic writes to guard against the possibility that two threads may occasionally attempt to write simultaneously to the same grid node. Fortunately, the GPU provides efficient atomic writes for single-precision floating point numbers. Note that use of single-precision arithmetic for particles can be sufficiently accurate, since moments are obtained by summing over a large number of particles and particle positions can be represented relative to the local mesh cell [62].

Computing fields (B2) is the most challenging block to parallelize, due to the use of an implicit field solver. We execute the GMRES algorithm to perform each Newton iteration. In each cycle of the GMRES algorithm, the implicit field solver must repeatedly evaluate a residual, perhaps $m = 20$ times. Evaluating the residual can be vectorized in the same way as for an explicit field solver and involves communication only of face data between processors responsible for adjacent subdomains. A greater challenge to communication scaling is the global communication needed for convergence. The GMRES algorithm involves computing on the order of $m^2/2$ (perhaps 200) inner products, which require blocking global reductions. More sophisticated versions of GMRES, however, have recently become available: ℓ^1-GMRES requires only m global reductions, and a pipelined version called $p(\ell)$-GMRES permits nonblocking global reductions which allow synchronization to be delayed by ℓ iterations and thus can be interleaved with computation [63].

Effective preconditioning avoids the need for a large number of GMRES iterations and is of more primary importance than improving the computational efficiency of later iterations. An effective preconditioning

strategy for Maxwell's equations is to use geometric multigrid at the refined level (which avoids the need to explicitly form large, sparse matrices) combined with algebraic multigrid at the coarse level (which avoids the long-range communication bottlenecks that arise in architectures that fail to implement a communication network that scales in a self-similar way to the coarsest levels).

In high performance computing, codes often have to be adapted in order to avoid the criticalities which prevent good performances or scaling on particular architectures. Performance bottlenecks can be mitigated through software modifications (e.g., optimizations) or attacked in a more radical way using algorithmic changes. An example of such possibility will be provided here regarding global communication.

As already argued in Section 2.5 and Fig. 5, block B2 is responsible for most of the global communication in the code. Of the two iterative solvers present in B2, the GMRES used for Eq. (21) and the CG for the Poisson's correction of Eq. (14), the second is the main offender with respect to the amount of global communication.

In fact, with reasonable values for the convergence criterion, Equation (21) converges in a very low number of iterations, usually around 10, due to the fact that the Left Hand Side (LHS) term which poses the highest convergence problems, $\nabla^2 \mathbf{E}^{n+\theta}$, has low magnitude when compared to the other LHS members [64]. Equation (14), however, converges in a number of iterations closer to 100. Alternative ways to enforce Gauss's law, less intensive under the point of view of global communication, are thus investigated, following the tutorial in [65].

Recall that Gauss's law is automatically enforced if the continuity equation for charge, Eq. (13), is respected when depositing currents at the grid points. This is not the case with the PIC moment-deposition scheme described earlier in Section 2. A number of schemes [66, 67, 68, 69, 25] have been devised to deposit currents while respecting Eq. (13). In practice, the motion of a particle from a start to an end point is broken into multiple intermediate segments, whose number and spatial orientation change with the scheme. Charge flux parcels are deposited at the end of each segment and not just at the end point of the motion. Since the coordinates of multiple relay points (i.e., the intermediate points at which the particle changes direction or crosses a cell boundary) and several particle to grid interpolation operations may need to be calculated, charge conservation methods can be computationally intensive. Preliminary considerations on the computing speed of the different charge conservation methods are provided in [65], based on the number of segments the particle motion is broken into and on the number of IF statements in each algorithm. It can be safely stated, however, that each of the methods, being compute bound but not requiring any global communication, is an interesting alternative to the Poisson's correction to enforce Gauss's law in architectures like MICs or GPUs.

4.3. Future implementation strategy on DEEP's Booster-Cluster architecture

The DEEP project is implementing a Cluster-Booster architecture consisting of the *Cluster*, a traditional cluster of Xeon nodes, and the *Booster*, a collection of Xeon Phi coprocessors connected by high-speed interconnects to form a 3D torus. The Cluster and Booster are connected by an Infiniband network [28]. The intent of the DEEP architecture is to provide a mechanism to offload parts of an algorithm that scale well to the Booster while leaving on the Cluster parts whose scaling would be more difficult to improve. iPic3D is one of the few codes (and the only code for space weather simulations) which has been selected for porting to the DEEP's Cluster-Booster architecture. We will illustrate here how we intend to port the code to the DEEP architecture. The actual porting is still in progress and will be illustrated better in future papers.

We intend to represent particles on the Booster and to perform the implicit field solve on the Cluster. This requires that we transfer particle moments (consisting of 10 scalar fields) from the Booster to the Cluster in order to advance the fields and that we transfer the field data (consisting of 6 scalar fields) from the Cluster to the Booster in order to advance the particles at each cycle of the algorithm.

Our rationale for solving fields on the Cluster is that the implicit field solve involves a large number of global reductions, which should be more efficient on the Cluster. Particles, however, can be advanced completely independently of one another, and the only communication needed between processors is local communication to transfer particles; this communication pattern is well-suited to the Booster's 3D torus interconnect.

In plasma simulation, use of a large number of particles per mesh cell helps to reduce particle noise. The DEEP architecture promises to allow us to significantly increase the number of particles per mesh cell in iPic3D.

While we do not aim to achieve full scaling of the field solve with this division on the DEEP architecture, we expect to be able to handle an amount of field data sufficiently large for our purposes. While field data must be transferred to and from the Booster, the amount of data will be much less than the particle data, and the transfer needs to be made only once per cycle. Furthermore, the independence of particles means that communication to and from the Booster can be spread out over time and done concurrently with moving particles or accumulating moments.

5. Sample space weather simulations with iPic3D

For illustration of the current, already outstanding capabilities of iPic3D in simulating space weather processes, we report below two typical examples of iPic3D simulations. The first is aimed at studying processes developing in many space weather events: the conversion of magnetic energy to kinetic energy. The second is a global model of the interaction of the Earth space environment (the terrestrial magnetosphere) with incoming solar plasma (the solar wind).

5.1. Simulations of null point reconnection in 3D

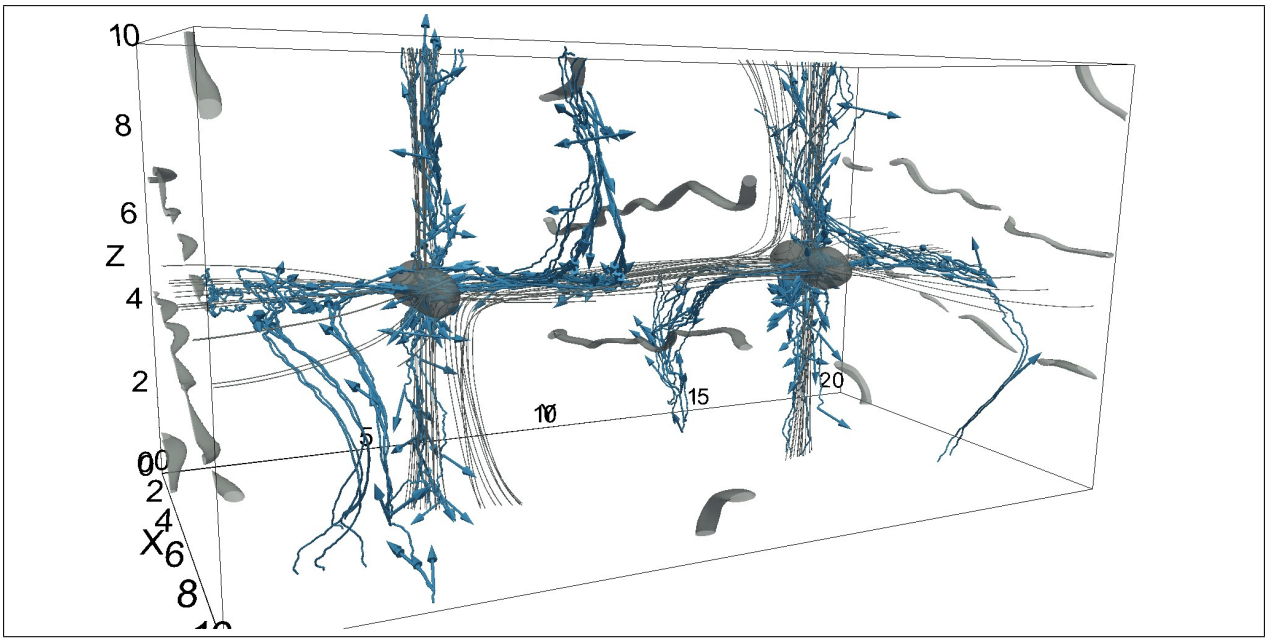


Figure 8: Three-dimensional simulations of magnetic reconnection with iPic3D, just before two null points are dissolved. The magnetic field lines are shown in grey; grey isocontours show regions of low magnetic field $B = 0.2B_0$; blue (darker arrows in the black and white version) are electron streamlines, demonstrating electron jets entering the null points along separatrices and being scattered away along spines. Note that only two of eight null points are shown here.

Using iPic3D, we have simulated the evolution of proton-electron plasma in a three-dimensional fully

periodic box with 8 uniformly spaced null points [70]:

$$B_x = -B_0 \cos\left(\frac{2\pi x}{L_x}\right) \sin\left(\frac{2\pi y}{L_y}\right),$$
$$B_y = B_0 \cos\left(\frac{2\pi y}{L_y}\right) \left[\sin\left(\frac{2\pi x}{L_x}\right) - 2\sin\left(\frac{2\pi z}{L_z}\right)\right],$$
$$B_z = 2B_0 \sin\left(\frac{2\pi y}{L_y}\right) \cos\left(\frac{2\pi z}{L_z}\right),$$

where B_0 is the magnetic field amplitude; L_x, L_y, and L_z are the sizes of the simulation domain in the corresponding directions. It is easy to show that this configuration satisfies the condition $\nabla \cdot \mathbf{B} = 0$.

The simulation domain represents a periodic cube with dimensions $L_x \times L_y \times L_z = 20\,d_i \times 20\,d_i \times 20\,d_i$ and with 400^3 cells and 27 particles of each species per cell. Our plasma consists of electrons and ions with mass ratio $m_i/m_e = 25$ and temperature ratio $T_i/T_e = 5$; the particles are initialized with Maxwellian velocity distributions with electron thermal velocity $v_{\mathrm{th},e}/c = 0.0346$. The initial particle density is uniform with $n_0 = 1$; the initial magnetic field amplitude is $B_0 = 0.02$. It is important to note that with such parameters the electron spatial scales are resolved well enough in our simulation: the electron skin depth $d_e = 0.2\,d_i = 4\Delta x$, where Δx is the grid spacing. The time step is set relative to ion plasma frequency: $\Delta t = 0.15/\omega_{pi}$. The ion cyclotron frequency is $\Omega_{ci} = eB_0/m_i c = 0.035\,\omega_{pi}$.

We use 16000 CPUs on a grid of $20 \times 20 \times 40$ to run this simulation for 10000 iterations ($\Omega_{ci} t = 50$). The simulation lasts 9 hours, which gives an efficiency of 0.2 ms/particle/iteration. The simulation is performed on the Curie supercomputer "thin" nodes. For parallel data reduction, routines written in Python are used to convert code output (HDF format) into parallel VTK files. Data visualization is done on a desktop computer using ParaView (running on a single CPU).

In our simulation, 82% of initial magnetic field energy is transferred to particle kinetic energy: the decay rate is almost 5 times higher than in traditional two-dimensional current sheet reconnection. Before a magnetic null point is dissolved by reconnection, electron beams stream towards the nulls along separatrices; electrons are scattered away along the spine, as shown in Fig. 8 for two null points located at [5, 5, 5] and [5, 15, 5]. This initial magnetic configuration was later studied in more details using the topological degree method to identify the details of magnetic topology and energy dissipation mechanisms [71]. Similar analysis of magnetic topology was lately applied to the variety of other iPic3D simulations in [72].

5.2. Global scale interactions between the solar wind and the magnetosphere

When the hot plasma ejected from the Sun interacts with a solid body in space, the trajectory of the plasma particles is modified: the plasma concentration, the electromagnetic field and the current systems around the object change. A large number of complex physical interactions can be observed in this system.

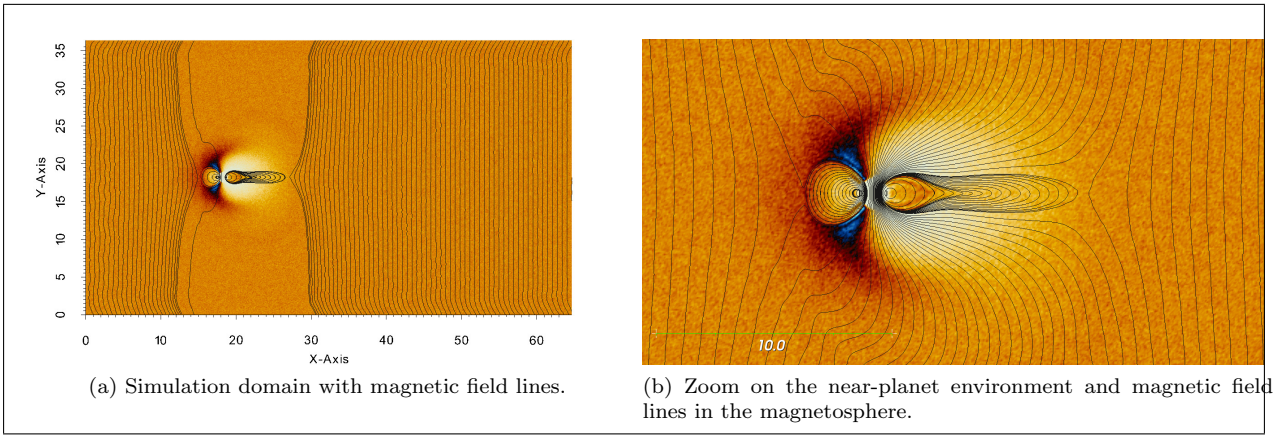
Up until recently, measurements were performed using isolated spacecrafts at discrete locations in time and space. The correlations between the different spatial and temporal plasma fluctuations were difficult to analyse using only one data point. Data from multiple spacecrafts were used to better understand unsteady phenomena. But with the advent of multi-point missions like Cluster II and THEMIS, the measurement of local spatial gradients and temporal correlations in the magnetosphere of the Earth has improved. Today we have a network of spacecrafts that constantly records measurements in the solar wind, in all the zones of the Earth's magnetosphere and around other planets of the solar system. However, obtaining an instantaneous picture of the full system at the global scale using such heterogeneous sources is still difficult [73].

A perfect addition to understand how the Sun affects the environment of solar system objects is to use powerful supercomputers and modern numerical methods to model the full system. Simulations at the scale of a planet have often been performed using Magneto-Hydro-Dynamics (MHD) codes. These models have shown good general agreement with measurements but rely on assumptions that may not be valid for the plasma flows observed in the solar wind [74]. Kinetic effects that may not be detected by MHD simulations include diffusion instabilities, magnetic reconnection, energy damping and electron runaway, all of which play an important role in the energy transfer between the solar wind and the planetary environment, one of

the main concerns in space weather research. A fully kinetic approach, where the ions and the electrons are represented as independent particles, is the best solution to attack the shortcomings of MHD simulations. The PIC technique is a powerful tool, but very CPU-intensive. To study the global scale interactions presented in this section, it is necessary to use implicit numerical methods on massively parallel computers.

Different authors have previously performed simulations of the kinetic phenomena in planetary magnetospheres. Omidi et al. [75] performed simulations on a Mercury-size magnetosphere using a hybrid code, where ions were treated as particles and electrons were considered a fluid. Cai et al. [76] used an explicit PIC code to perform simulations of a 3D magnetosphere. In the first case, the resolution of the electron kinetic scales was avoided, and in the second case, a "strong flow" case was simulated to avoid numerical instabilities of the explicit PIC code. The simulations presented in this section solve the global interaction at the electron, ion and planetary scales with physical parameters closer to nature, extending earlier work with iPic3D on the solar wind interaction with lunar magnetic anomalies and the formation of mini-magnetospheres [77, 78, 79]. The preliminary simulations discussed in this section are performed using the resources from the Tier-1 cluster in the Flemish Supercomputer Center (VSC) and the Curie and Fermi supercomputers from the PRACE infrastructure. Computations are performed using 512 to 4096 processors. Thanks to a new PRACE grant, we are planning to perform in the near future simulations using up to 100000 cores.

The numerical experiment shown in Fig. 9 is a two-dimensional domain discretized with a cartesian grid with a spatial resolution of $\Delta x = \Delta y = 0.06325\, d_i$. The total length of the box is $L_x = 64.768\, d_i$ in the x direction and $L_y = 36.432\, d_i$ in the y direction. A total of $N = 1024 \times 576 = 589\,824$ cells are used, each one of which contained 50 particles at the initialization of the simulations, for a total of $N_p = 29\,491\,200$ particles.



(a) Simulation domain with magnetic field lines.

(b) Zoom on the near-planet environment and magnetic field lines in the magnetosphere.

Figure 9: Electron concentration for the solar wind - magnetosphere simulation and magnetic field lines. Lengths are normalised with respect to the ion inertial lengths d_i.

A particle-absorbing sphere is located at $(x_c, y_c) = (L_y/2, L_y/2)$, and a dipolar magnetic field is imposed at the same location. This field is superimposed on a southward Interplanetary Magnetic Field (IMF) of strength $B_{\mathrm{IMF}} \approx 100$ nT.

At each iteration, particles reaching the right boundary are eliminated from the simulation (outflow condition), and in the remaining boundaries new particles are injected with a drift velocity in the x direction (inflow condition) using a Maxwellian distribution.

Boundary fields are fixed so as to impose a southward IMF and a consistent solar wind velocity of $V_{sw}/c = 0.015$, where c is the light speed. Different thermal velocities are imposed in the ions and electrons with values of $v_{\mathrm{th},e}/c = 7.4 \times 10^{-3}$ and $v_{\mathrm{th},e}/c = 0.05$. With these conditions, we ensure that the Alfvén Mach number is $M_A = 5$ and that the plasma beta is $\beta \approx 1$. A mass ratio of $m_i/m_e = 250$ is used.

Figure 9a shows a general view of the simulation box colored according to the electron concentration. The normalized scale goes from low concentration in white to high concentration in light blue. Magnetic

field lines are plotted using a stream tracer, each line originating at 100 equidistant points in the $y = L_y/2$ axis. The field lines show the creation of a tail in the solar wind flow direction. Due to the antiparallel nature of the closed field lines, two points of reconnection can be observed: one in the subsolar point and one in the tail. A transition between the IMF and the magnetospheric field can be observed in the line that crosses at the center of the planet, showing that the simulation captures the initial state of a shock-like feature. In Fig. 9b, the field lines are traced in order to highlight the structure of the magnetic field inside the magnetosphere. The transition between the IMF and the closed field lines is more obvious. It is also shown how the solar wind pressure wraps the IMF around the tail.

Strong variations in electron concentration can be observed in both images, the most important of which is observed in the north and south cusp regions. Electron concentration is also enhanced between the magnetopause and the upstream solar wind, and a cavity of low electron concentration, typical of the magnetotail lobes, can be observed behind the planet. An interesting feature captured by the simulation is the formation of a high-density zone of trapped particles in the closed magnetic lines close to the planet.

Our objective now is to extend this initial simulation to planetary scales comparable with the Earth, without compromising on resolution, for the study of the diffusion instabilities near the magnetopause in the equatorial plane and the reconnection points. We are currently performing simulations with the same spatial resolution but with a box ten times bigger in both axes compared to the previous simulation ($L_x = 647.68\, d_i$ and $L_y = 259.072\, d_i$). We are particularly interested in the capture of the bow shock, the tail formation and reconnection features and the cusp currents. Three-dimensional cases are also being simulated using lower resolutions.

6. Conclusions

Space weather is a fast-growing area of science rich in opportunities and challenges. A better understanding of space weather processes will inevitably result in a greater understanding of plasma behaviour as a whole, with important consequences for industrial and energy-related applications of plasma physics. The space weather community is called to develop fast and reliable forecasting tools. These tools have to be able to forecast both a black swan, Carrington-like event and the multiple, less disruptive but economically taxing space weather events that impact human technology on a day-to-day basis.

To achieve these aims, physics, rather than statistics, must be at the base of the developing of space weather forecasting tools. Such a physics-based description of space weather must be flexible and comprehensive enough to describe multiple processes, from solar storms to plasma instabilities, and multiple spatial and temporal scales. It also has to be suited to be implemented into algorithms and codes that exploit at best the characteristic of exascale oriented HPC machines.

We have identified a suitable model in fully kinetic, Particle In Cell methods. In Section 2 we have described an algorithm, the PIC Implicit Moment Method, and a code, iPic3D, which show excellent promise in these respects. We have then proceeded, in Section 3, to describe the algorithmic improvements that can make the method even more suitable for our aims: the fully implicit method (Section 3.1), the Multi-Level Multi-Domain method (Section 3.2) and the fluid-kinetic method (Section 3.3).

These algorithmic achievements are implemented within the existing iPic3D structure. However, algorithmic development is not the only way to go for better, timely space weather forecasts: as HPC capabilities develop, space weather forecasting modelling has to develop with them. Within the co-design approach, a feedback is established between application developers and machine designers. In particular, application developers need to have a clear understanding of new HPC architecture and to be able to adapt their codes in that direction. To fit the GPU and Xeon Phi architecture, priorities are vectorisation, parallelisation of the workflow and optimisation of internal communication. In Section 4, we have described the current status of the iPic3D code with regards to these issues and highlighted possible lines of development that will enable a better exploitation of current and future HPC hardware. We have also described our experience within the HPC project DEEP and our implementation choices for the porting of iPic3D towards the DEEP Cluster- Booster architecture.

Finally, in Section 5, we have provided successful examples of the simulation of space weather processes with the current version of the iPic3D code.

Acknowledgments

The authors wish to dedicate the present paper to the memory of Alec Johnson. He will be missed as a colleague and as a friend.
The present work is supported by the Onderzoekfonds KU Leuven (Research Fund KU Leuven) and by the European Commission's Seventh Framework Programme (FP7/2007-2013) via the Deep project (www.deep-project.eu) and the SWIFF project (www.swiff.eu). The simulations were conducted in part on the resources of the NASA Advanced Supercomputing Division (NAS), of the NASA Center for Computational Sciences Division (NCCS) and of the Vlaams Supercomputer Centrum (VSC). Support from the PRACE allocation number 2011050747 and from the PRACE project SWEET allowed the use of the Curie and Fermi supercomputers for the space weather examples reported in Section 5.

Bibliography

[1] D. N. Baker, What is space weather?, Adv. Space Res. 22 (1) (1998) 7 – 16, 10.1016/S0273-1177(97)01095-8.
[2] G. Siscoe, The space-weather enterprise: past, present, and future, J. Atmos. Sol. Terr. Phys. 62 (14) (2000) 1223 – 1232, 10.1016/S1364-6826(00)00074-2.
[3] V. Bothmer, I. A. Daglis, Space Weather – Physics and Effects, Praxis Publishing, 2007.
[4] L. J. Lanzerotti, Report of the assessment committee for the national space weather program, `http://www.nswp.gov/nswp_acreport0706.pdf` (June 2006).
[5] M. Innocenti, G. Lapenta, B. Vršnak, F. Crespon, C. Skandrani, M. Temmer, A. Veronig, L. Bettarini, S. Markidis, M. Skender, Improved forecasts of solar wind parameters using the kalman filter, Space Weather 9 (10) (2011) S10005. doi:10.1029/2011SW000659.
[6] S. Markidis, G. Lapenta, Rizwan-uddin, Multi-scale simulations of plasma with iPIC3D, Mathematics and Computers in Simulation 80 (7) (2010) 1509 – 1519.
[7] K. J. Bowers, B. J. Albright, L. Yin, W. Daughton, V. Roytershteyn, B. Bergen, T. J. T. Kwan, Advances in petascale kinetic plasma simulation with VPIC and Roadrunner, Journal of Physics Conference Series 180 (1) (2009) 012055–+. doi:10.1088/1742-6596/180/1/012055.
[8] R. Hockney, J. Eastwood, Computer simulation using particles, Taylor & Francis, 1988.
[9] C. Birdsall, A. Langdon, Plasma Physics Via Computer Simulation, Taylor & Francis, London, 2004.
[10] G. Lapenta, Particle simulations of space weather, Journal of Computational Physics 231 (3) (2012) 795–821.
[11] G. Lapenta, S. Markidis, S. Poedts, D. Vucinic, Space weather prediction and exascale computing (2012). doi:10.1109/MCSE.2012.86.
[12] R. Mason, Implicit moment particle simulation of plasmas, J. Computat. Phys. 41 (1981) 233.
[13] J. Denavit, Time-filtering particle stimulations with $\omega\_pe\Delta t \gg 1$, J. Computat. Phys. 42 (1981) 337.
[14] J. Brackbill, D. Forslund, Simulation of low frequency, electromagnetic phenomena in plasmas, J. Computat. Phys. 46 (1982) 271.
[15] A. Langdon, B. Cohen, A. Friedman, Direct implicit large time-step particle simulation of plasmas, J. Computat. Phys. 51 (1983) 107–138.
[16] H. X. Vu, J. U. Brackbill, CELEST1D: An implicit, fully-kinetic model for low-frequency, electromagnetic plasma simulation, Comput. Phys. Comm. 69 (1992) 253.
[17] H. Vu, J. Brackbill, Accurate numerical solution of charged particle motion in a magnetic field, J. Computat. Phys. 116 (1995) 384.
[18] K. Noguchi, C. Tronci, G. Zuccaro, G. Lapenta, Formulation of the relativistic moment implicit particle-in-cell method, Physics of plasmas 14 (2007) 042308.
[19] R. L. Morse, C. W. Nielson, Numerical Simulation of the Weibel Instability in One and Two Dimensions, Physics of Fluids 14 (1971) 830–840. doi:10.1063/1.1693518.
[20] A. B. Langdon, On enforcing Gauss' law in electromagnetic particle-in-cell codes, Computer Physics Communications 70 (1992) 447–450. doi:10.1016/0010-4655(92)90105-8.
[21] C. Munz, P. Omnes, R. Schneider, E. Sonnendrücker, U. Voß, Divergence Correction Techniques for Maxwell Solvers Based on a Hyperbolic Model, Journal of Computational Physics 161 (2000) 484–511. doi:10.1006/jcph.2000.6507.
[22] D. Knoll, G. Lapenta, J. Brackbill, A multilevel iterative field solver for implicit, kinetic, plasma simulation, Journal of computational physics 149 (2) (1999) 377–388.
[23] P. Kumar, S. Markidis, G. Lapenta, K. Meerbergen, D. Roose, High performance solvers for implicit particle in cell simulation, Procedia Computer Science 18 (2013) 2251–2258.
[24] P. Ricci, G. Lapenta, J. Brackbill, A simplified implicit maxwell solver, Journal of computational physics 183 (1) (2002) 117–141.
[25] G. Chen, L. Chacón, D. C. Barnes, An energy- and charge-conserving, implicit, electrostatic particle-in-cell algorithm, Journal of Computational Physics 230 (18) (2011) 7018–7036.
[26] S. Markidis, G. Lapenta, W. VanderHeyden, Z. Budimlić, Implementation and performance of a particle-in-cell code written in Java, Concurrency and Computation: Practice and Experience 17 (7-8) (2005) 821–837.

[27] G. Lapenta, Particle simulations of space weather, Journal of Computational Physics 231 (3) (2012) 795 – 821. doi:http://dx.doi.org/10.1016/j.jcp.2011.03.035.
URL http://www.sciencedirect.com/science/article/pii/S0021999111001860
[28] D. A. Mallon, N. Eicker, M. E. Innocenti, G. Lapenta, T. Lippert, E. Suarez, On the scalability of the clusters-booster concept: a critical assessment of the deep architecture, Proceedings of the Future HPC Systems: the Challenges of Power-Constrained Performance (2012) 3doi:10.1145/2322156.2322159.
URL http://dl.acm.org/citation.cfm?id=2322159
[29] Scalasca, http://www.scalasca.org/.
[30] D. Biskamp, Magnetic reconnection in plasmas, Vol. 3, Cambridge University Press, 2005.
[31] Judge, http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUDGE/JUDGE_node.html.
[32] Juqueen, http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/JUQUEEN_node.html.
[33] Top500, http://www.top500.org/lists/2013/06/.
[34] Y. Saad, Iterative methods for sparse linear systems, Society for Industrial and Applied Mathematics, 2003.
[35] D. A. Knoll, D. E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, Journal of Computational Physics 193 (2) (2004) 357–397.
[36] S. Markidis, Development of implicit kinetic simulation methods, and their application to ion beam propagation in current and future neutralized drift compression experiments., Ph.D. thesis, University of Illinois at Urbana-Champaign (2010).
[37] S. Markidis, G. Lapenta, The energy conserving particle-in-cell method, Journal of Computational Physics 230 (18) (2011) 7037–7052.
[38] G. Lapenta, S. Markidis, Particle acceleration and energy conservation in particle in cell simulations, Physics of Plasmas 18 (2011) 072101.
[39] J.-L. Vay, P. Colella, A. Friedman, D. Grote, P. McCorquodale, D. Serafini, Implementations of mesh refinement schemes for particle-in-cell plasma simulations, Computer physics communications 164 (1) (2004) 297–305.
[40] K. Fujimoto, S. Machida, Electromagnetic full particle code with adaptive mesh refinement technique: Application to the current sheet evolution, Journal of Computational Physics 214 (2) (2006) 550 – 566. doi:10.1016/j.jcp.2005.10.003.
[41] M. Innocenti, G. Lapenta, S. Markidis, A. Beck, A. Vapirev, A multi level multi domain method for particle in cell plasma simulations, Journal of Computational Physics 238 (0) (2013) 115 – 140. doi:10.1016/j.jcp.2012.12.028.
URL http://www.sciencedirect.com/science/article/pii/S0021999112007590
[42] A. Beck, M. Innocenti, G. Lapenta, S. Markidis, Multi-level multi-domain algorithm implementation for two-dimensional multiscale particle in cell simulations, Journal of Computational Physics (0) (2014) 430 – 443. doi:http://dx.doi.org/10.1016/j.jcp.2013.12.016.
URL http://www.sciencedirect.com/science/article/pii/S0021999113008152
[43] M. Innocenti, A. Beck, T. Ponweiser, S. Markidis, G. Lapenta, Introduction of temporal sub-stepping in the multi-level multi-domain semi-implicit particle-in-cell code parsek2d-mlmd, Computer Physics Communications 189 (0) (2015) 47 – 59. doi:http://dx.doi.org/10.1016/j.cpc.2014.12.004.
URL http://www.sciencedirect.com/science/article/pii/S001046551400410X
[44] M. Innocenti, A. Beck, S. Markidis, G. Lapenta, Momentum conservation in multi-level multi-domain (mlmd) simulations, Journal of Computational Physics 312 (2016) 14 – 18. doi:http://dx.doi.org/10.1016/j.jcp.2016.02.026.
URL http://www.sciencedirect.com/science/article/pii/S0021999116000905
[45] M. E. Innocenti, C. Norgren, D. L. Newman, M. V. Goldman, S. Markidis, G. Lapenta, Study of electric and magnetic field fluctuations from lower hybrid drift instability waves in the terrestrial magnetotail with the fully kinetic, semi-implicit, adaptive multi level multi domain method, Physics of Plasmas - accepted.
[46] G. Lapenta, Particle rezoning for multidimensional kinetic particle-in-cell simulations, Journal of Computational Physics 181 (2002) 317–337. doi:10.1006/jcph.2002.7126.
[47] F. F. Chen, M. Lieberman, Introduction to plasma physics and controlled fusion/Francis F., : Plenum Press, New York, 1984.
[48] S. Markidis, P.Henri, G.Lapenta, K. Roonmark, M. Hamrin, E. Laure, The fluid-kinetic particle-in-cell solver for plasma simulations, submitted to Journal of Computational Physics (2013).
[49] NVIDIA, Graphics processing unit (gpu) [cited May 2016].
URL http://www.nvidia.com/object/gpu.html
[50] Intel, Intel xeon phi product family [cited May 2016].
URL http://www.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-phi-coprocessor-brief.html
[51] A. Vapirev, J. Deca, G. Lapenta, S. Markidis, I. Hur, J.-L. Cambier, Initial results on computational performance of intel many integrated core, sandy bridge, and graphical processing unit architectures: implementation of a 1d c++/openmp electrostatic particle-in-cell code, Concurrency and Computation: Practice and Experience 27 (3) (2015) 581–593. doi:10.1002/cpe.3248.
URL http://dx.doi.org/10.1002/cpe.3248
[52] N. G. Dickson, K. Karimi, F. Hamze, Importance of explicit vectorization for CPU and GPU software performance, CoRR abs/1004.0024.
[53] D. Kirk, W. mei w. hwu, Programming Massively Parallel Processors: A Hands-on Approach (2010) 280.
[54] H. Wang, S. Potluri, M. Luo, A. Singh, S. Sur, D. Panda, MVAPICH2-GPU: optimized GPU to GPU communication for InfiniBand clusters, Computer Science - Research and Development 26 (3-4) (2011) 257–266. doi:10.1007/s00450-011-0171-3.
URL http://dx.doi.org/10.1007/s00450-011-0171-3
[55] W. Ma, S. Krishnamoorthy, O. Villay, K. Kowalski, Acceleration of streamed tensor contraction expressions on GPGPU-

based clusters, in: Cluster Computing (CLUSTER), 2010 IEEE International Conference on, 2010, pp. 207–216. doi:10.1109/CLUSTER.2010.26.

[56] D. Jacobsen, J. Thibault, I. Senocak, An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters, American Institute of Aeronautics and Astronautics, 2010.
URL http://dx.doi.org/10.2514/6.2010-522

[57] E. Phillips, M. Fatica, Implementing the himeno benchmark with CUDA on GPU clusters, in: Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on, 2010, pp. 1–10. doi:10.1109/IPDPS.2010.5470394.

[58] J. W. Choi, A. Singh, R. W. Vuduc, Model-driven autotuning of sparse matrix-vector multiply on GPUs, SIGPLAN Not. 45 (5) (2010) 115–126. doi:10.1145/1837853.1693471.
URL http://doi.acm.org/10.1145/1837853.1693471

[59] Z. Fan, F. Qiu, A. E. Kaufman, Zippy: A framework for computation and visualization on a gpu cluster, Computer Graphics Forum 27 (2) (2008) 341–350. doi:10.1111/j.1467-8659.2008.01131.x.
URL http://dx.doi.org/10.1111/j.1467-8659.2008.01131.x

[60] J. Stuart, J. Owens, Message passing on data-parallel architectures, in: Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on, 2009, pp. 1–12. doi:10.1109/IPDPS.2009.5161065.

[61] O. Lawlor, Message passing for GPGPU clusters: cudaMPI, in: Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on, 2009, pp. 1–8. doi:10.1109/CLUSTR.2009.5289129.

[62] G. Chen, L. Chacón, D. C. Barnes, An efficient mixed-precision, hybrid CPU–GPU implementation of a nonlinearly implicit one-dimensional particle-in-cell algorithm, Journal of Computational Physics 231 (16) (2012) 5374–5388.

[63] P. Ghysels, T. J. Ashby, K. Meerbergen, W. Vanroose, Hiding global communication latency in the GMRES algorithm on massively parallel machines, SIAM Journal on Scientific Computing 35 (1).

[64] G. Lapenta, J. Brackbill, P. Ricci, Kinetic approach to microscopic-macroscopic coupling in space and laboratory plasmas, Physics of plasmas 13 (2006) 055904.

[65] T. Umeda, Y. Omura, T. Tominaga, H. Matsumoto, Charge conservation methods for computing current densities in electromagnetic particle-in-cell simulations, Proceedings of ISSS-7 (2005) 26–31.

[66] J. W. Eastwood, The virtual particle electromagnetic particle-mesh method, Computer Physics Communications 64 (2) (1991) 252–266.

[67] J. Villasenor, O. Buneman, Rigorous charge conservation for local electromagnetic field solvers, Computer Physics Communications 69 (2) (1992) 306–316.

[68] T. Z. Esirkepov, Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor, Computer Physics Communications 135 (2) (2001) 144–153.

[69] T. Umeda, Y. Omura, T. Tominaga, H. Matsumoto, A new charge conservation method in electromagnetic particle-in-cell simulations, Computer Physics Communications 156 (1) (2003) 73–85.

[70] V. Olshevsky, G. Lapenta, S. Markidis, Energetics of kinetic reconnection in three-dimensional null points cluster, Phys. Rev. Lett..

[71] V. Olshevsky, A. Divin, E. Eriksson, S. Markidis, G. Lapenta, Energy Dissipation in Magnetic Null Points at Kinetic Scales, The Astrophysical Journal 807 (2015) 155. arXiv:1509.07961, doi:10.1088/0004-637X/807/2/155.

[72] V. Olshevsky, J. Deca, A. Divin, I. B. Peng, S. Markidis, M. E. Innocenti, E. Cazzola, G. Lapenta, Magnetic null points in kinetic simulations of space plasmas, The Astrophysical Journal 819 (1) (2016) 52.
URL http://stacks.iop.org/0004-637X/819/i=1/a=52

[73] T. Horbury, P. Louarn, M. Fujimoto, W. Baumjohann, L. Blomberg, S. Barabash, P. Canu, K. H. Glassmeier, H. Koskinen, R. Nakamura, C. Owen, T. Pulkkinen, A. Roux, J. A. Sauvaud, S. J. Schwartz, K. Svenes, A. Vaivads, Cross-scale : A multi-spacecraft mission to study cross-scale coupling in space plasmas, in: European Space Agency, (Special Publication) ESA SP, no. 598, 2006, pp. 561–568, qC 20101129.

[74] M. M. Echim, J. Lemaire, Ø. Lie-Svendsen, A Review on Solar Wind Modeling: Kinetic and Fluid Aspects, Surveys in Geophysics 32 (2011) 1–70. arXiv:1306.0704, doi:10.1007/s10712-010-9106-y.

[75] N. Omidi, X. Blanco-Cano, C. Russell, H. Karimabadi, Global hybrid simulations of solar wind interaction with Mercury: Magnetosperic boundaries, Advances in Space Research 38 (4) (2006) 632 – 638, ¡ce:title¿Mercury, Mars and Saturn¡/ce:title¿. doi:http://dx.doi.org/10.1016/j.asr.2005.11.019.
URL http://www.sciencedirect.com/science/article/pii/S0273117705014067

[76] D. Cai, K. ichi Nishikawa, B. Lembege, Magnetotail field topology in a three-dimensional global particle simulation, Plasma Physics and Controlled Fusion 48 (12B) (2006) B123.
URL http://stacks.iop.org/0741-3335/48/i=12B/a=S13

[77] J. Deca, A. Divin, G. Lapenta, B. Lembège, S. Markidis, M. Horányi, Electromagnetic Particle-in-Cell Simulations of the Solar Wind Interaction with Lunar Magnetic Anomalies, Physical Review Letters 112 (15) (2014) 151102. doi:10.1103/PhysRevLett.112.151102.

[78] J. Deca, A. Divin, B. Lembège, M. Horányi, S. Markidis, G. Lapenta, General mechanism and dynamics of the solar wind interaction with lunar magnetic anomalies from 3-d pic simulations, Journal of Geophysical Research: Space Physics (2015) 6443–64632015JA021070. doi:10.1002/2015JA021070.
URL http://dx.doi.org/10.1002/2015JA021070

[79] J. Deca, A. Divin, X. Wang, B. Lembège, M. Horányi, S. Markidis, G. Lapenta, 3-d full-kinetic simulation of the solar wind interaction with a vertical dipolar lunar magnetic anomaly, Geophysical Research Letters (2016) n/a–n/a2016GL068535. doi:10.1002/2016GL068535.
URL http://dx.doi.org/10.1002/2016GL068535