Journal of Computational Physics 491 (2023) 112383

Contents lists available at ScienceDirect

Journal of Computational Physics

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

An implicit particle code with *exact* energy and charge conservation for electromagnetic studies of dense plasmas

Justin Ray Angus [*], William Farmer, Alex Friedman, Debojyoti Ghosh, Dave Grote, David Larson, Anthony Link

Lawrence Livermore National Laboratory, Livermore, CA 94551, USA

ARTICLE INFO

Article history:
Received 3 March 2023
Received in revised form 5 June 2023
Accepted 14 July 2023
Available online 20 July 2023

Keywords:
Particle-in-cell
Monte-Carlo
High-energy-density plasmas
Implicit

ABSTRACT

A collisional particle code based on implicit energy- and charge-conserving methods is presented. A modified version of the *particle-suppressed* Jacobian-Free Newton-Krylov method that can enhance the solver efficiency is introduced. Mathematically, it is shown that this new approach can be viewed as a fixed-point iteration method for the particle positions. The model can *exactly* conserve global energy and local charge and can efficiently use time steps larger than the plasma period. The algorithm's ability to simulate dense plasmas accurately and efficiently is quantified by simulating the dynamic compression of a plasma slab via a magnetic piston in 1D planar geometry.

© 2023 Elsevier Inc. All rights reserved.

1. Introduction

The Maxwell-Boltzmann set of equations provides a full collisional-kinetic description of ideal plasmas and the electromagnetic fields with which they interact. The particle-in-cell (PIC) method is a commonly-used numerical approach for studying collisionless plasma systems [1,2]. Collisional physics can additionally be included using Monte-Carlo collision (MCC) methods [3,4]. Coupling a PIC algorithm with a MCC algorithm is straightforward. However, in electromagnetic systems, it is often found that the coupling of these two methods leads to a rapid unphysical change of the energy in the system. When collisions are included, conventional explicit PIC models are known to suffer from enhanced plasma heating [5], while conventional implicit PIC models (i.e., direct-implicit [6]) typically display enhanced plasma cooling [7,8]. It is important to note that this is true even when the PIC and MCC algorithms independently conserve energy well. Energy-conservation issues, along with other time-step and grid-size restraints, limit the applicability of conventional PIC-MCC methods for high-fidelity simulations of dense plasma systems, such as those found in inertial-confinement-fusion (ICF) and dense Z-pinch (DZP) experiments.

Plasmas found in ICF and DZP systems can be characterized as dense because the dynamic timescales of interest are often long compared to the timescales for plasma oscillations and collisions. While fluid models can be efficiently applied to study dense plasmas, they fail to capture kinetic effects associated with non-thermal particles. Examples include non-local heat flux at shock fronts [9] and transport/deposition of high-energy beam-like particles [10,11]. In this work we present a collisional particle code using the PIC-MCC method suitable for high fidelity simulations of dense plasmas. The algorithm is based on the relatively new fully implicit PIC method [12,13], which has the attractive property of being *exactly* energy conserving. It is shown in Ref. [14] that the fully implicit PIC method retains this property even when coupled to a

* Corresponding author.
E-mail address: angus1@llnl.gov (J.R. Angus).

https://doi.org/10.1016/j.jcp.2023.112383
0021-9991/© 2023 Elsevier Inc. All rights reserved.

model for collisions. Maintaining energy conservation is crucial for studying things like the dynamic compression of plasmas [8] - a feature that is common in both ICF and DZP experiments. Nonphysical plasma pressures resulting from numerical heating/cooling can lead to anomalously large/small compression ratios, which in turn can have a strong effect on the neutron yield in fusion-producing plasmas.

The fully implicit PIC method [12,13] is distinguished from conventional implicit PIC methods (which are only semi-implicit [15,16]) in that the macro-particle quantities are treated in a fully implicit manner. There is full consistency in this method between the particles (macro- is dropped for convenience), their moments, and the fields, permitting stable and robust integration with large time steps (relative to the plasma period and the Courant time step associated with the speed of light) and large grid cells (with respect to the Debye length and the species inertial lengths), while ensuring *exact* global energy conservation and local charge conservation. However, a drawback of this method is that advancing the system in time, which requires the solution of a large nonlinear system, is not trivial and can be computationally intensive. The feasibility of using the fully implicit PIC method is made possible using what we refer to as the *particle-suppressed* Jacobian-Free Newton Krylov (PS-JFNK) method [12,13]. This method uses a nonlinear elimination process to decouple the particle quantities from the system of unknowns, allowing one to apply the JFNK method to the field equations only while achieving the solution of the full field-particle system.

The PS-JFNK method, while computationally practical, is still relatively expensive (per particle per time step) as it can require the particles to be re-advanced many times during a time step. Fluid-based preconditioning methods [17,18] have been used to minimize the number of linear iterations in the PS-JFNK solve, which in turn minimizes the number of times the particles are re-advanced. Here, a different approach to improve the efficiency of the PS-JFNK method is taken by making assumptions that remove the need to re-advance the particles during the linear stage altogether. This is accomplished using the mass matrix formalism that has origins in conventional semi-implicit PIC methods [16] and is a central feature of energy-conserving semi-implicit (ECSIM) PIC methods [19,20]. Practically speaking, this method is implemented as a modified PS-JFNK method, but it is shown that this algorithm can be viewed as a modified fixed-point iteration method for the particle positions. From this viewpoint, it is shown how the algorithm can be reformulated as a predictor-corrector method for the particle positions that has a computational efficiency within a factor of two of the ECSIM PIC method.

The remainder of this paper is as follows. The Maxwell-Boltzmann system and the numerical algorithm are presented in Sec. 2. The original PS-JFNK method for advancing the nonlinear field-particle equations is discussed in Sec. 3. The new, modified implementation of this method (the main feature of this work) is given in Sec. 4. Results from numerical simulations are presented in Sections 5-6. In particular, in Sec. 6, a detailed parameter study in terms of grid size, time step, and number of particles is presented from simulations of the dynamic compression of a collisional plasma slab via a magnetic piston in 1D planar geometry. This test problem serves to illustrate the ability of the algorithm to study the dynamic compression of plasmas accurately and efficiently. The presentation of the aforementioned predictor-corrector algorithm is given in Sec. 7. Benefits and limitations of this algorithm are discussed. Several important aspects of the numerical algorithms, such as how they relate to previous work using fluid-based preconditioning methods and the computational cost associated with forming/using the mass matrices, are discussed in Sec. 8. A summary of the manuscript is given in Sec. 9.

2. The Maxwell-Boltzmann system and the implicit θ-PIC-MCC algorithm

The fully implicit PIC-MCC algorithm is described in this section. Starting with a presentation of the Maxwell-Boltzmann set of equations, the discrete numerical formulation follows, and a corresponding discrete energy conservation law is presented. Finally, the interpolation function used to achieve local charge conservation is discussed.

2.1. The Maxwell-Boltzmann set of equations

The Boltzmann equation governing the evolution of the phase-space distribution function of species α, f_α, can be written as

$$\frac{\partial f_\alpha}{\partial t} + \frac{d\mathbf{x}}{dt}\cdot\frac{\partial f_\alpha}{\partial \mathbf{x}} + \frac{d\mathbf{v}}{dt}\cdot\frac{\partial f_\alpha}{\partial \mathbf{v}} = C_\alpha. \tag{1}$$

The left-hand side is the phase-space continuity law governing long-range (i.e., collisionless) phenomena, and C_α on the right-hand side is the operator representing discrete discontinuities in phase space owing to short-range collisional processes. The time derivatives of the phase space coordinates are obtained from the equations of motion, which in the non-relativistic limit are

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}, \quad \frac{d\mathbf{v}}{dt} = \frac{q_\alpha}{m_\alpha}\left[\mathbf{E}(\mathbf{x}) + \mathbf{v}\times\mathbf{B}(\mathbf{x})\right], \tag{2}$$

where m_α and q_α are the mass and charge, respectively, of species α. The electric and magnetic fields, $\mathbf{E}$ and $\mathbf{B}$, are governed by the laws of Ampère and Faraday. These laws, along with the divergence constraints on $\mathbf{E}$ and $\mathbf{B}$, are

$$\frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t} = \nabla\times\mathbf{B} - \mu_0\mathbf{J}, \quad \nabla\cdot\epsilon_0\mathbf{E} = \rho, \tag{3}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla\times\mathbf{E}, \qquad \nabla\cdot\mathbf{B} = 0. \tag{4}$$

Here, $c = 1/\sqrt{\epsilon_0 \mu_0}$ is the speed of light, ρ is the charge density, and $\mathbf{J}$ is the current density. These latter quantities are obtained from the species distribution functions:

$$\rho = \sum_{\alpha \in s} q_\alpha \int f_\alpha d\mathbf{v}^3, \quad \mathbf{J} = \sum_{\alpha \in s} q_\alpha \int \mathbf{v} f_\alpha d\mathbf{v}^3, \tag{5}$$

where $d\mathbf{v}^3 \equiv \Pi_{i=1}^3 dv_i$ and the sum is over all species α. For a fully ionized ideal plasma, C_α in Eq. (1) is that corresponding to screened Coulomb collisions and is given by the Landau-Fokker-Planck equation:

$$C_\alpha = -\sum_{\beta \in s} \frac{\partial}{\partial v_j} \frac{q_\alpha^2 q_\beta^2 \log \Lambda}{8\pi \epsilon_0^2 m_\alpha} \int_{\mathbf{v}'} \left[\frac{\delta_{jk}}{u} - \frac{u_j u_k}{u^3} \right] \left[\frac{f_\alpha}{m_\beta} \frac{\partial f_\beta(\mathbf{v}')}{\partial v'_k} - \frac{f_\beta(\mathbf{v}')}{m_\alpha} \frac{\partial f_\alpha}{\partial v_k} \right] d\mathbf{v}', \tag{6}$$

where $u = |\mathbf{v} - \mathbf{v}'|$ is the relative speed, $\log \Lambda$ is the Coulomb logarithm, and the sum is over all species.

2.2. The implicit θ-PIC-MCC algorithm

The specific numerical stencil considered here to solve Eqs. (1)-(6) is the same as that presented previously in Ref. [14], which is an implicit θ-PIC method [12,15,21] coupled with a binary MCC model for Coulomb collisions [22]. The model is fully electromagnetic and fully implicit [12,13]. Coulomb collisions are included in the usual time-split fashion. That is, after the Vlasov-Maxwell equations are advanced in time using the θ-PIC algorithm, then the particle velocities are adjusted via the collision algorithm. The electric and magnetic fields used below to advance the system from time t_n to time $t_{n+1} = t_n + \Delta t$ are computed using quantities at $t_{n+\theta}$, which for arbitrary vector $\mathbf{h}$ is

$$\mathbf{h}^{n+\theta} = (1-\theta)\,\mathbf{h}^n + \theta \mathbf{h}^{n+1}. \tag{7}$$

The implicit-biasing parameter θ is chosen such that $1/2 \le \theta \le 1$. The lower bound of $1/2$ is required for numerical stability [14,15].

In a PIC scheme, the distribution function is approximated as a collection of discrete particles in the Klimontovich sense as

$$f_\alpha(\mathbf{x}, \mathbf{v}, t) \approx \sum_{p \in \alpha} w_p \delta\left(\mathbf{x} - \mathbf{x}_p(t)\right) \delta\left(\mathbf{v} - \mathbf{v}_p(t)\right), \tag{8}$$

where w_p is the particle weight. The mass and charge associated with particle $p \in \alpha$ are $m_p = w_p m_\alpha$ and $q_p = w_p q_\alpha$, respectively. In practice, the spatial δ function in Eq. (8) is replaced by a finite-width spline (i.e., shape function). The discrete version of the equations governing the phase-space trajectories of the particles (Eqs. (2)) are

$$\frac{\mathbf{x}_p^{n+1} - \mathbf{x}_p^n}{\Delta t} = \bar{\mathbf{v}}_p, \quad \frac{\mathbf{v}_p^{n+1} - \mathbf{v}_p^n}{\Delta t} = \frac{q_p}{m_p}\left(\mathbf{E}_p^{n+\theta} + \bar{\mathbf{v}}_p \times \mathbf{B}_p^{n+\theta}\right), \tag{9}$$

where $\bar{\mathbf{v}}_p \equiv \mathbf{v}_p^{n+1/2}$ is the time-centered particle velocity and $\mathbf{E}_p^{n+\theta} = \sum_g S_{gp}^{\mathbf{E}} \mathbf{E}_g^{n+\theta}$ and $\mathbf{B}_p^{n+\theta} = \sum_g S_{gp}^{\mathbf{B}} \mathbf{E}_g^{n+\theta}$ are the fields interpolated from the grid to the particle positions using the shape functions, $S_{gp}^{\mathbf{E}}$ and $S_{gp}^{\mathbf{B}}$.

The discrete versions of the laws of Ampère and Faraday, Eqs. (3)-(4), used to advance $\mathbf{E}$ and $\mathbf{B}$ at grid location $\mathbf{x}_g$ are given as

$$\frac{\mathbf{E}_g^{n+1} - \mathbf{E}_g^n}{c^2 \Delta t} = \nabla \times \mathbf{B}_g^{n+\theta} - \mu_0 \bar{\mathbf{J}}_g, \qquad \frac{\mathbf{B}_g^{n+1} - \mathbf{B}_g^n}{\Delta t} = -\nabla \times \mathbf{E}_g^{n+\theta}, \tag{10}$$

where $\bar{\mathbf{J}}_g \equiv \mathbf{J}_g^{n+1/2}$ is the time-centered current density at grid location $\mathbf{x}_g$ defined as

$$\bar{\mathbf{J}}_g = \sum_s \sum_{p \in s} \frac{q_p \bar{\mathbf{v}}_p}{\Delta V_g} S_{gp}^{\mathbf{J}}. \tag{11}$$

Here, $S_{gp}^{\mathbf{J}}$ is the shape function used to interpolate the current density associated with particle p to grid position $\mathbf{x}_g$. The same shape function used in Eq. (11) to deposit the particle current to the grid is also used to obtain the electric field used to advance the particles velocities in Eq. (9), $S_{gp}^{\mathbf{E}} = S_{gp}^{\mathbf{J}}$. This condition is required to achieve *exact* energy conservation (when using $\theta = 1/2$) [12,14]. The volume element in Eq. (11) is a modified volume element defined as $\Delta V_g = \Pi_{i=1}^D \Delta x_i \alpha_{g_i}$, where Δx_i is the grid spacing in direction i. The coefficient α_{g_i} equals one if the grid location x_{g_i} is in the interior of the domain and is equal to $1/2$ if x_{g_i} lives on the domain boundary.

The standard Yee grid [23] is used for the electric and magnetic fields; $\mathbf{J}$ and $\mathbf{E}$ are defined along cell edges and $\mathbf{B}$ is defined on cell faces. The Yee-grid formalism ensures that the physical condition $\nabla \cdot \mathbf{B} = 0$ is satisfied discretely.

2.3. Discrete energy law

A derivation of the discrete energy conservation law for the θ-PIC scheme considered here can be found in Refs. [12,14] considering periodic boundaries. Below, a generalized derivation that includes finite fluxes of electromagnetic energy at domain boundaries is presented [15]. For simplicity, we consider a planar geometry with uniform cell spacing Δx_i in each direction i. We also assume that the domain boundaries are aligned with the tangential components of the electric field. The energy in the fields at time step n is defined as

$$W^n_{fields} = W^n_E + W^n_B = \frac{\epsilon_0}{2}\sum_g \mathbf{E}^n_g \cdot \mathbf{E}^n_g \Delta V_g + \frac{1}{2\mu_0}\sum_g \mathbf{B}^n_g \cdot \mathbf{B}^n_g \Delta V_g. \tag{12}$$

With some algebraic manipulation of Eqs. (10) [14], the discrete energy law for the fields can be expressed as

$$\frac{W^{n+1}_{fields} - W^n_{fields}}{\Delta t} + \sum_g \nabla \cdot \boldsymbol{\mathcal{S}}^{n+\theta}_g \Delta V_g = -\sum_g \bar{\mathbf{J}}_g \cdot \mathbf{E}^{n+\theta}_g \Delta V_g - \frac{\phi}{\Delta t}\sum_g \left[\epsilon_0 \left|\mathbf{E}^{n+1}_g - \mathbf{E}^n_g\right|^2 + \frac{1}{\mu_0}\left|\mathbf{B}^{n+1}_g - \mathbf{B}^n_g\right|^2\right]\Delta V_g, \tag{13}$$

where $\phi = \theta - 1/2$. The second term on the left-hand-side is the discrete version of the volume integral of the divergence of the Poynting flux, $\boldsymbol{\mathcal{S}} = \mathbf{E} \times \mathbf{B}/\mu_0$. The explicit definition of this quantity is

$$\sum_g \nabla \cdot \boldsymbol{\mathcal{S}}^{n+\theta}_g \Delta V_g \equiv \frac{1}{\mu_0}\sum_g \left[\mathbf{B}^{n+\theta}_g \cdot \nabla \times \mathbf{E}^{n+\theta}_g - \mathbf{E}^{n+\theta}_g \cdot \nabla \times \mathbf{B}^{n+\theta}_g\right]\Delta V_g. \tag{14}$$

This term is identically zero for periodic boundary conditions. With some effort, this term can be written as a surface integral of an appropriately defined discrete definition of the Poynting flux,

$$\sum_g \nabla \cdot \boldsymbol{\mathcal{S}}^{n+\theta}_g \Delta V_g = \sum_{g\in\mathbf{A}} \boldsymbol{\mathcal{S}}^{n+\theta}_g \cdot \Delta\mathbf{A}. \tag{15}$$

The discrete Poynting flux, $\boldsymbol{\mathcal{S}}^{n+\theta}_g$, is defined on cell faces (it is co-located with $\mathbf{B}^{n+\theta}_g$). For one-dimensional problems (i.e., $D = 1$), $\Delta\mathbf{A} = \pm\hat{\mathbf{x}}$ for the upper/lower boundaries and $\boldsymbol{\mathcal{S}}^{n+\theta}_g = \mathcal{S}^{n+\theta}_{x,i}\hat{\mathbf{x}}$, and the surface integral is given as

$$\sum_{g\in\mathbf{A}} \boldsymbol{\mathcal{S}}^{n+\theta}_g \cdot \Delta\mathbf{A} = \mathcal{S}^{n+\theta}_{x,i}\Big|_{i=0}^{N_x} = \frac{1}{\mu_0}\left(E^{n+\theta}_{y,i}B^{n+\theta}_{z,i} - E^{n+\theta}_{z,i}B^{n+\theta}_{y,i}\right)\Big|_{i=0}^{N_x}, \tag{16}$$

where N_x is the number of grid cells. $E^{n+\theta}_y$ and $E^{n+\theta}_z$ live on the boundaries at $i = 0$ and $i = N_x$. $B^{n+\theta}_z$ and $B^{n+\theta}_y$ do not and are found by averaging their native values from the surrounding positions. E.g., $B^{n+\theta}_{y,i} = \left(B^{n+\theta}_{y,i+1/2} + B^{n+\theta}_{y,i-1/2}\right)/2$. An extension of Eq. (15) to $D > 1$ is given in Appendix A.

The energy in the particles at time step n is given as

$$W^n_{parts} = \sum_s \sum_{p\in s} \frac{m_p}{2}\left|\mathbf{v}^n_p\right|^2. \tag{17}$$

The change in energy in the particles after a single PIC advance (assuming no particles leave or enter the domain) is [14,12]

$$W^{n+1}_{parts} - W^n_{parts} = \sum_s \sum_{p\in s} m_p \bar{\mathbf{v}}_p \cdot \left(\mathbf{v}^{n+1}_p - \mathbf{v}^n_p\right) = \sum_g \bar{\mathbf{J}}_g \cdot \mathbf{E}^{n+\theta}_g \Delta V \Delta t. \tag{18}$$

The derivation of Eq. (18) requires that $S^{\mathbf{E}}_{gp} = S^{\mathbf{J}}_{gp}$, as mentioned previously. The change in total energy in the system, $W_{tot} = W_{fields} + W_{parts}$, when going from time step n to time step $n+1$ is obtained by combining Eqs. (13) and (18):

$$W^{n+1}_{tot} - W^n_{tot} + \sum_g \nabla \cdot \boldsymbol{\mathcal{S}}^{n+\theta}_g \Delta V_g \Delta t = -\phi \sum_g \left[\epsilon_0\left|\mathbf{E}^{n+1}_g - \mathbf{E}^n_g\right|^2 + \frac{1}{\mu_0}\left|\mathbf{B}^{n+1}_g - \mathbf{B}^n_g\right|^2\right]\Delta V_g. \tag{19}$$

The second term on the left side of Eq. (19) represents the electromagnetic energy fluxing across physical boundaries. The expression on the right side of Eq. (19) represents the unphysical numerical change in global energy after a time step. The sum in this term is positive definite and thus there is always a net energy loss when using $\phi > 0$. Another important aspect of Eq. (19) is that it is independent of whether collisions are included. Thus, the only way collisions affect the discrete change in energy after a time step (for equally weighted particles) is indirectly through changing the frequency spectrum and/or magnitude of the energy in the fields. Nevertheless, global energy is *exactly* conserved when using $\phi = 0$. It should be emphasized that this particular property of the θ-PIC algorithm is a significant advantage compared to conventional explicit and semi-implicit PIC methods, where coupling with a collision model can lead to a rapid unphysical change of the energy in the system [5,14].

2.4. Shape functions and charge conservation

One may choose any well-behaved shape function for interpolation in the algorithm. The energy conservation law presented above only requires that $S^{\mathbf{E}}_{gp} = S^{\mathbf{J}}_{gp}$. For the simulation results presented later in this paper, a first-order charge-conserving shape function for $S^{\mathbf{J}}_{gp}$ [18], which we refer to as the CC1 scheme, is used. Charge-conserving shape functions ensure that a discrete version of the continuity law for charge is identically preserved after each time step for each particle [24,25]. In turn, this ensures that the discrete divergence of the electric field on the grid is consistent with Gauss's law. That is,

$$\frac{\rho_g^{n+1} - \rho_g^n}{\Delta t} + \nabla \cdot \bar{\mathbf{J}}_g = 0 \;\Rightarrow\; \nabla \cdot \epsilon_0 \mathbf{E}_g^{n+1} = \rho_g^{n+1}. \tag{20}$$

Here, ρ_g is defined at cell nodes and the divergence operator is the second-order central differencing operator.

The CC1 scheme considered in this work uses a combination of the first-order spline (a.k.a., cloud-in-cell or CIC), S^1, and a second-order spline (a.k.a., triangular-shaped-cloud or TSC), S^2. These shape functions for interpolation in direction i with $\Delta_i \equiv \left(x_{g_i} - x_{p,i}\right)/\Delta x_i$ are defined as

$$S^1\left(x_{g_i} - x_{p,i}\right) = \begin{cases} 1 - |\Delta_i|, & \text{if } |\Delta_i| < 1 \\ 0, & \text{else,} \end{cases} \tag{21}$$

and

$$S^2\left(x_{g_i} - x_{p,i}\right) = \begin{cases} \frac{3}{4} - \Delta_i^2, & \text{if } |\Delta_i| < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2} - |\Delta_i|\right)^2, & \text{if } \frac{1}{2} < |\Delta_i| < \frac{3}{2} \\ 0, & \text{else.} \end{cases} \tag{22}$$

The CC1 scheme uses S^1 for interpolation in the direction parallel to the current density and S^2 for the other directions. This interpolation stencil involves the particle position at both the previous and future time steps and an orbit-average is used when particles cross a cell with the cell crossing locations defined at cell centers. This orbit-averaged method is described in Ref. [26], and is not to be confused with the sub-cycling method described in Refs. [13,18]. For example, the shape function for J_x at grid position $\mathbf{x}_g = [x_{i+1/2}, y_j]$ in a 2D simulation is

$$S^{J_x}_{i+1/2,j,p} = \sum_{\nu=0}^{N_\nu - 1} S^1\left(x_{i+1/2} - x_p^{\nu+1/2}\right) \frac{S^2\left(y_j - y_p^{\nu+1}\right) + S^2\left(y_j - y_p^{\nu}\right)}{2} \left[\frac{x_p^{\nu+1} - x_p^{\nu}}{x_p^{n+1} - x_p^{n}}\right], \tag{23}$$

where the sum is over N_ν sub-orbits defined by cell crossings for particle p going from $\mathbf{x}_p^n$ to $\mathbf{x}_p^{n+1}$ during time step Δt. In this notation, $x_p^{\nu=0} \equiv x_p^n$ and $x_p^{\nu+1=N_\nu} \equiv x_p^{n+1}$. The last term in Eq. (23) is the orbit-average factor. The CC1 scheme identically preserves local charge density for each particle defined at cell nodes as

$$\rho_g^n = \frac{1}{\Delta V_g} \sum_p q_p \Pi_{i=1}^{D} S^2\left(x_{g_i} - x_{p,i}^n\right). \tag{24}$$

Further details on the CC1 scheme can be found in Refs. [18,26].

The S^1 shape function given in Eq. (21) with the time-centered particle positions is used for each component of the magnetic field; $S^{B_x}_{gp} = S^{B_y}_{gp} = S^{B_z}_{gp} = \Pi_{i=1}^{D} S^1\left(x_{g_i} - \bar{x}_{p,i}\right)$. For reduced dimensional simulations with $D < 3$, the shape functions for the out-of-plane components of $\mathbf{J}$ and $\mathbf{E}$ are also computed using this stencil.

3. The nonlinear field-particle system and the particle-suppressed Jacobian-Free Newton-Krylov method

3.1. Formal presentation of the nonlinear system residual

The coupled field-particle system in the implicit θ-PIC method forms a nonlinear set of equations. The advance of the system variables from time t_n to time $t_{n+1} = t_n + \Delta t$ is done in a two-step process. First, a nonlinear system is solved for the fields at time $t_{n+\theta}$ and the particles at time $t_{n+1/2}$. Then, the field and particle quantities are transformed to time t_{n+1} using Eq. (7). The nonlinear system variables can be formally written as $\mathbf{X} = \left[\mathbf{X}_f, \mathbf{X}_p\right]$, where $\mathbf{X}_f = \left[\mathbf{E}_g^{n+\theta}, \mathbf{B}_g^{n+\theta}\right]$ represents the set of field variables on the numerical grid and $\mathbf{X}_p = \left[\bar{\mathbf{x}}_p, \bar{\mathbf{v}}_p\right]$ represents the set of variables associated with the particles. The nonlinear system residual is similarly expressed as $\mathbf{F}(\mathbf{X}) = \left[\mathbf{F}_f(\mathbf{X}), \mathbf{F}_p(\mathbf{X})\right] = 0$, where the nonlinear residuals for the field and particle systems are defined as

$$\mathbf{F}_f\left(\mathbf{X}_f,\mathbf{X}_p\right)=\begin{cases}\mathbf{E}_g^{n+\theta}-\mathbf{E}_g^{n}-\theta\Delta t c^2\left[\nabla\times\mathbf{B}_g^{n+\theta}-\mu_0\bar{\mathbf{J}}_g\left(\mathbf{X}_p\right)\right]=0,\\ \mathbf{B}_g^{n+\theta}-\mathbf{B}_g^{n}+\theta\Delta t\nabla\times\mathbf{E}_g^{n+\theta}=0,\end{cases} \tag{25}$$

$$\mathbf{F}_p\left(\mathbf{X}_f,\mathbf{X}_p\right)=\begin{cases}\bar{\mathbf{x}}_p-\mathbf{x}_p^{n}-\frac{\Delta t}{2}\bar{\mathbf{v}}_p=0,\\ \bar{\mathbf{v}}_p-\mathbf{v}_p^{n}-\frac{\Delta t}{2}\frac{q_p}{m_p}\left[\mathbf{E}_p^{n+\theta}\left(\bar{\mathbf{x}}_p,\mathbf{E}_g^{n+\theta}\right)+\bar{\mathbf{v}}_p\times\mathbf{B}_p^{n+\theta}\left(\bar{\mathbf{x}}_p,\mathbf{B}_g^{n+\theta}\right)\right]=0.\end{cases} \tag{26}$$

The number of degrees of freedom associated with $\mathbf{X}_f$ is $6N_g$ and that associated with $\mathbf{X}_p$ is $6N_p$, where N_g is the total number of grid points and N_p is the total number of particles.

3.2. The particle-suppressed Jacobian-Free Newton-Krylov method

The Jacobian-Free Newton-Krylov (JFNK) method [27] is a standard approach to solving large nonlinear systems. JFNK is a nested iterative method wherein inner linear iterations are used to solve for the Newton correction at each outer nonlinear Newton iteration. The linear stage is done using a Jacobian-free method, such as Generalized Minimal RESidual (GMRES). The important aspects of JNFK relevant to the discussion here are 1) at each nonlinear iteration $nl+1$, the Newton correction $\delta\mathbf{X}$ is found by solving the linear system $\mathbf{F}'\left(\mathbf{X}^{(nl)}\right)\delta\mathbf{X}=-\mathbf{F}\left(\mathbf{X}^{(nl)}\right)$ using GMRES, and 2) direct computation/storage of the system Jacobian, $\mathbf{F}'\equiv\partial\mathbf{F}/\partial\mathbf{X}$, is not required. Instead, what is needed at each linear iteration is the action of the system Jacobian on a given vector $\mathbf{V}$. This, in turn, can be computed numerically as

$$\frac{\partial\mathbf{F}}{\partial\mathbf{X}}\mathbf{V}=\frac{\mathbf{F}\left(\mathbf{X}^{(nl)}+\epsilon\mathbf{V}\right)-\mathbf{F}\left(\mathbf{X}^{(nl)}\right)}{\epsilon}, \tag{27}$$

where ϵ is a small parameter. In other words, the JFNK method requires one to compute the system residual $\mathbf{F}$ for some given state vector $\mathbf{X}$ at each nonlinear *and* at each linear iteration.

Solving Eqs. (25)-(26) directly with JFNK is often impractical for problems where $N_p\gg N_g$. One reason for this is because of the excessive memory requirements that are needed to build the Krylov subspace [17]. However, it turns out that there exist reformulations of the nonlinear field-particle system where the system residual can be cast in terms of the field quantities alone. This method (PS-JFNK) uses nonlinear elimination to self-consistently update the particle quantities, $\mathbf{X}_p$, for given field quantities, $\mathbf{X}_f$, at each evaluation of $\mathbf{F}_f$ in the JFNK process. A thorough mathematical formulation of the PS-JFNK method can be found in Sec. 4 of Ref. [13]. A brief overview of this formulation is given here. As presented in Eqs. (25)-(26), the system variables and residual can both be separated into two sets - one for the field quantities and one for the particle quantities. We now assume that $\mathbf{F}_p\left(\mathbf{X}_f,\mathbf{X}_p\right)=0$ can be written in an explicit form as $\mathbf{X}_p=\mathbf{f}_p\left(\mathbf{X}_f\right)$. That is, given $\mathbf{X}_f$, it is straightforward to solve for $\mathbf{X}_p$. This is possible because 1) the full set of particle quantities is actually N_p independent sets of 6 quantities (i.e., each particle is independent of all other particles) and 2) for given values of the fields on the grid, the position and velocity vector of each particle from Eqs. (9) can readily be obtained using a Newton method [12] or Picard iterations [13,18,28]. It then follows that the residual for the fields can be reformulated as

$$\mathbf{F}_f\left(\mathbf{X}_f,\mathbf{X}_p=\mathbf{f}_p\left(\mathbf{X}_f\right)\right)=\mathbf{G}_f\left(\mathbf{X}_f\right)=0. \tag{28}$$

This new residual, $\mathbf{G}_f$, has a lower dimensionality than $\mathbf{F}$, but has the same nonlinear solution.

The JFNK method requires one to compute the system residual at each nonlinear *and* at each linear iteration. For each residual calculation in the PS-JFNK process, one must self-consistently update the particle quantities and then compute the current density from the particles. Thus, each residual evaluation has a computational cost similar to that needed for a full time step advance in a conventional explicit PIC code. The cost actually exceeds that of an explicit PIC step because the self-consistent update of the particles is itself an iterative process. In order to improve the algorithm's efficiency, fluid-based preconditioning methods [17,18] have been used to minimize the number of linear iterations in the PS-JFNK solve. Here, a different approach is taken to improve the efficiency of the PS-JFNK method by making certain assumptions that remove the need to re-advance the particles during the linear GMRES iterations. This is accomplished using the mass matrix formalism that is a central feature of ECSIM PIC methods [19,20,29].

3.3. The mass matrix and energy-conserving semi-implicit PIC

The derivation of the expression for the current density using the mass matrices starts by using standard vector identities with Eq. (9) to write the time-centered particle velocity as

$$\bar{\mathbf{v}}_p=\frac{\mathbf{v}_p^{n}+\alpha_s\mathbf{E}_p^{n+\theta}+\left(\mathbf{v}_p^{n}+\alpha_s\mathbf{E}_p^{n+\theta}\right)\times\mathbf{b}_p+\left(\mathbf{v}_p^{n}+\alpha_s\mathbf{E}_p^{n+\theta}\right)\cdot\mathbf{b}_p\mathbf{b}_p}{1+b_p^2}, \tag{29}$$

where $\alpha_s=\frac{\Delta t}{2}q_p/m_p$ and $\mathbf{b}_p=\alpha_s\mathbf{B}_p^{n+\theta}$. Substituting $\bar{\mathbf{v}}_p$ from Eq. (29) into Eq. (11) produces

$$\bar{\mathbf{J}}_g = \sum_s \sum_{p\in s} \frac{q_p}{\Delta V_g} \left(\frac{\mathbf{v}_p^n + \mathbf{v}_p^n \times \mathbf{b}_p + \mathbf{v}_p^n \cdot \mathbf{b}_p \mathbf{b}_p}{1+b_p^2} + \alpha_s \frac{\mathbf{E}_p^{n+\theta} + \mathbf{E}_p^{n+\theta} \times \mathbf{b}_p + \mathbf{E}_p^{n+\theta} \cdot \mathbf{b}_p \mathbf{b}_p}{1+b_p^2} \right) S_{gp}^{\mathbf{J}}. \tag{30}$$

If we now make the assumptions that the particle positions and the magnetic field in the Lorentz force are fixed, then the current density can be expressed as a linear function of the electric field on the grid as

$$\bar{\mathbf{J}}_g = \hat{\mathbf{J}}_g + \sum_{g'} \mathbb{M}_{gg'} \cdot \mathbf{E}_{g'}^{n+\theta}. \tag{31}$$

Defining the particle magnetic field rotation matrix as

$$\mathbb{R}_p \equiv \frac{1}{1+b_p^2} \left(\mathbb{I} - \mathbb{I} \times \mathbf{b}_p + \mathbf{b}_p \mathbf{b}_p \right), \tag{32}$$

the new quantities introduced in Eq. (31), $\hat{\mathbf{J}}_g$ and $\mathbb{M}_{gg'}$, may be defined as

$$\hat{\mathbf{J}}_g \equiv \sum_s \sum_{p\in s} \frac{q_p}{\Delta V_g} \mathbb{R}_p \mathbf{v}_p^n S_{gp}^{\mathbf{J}} \quad \text{and} \quad \mathbb{M}_{gg'}^{ij} \equiv \sum_s \frac{\alpha_s}{\Delta V_g} \sum_{p\in s} q_p \mathbb{R}_p^{ij} S_{g'p}^{E_j} S_{gp}^{J_i}, \tag{33}$$

$\mathbb{M}_{gg'}$ is a nonlocal tensor quantity that is referred to as the mass matrix. Recall that *exact* energy conservation requires that $S_{gp}^{E_i} = S_{gp}^{J_i}$ for each of $i = x, y, z$. More detail on the mass matrix tensor $\mathbb{M}_{gg'}$ using a charge-conserving shape function is given in Appendix B.

The assumptions used to obtain Eq. (31), fixed particle positions and a fixed magnetic field in the Lorentz force, are explicit features of the ECSIM PIC method [19]. In that method, the particle positions are staggered in time with respect to the particle velocities and the field quantities, and the magnetic field from the previous time step, $\mathbf{B}_g^n$, is used in the Lorentz force. Since the energy conservation law presented in Eq. (19) does not depend on either the particle positions nor on the magnetic field in the Lorentz force, the ECSIM method is *exactly* energy conserving.

In terms of computational efficiency per particle per time step, the ECSIM method is superior to fully implicit since the equations governing the fields are linear (no Newton iterations), the particle positions and velocities are decoupled (no Picard iterations), and Eq. (31) can be used for the linear field solve, which does not require any particle-based operations during the GMRES iterations. However, the ECSIM method is not compatible with charge-conserving current deposition schemes, the inclusion of relativistic effects [30], and the particle orbits are only first order accurate with respect to the Lorentz force. Alternative methods to achieve charge conservation have been developed for ECSIM PIC [29,31]. Below we introduce a modified implementation of the PS-JFNK method for solving the fully implicit system that leverages ideas from the ECSIM method to enhance its computational efficiency.

4. The modified PS-JFNK method for solving the implicit θ-PIC-MCC equations

The new, modified PS-JFNK method for advancing the θ-PIC-MCC system from time step t_n to time step $t_{n+1} = t_n + \Delta t$ is outlined in **Algorithm 1**. The main modification is that the grid-based expression in Eq. (31) is used when computing the action of the system Jacobian on a given field vector during the linear GMRES iterations. That is, variations via the particle positions and the magnetic field in the Lorentz force are neglected in the system Jacobian used to compute the Newton correction for the fields. As the particle quantities are self-consistently updated at each Newton iteration $nl > 0$ using the Picard method given in **Algorithm 2**, these modifications do not alter the converged solution. After the field-particle system converges, the particle velocities are updated via the binary MCC algorithm. These velocities serve as the initial velocities for the next time step advance.

Algorithm 1 Modified PS-JFNK method for advancing the θ-PIC-MCC system from t_n to $t_{n+1} = t_n + \Delta t$.

```
 1: Initial half time step advance of particle positions: $\bar{\mathbf{x}}_p^{(nl=0)} = \mathbf{x}_p^n + \mathbf{v}_p^n \Delta t/2$.
 2: Initial guess for fields at $t_{n+\theta}$: $\mathbf{E}_g^{(nl=0)} = \mathbf{E}_g^n$ and $\mathbf{B}_g^{(nl=0)} = \mathbf{B}_g^n$.
 3:     Compute mass matrix quantities, $\hat{\mathbf{J}}_g$ and $\mathbb{M}_{gg'}$, from Eqs. (33) using $\bar{\mathbf{x}}_p^{(nl)}$ and $\mathbf{B}_g^{(nl)}$.
 4:     Compute the norm of the field residual from Eq. (25), $|\mathbf{F}_f^{(nl)}|$, using Eq. (31) for $\bar{\mathbf{J}}_g$.
 5:     If $nl > 0$, check $|\mathbf{F}_f^{(nl)}|$ for convergence. If converged, go to step 10. Else, proceed to step 6.
 6:     Use GMRES to compute the Newton correction, $\delta\mathbf{X}_f$, using Eq. (31) for $\bar{\mathbf{J}}_g$.
 7:     Update the field quantities: $[\mathbf{E}_g^{(nl+1)}, \mathbf{B}_g^{(nl+1)}] = [\mathbf{E}_g^{(nl)}, \mathbf{B}_g^{(nl)}] + \delta\mathbf{X}_f$.
 8:     With $\mathbf{E}_g^{(nl+1)}$, $\mathbf{B}_g^{(nl+1)}$, and $\bar{\mathbf{x}}_p^{(nl)}$, use Algorithm 2 to update the particle quantities, $\bar{\mathbf{x}}_p^{(nl+1)}$ and $\bar{\mathbf{v}}_p^{(nl+1)}$.
 9:     Update the nonlinear iteration number $nl$ and return to step 3.
10: Solution converged. Use Eq. (7) to transform the system variables from $t_{n+\theta}$ and $t_{n+1/2}$ to $t_{n+1}$.
11: Modify the particle velocities, $\mathbf{v}_p^{n+1}$, using the binary MCC algorithm for Coulomb collisions.
```

Algorithm 2 Picard method for self-consistent update of particle quantities for fixed $\mathbf{E}_g^{n+\theta}$ and $\mathbf{B}_g^{n+\theta}$ and provided an initial guess for the mean particle positions, $\bar{\mathbf{x}}_p^{(k=0)}$.

```
1: Interpolate fields from grid to particles: $\mathbf{E}_p^{(k)}\left(\mathbf{E}_g^{n+\theta}, \bar{\mathbf{x}}_p^{(k)}\right)$ and $\mathbf{B}_p^{(k)}\left(\mathbf{B}_g^{n+\theta}, \bar{\mathbf{x}}_p^{(k)}\right)$.
2: Compute the mean particle velocity, $\bar{\mathbf{v}}_p^{(k)}$, using Eq. (29).
3: Compare $|\bar{\mathbf{x}}_p^{(k)} - \mathbf{x}_p^n|$ with $|\bar{\mathbf{v}}_p^{(k)}\Delta t/2|$ for convergence. Finish if converged. Else, proceed to step 4.
4: Re-advance the particle positions: $\bar{\mathbf{x}}_p^{(k+1)} = \mathbf{x}_p^n + \bar{\mathbf{v}}_p^{(k)}\Delta t/2$.
5: Update the iteration number $k$ and return to step 1.
```

While there is some additional overhead cost associated with computing the mass matrices at each Newton iteration, there can be significant computational savings in the overall scheme by removing the need to do particle-based operations during the linear iterations. Furthermore, the elements of the mass matrices can be directly used for preconditioning the linear system being solved by GMRES [18,19]. The cost of a linear iteration is relatively cheap in our implementation since it scales with N_g and not with N_p, but preconditioning can still be essential for wall time improvements at sufficiently large time steps where the number of GMRES iterations needed to converge become large. A preconditioner matrix approximates the full matrix (which in the PS-JFNK method is $\partial\mathbf{F}_f/\partial\mathbf{X}_f$) but is more readily *inverted*. The preconditioner matrix $\mathbb{P}$ used in this work consists of the source-free Maxwell equations operator and the local, diagonal elements of $\mathbb{M}_{gg'}$ and is given as

$$\mathbb{P} = \begin{bmatrix} \mathbb{I} + \theta\Delta t\epsilon_0\mathbb{I}\cdot\mathbb{M}_{gg}\mathbb{I} & \theta\Delta t\epsilon_0\mathbb{I}\times\nabla \\ -\theta\Delta t\mathbb{I}\times\nabla & \mathbb{I} \end{bmatrix}. \tag{34}$$

The first step in **Algorithm** 1 is to do a half time step advance of the particle positions using the velocities from the end of the previous step. This is done in place of an initial self-consistent update of the particles via **Algorithm** 2 using the initial guess for the fields, which may or may not be close to the converged solution. In either case, there is no point wasting effort to achieve a self-consistent solution for the particle quantities with the initial guess for the fields. Note that the initial guess for the mean particle position used when **Algorithm** 2 is called from **Algorithm** 1 at nonlinear iteration *nl* is the value from the end of the previous Newton iteration. Thus, fewer iterations are needed for **Algorithm** 2 to converge as the full nonlinear system gets closer to convergence.

The numerical algorithm outlined in **Algorithms** 1-2 for advancing the implicit θ-PIC-MCC set of equations is implemented in a particle code framework called PICNIC. PICNIC is written in C++/Fortran and uses the Chombo library [32] for data containers and MPI handling. The PETSc library [33] is used for the JFNK solver. Linear equations involving the preconditioner matrix $\mathbb{P}$ given in Eq. (34) are solved using the additive Schwarz method in PETSc. Results from some numerical simulation studies using this algorithm are given below in Secs. 5-6. A Further discussion and mathematical analysis of the modified PS-JFNK method presented here is provided in Secs. 7-8.

5. 2D simulations of electron-proton thermalization

Collisional thermalization of a homogeneous electron-proton plasma in a 2D periodic domain is considered for the first simulation test. The plasma is initialized with $n_i = n_e = n_0 = 1.0\times10^{30}\,/\mathrm{m}^3$, $T_e = 150\,\mathrm{eV}$, and $T_i = 50\,\mathrm{eV}$. The simulation domain is square with side lengths $L = 10\delta_e$, where $\delta_e = c/\omega_{pe}$ is the electron skin depth and $\omega_{pe} = \sqrt{n_0e^2/m_e\epsilon_0}$ the plasma frequency. The grid resolution in each direction is set to $0.25\delta_e$ and the time step is set by $\omega_{pe}\Delta t = 0.1$. This time step satisfies the CFL condition with respect to the speed of light and collisions are well resolved as $\omega_{pe}\tau_e = 6.47 - 11.9$. Here, $\tau_e[\mathrm{s}] = 3.44\times10^5 T_e[\mathrm{eV}]^{3/2}/n_e[1/\mathrm{cm}^3]/\log\Lambda$ is the characteristic electron collision time. $\log\Lambda = 3$ is used for all collisions (e-e, i-i, and i-e) in this numerical test. The particles are initialized using 256 equally weighted particles per cell ($N_{ppc} = 256$) for each species. The initial particle positions are uniformly spaced on the grid and their velocities are obtained by randomly sampling from a Maxwellian distribution function. The relative tolerance for the Newton, Picard, and GMRES solvers for this problem are 10^{-12}, 10^{-10}, and 10^{-6}, respectively.

Time traces of the spatially-averaged electron and ion temperatures are shown in the left panel of Fig. 1. The temperature profiles are compared with those obtained by solving the following 0D thermalization model:

$$\frac{dT_e}{dt} = -\nu_T\left(T_e - T_i\right) = -\frac{dT_i}{dt}, \tag{35}$$

where $\nu_T \equiv 2m_e/M_i/\tau_e$ is the characteristic thermalization rate. The black dashed curves in the left panel of Fig. 1, obtained by solving Eqs. (35) numerically using a forward Euler method, agree well with the simulation. The fact that the temperature profiles from the simulation relax at a slightly slower rate than what the 0D model predicts is expected [34]. Furthermore, because a small amount of plasma energy gets converted into electrostatic and electromagnetic modes on the grid [14], the temperatures from the simulation relax to a slightly smaller value than what the 0D model predicts.

The change of total energy in the system, $W_{tot} = W_{fields} + W_{parts}$, relative to the initial energy in the system $W_0 = 3n_0T_0$ is shown in the middle panel of Fig. 1. The magnitude of the relative error in global energy conservation is around 5×10^{-16}

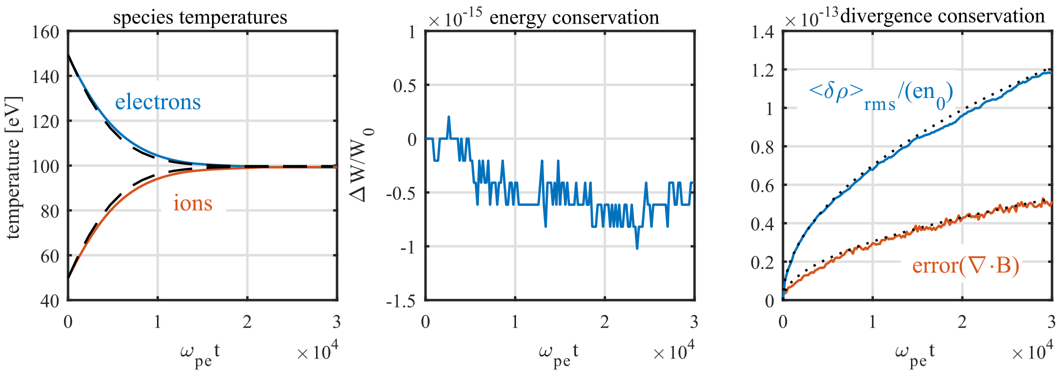

Fig. 1. Results from 2D simulations of the thermalization of a homogeneous electron-proton plasma. Time profiles of the species temperatures, averaged over the simulation domain, are shown in the left panel. The black dashed curves in the left panel are obtained from the 0D thermalization model (Eqs. (35)). The relative change in total energy, $\Delta W/W_0 = \left(W_{fields}(t) + W_{parts}(t) - W_0\right)/W_0$, is shown in the middle panel. Measures to assess the errors in local charge conservation and the $\nabla\cdot\mathbf{B}=0$ constraint via Eqs. (36) are shown in the right panel. The black dotted curves here scale with $\sqrt{\omega_{pe}t}$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

- on the order of machine precision. The mathematical formulas used to assess the error in local charge conservation and the $\nabla\cdot\mathbf{B}=0$ constraint are

$$\langle\delta\rho\rangle_{rms} = \sqrt{\frac{\sum_g \left|\nabla\cdot\epsilon_0\mathbf{E}_g^n - \rho_g^n\right|^2}{N_g}}, \quad \text{and} \quad \text{error}\,(\nabla\cdot\mathbf{B}) = \frac{1}{N_g}\sum_g \frac{\left|\nabla\cdot\mathbf{B}_g^n\right|\Delta V_g}{\sum\left|\mathbf{B}_g^n\right|\cdot\Delta\mathbf{A}}. \tag{36}$$

$\langle\delta\rho\rangle_{rms}$ is the root-mean-square (rms) value of the error in Gauss's law on the grid. The error measurement for the $\nabla\cdot\mathbf{B}$ constraint given in Eq. (36) is a grid-averaged value of a local, positive-definite, non-dimensional measure of $\nabla\cdot\mathbf{B}$. The time evolution of these error quantities (with $\langle\delta\rho\rangle_{rms}$ normalized by en_0) are shown in the right panel of Fig. 1. The amplitudes for these values are relatively small on this time scale, being on the order of 10^{-13} after 3×10^4 plasma periods, and grow in time like $\sqrt{\omega_{pe}t}$. A random-walk behavior for these quantities is expected since both $\delta\rho_g^n = \nabla\cdot\epsilon_0\mathbf{E}_g^n - \rho_g^n$ and $\nabla\cdot\mathbf{B}_g^n$ are randomly fluctuating quantities with zero mean values.

6. 1D simulations of the dynamic pinch in a planar geometry

The dynamic compression of a plasma to a high energy density state is a phenomenon that is typical of both DZP and ICF experiments. Some differences are the compression mechanism and the geometry. While laser or x-ray radiation drives the compression of spherical targets in laser-driven ICF experiments, the compression is driven by a magnetic-piston via an axial current in a cylindrical geometry in DZP experiments. As a simple surrogate to quantify an algorithm's ability to study the dynamic compression of a plasma accurately and efficiently, we consider the magnetic-piston driven compression of a plasma slab carrying a constant current in its outer edge in 1D Cartesian geometry. This problem is the planar analog of the dynamic pinch [35,36].

The simulation setup is as follows. We consider a fully ionized deuterium plasma with initial temperatures $T_e = T_i = T_0 = 1\,\text{eV}$ and initial densities $n_e = n_i = n_0 = 1.0\times10^{17}/\text{cm}^3$. The plasma slab is bounded between $x=0$ and $x = R_{p0} = 1.5\,\text{cm}$ with homogeneous profiles. The lower boundary at $x=0$ is a symmetry plane. The compression is driven by the $J_z\times B_y \sim I^2$ force associated with a current I contained at the upper boundary of the plasma. The plasma current is realized by specifying a time profile for B_y at the upper boundary of the simulation domain, which is at $x=1.54\,\text{cm}$. This value is sufficiently larger than R_{p0} to ensure that no particles leave the domain. The boundary magnetic field rises linearly from 0 to a peak value of $B_0 = 2.667\,\text{T}$ over 8.154 ns and remains constant for later times. The value of B_0 corresponds to that at the outer edge of a 1.5 cm radius plasma column carrying 200 kA of current, as was considered in Parameter set 4 of Ref. [8].

The 1D dynamic pinch has three stages [36]. The first is the initial compression (i.e., run-in) stage, where the magnetic piston drives a shockwave towards the axis that outruns the piston. The next stage is the reflected shock stage, which begins when the incoming shock front reaches the axis and ends at maximum compression (i.e., stagnation) of the plasma column. The final stage is the expansion stage, where the plasma expands outwards as its pressure at stagnation exceeds the magnetic pressure. Estimates of the plasma conditions during the run-in stage can be made using the Rankine-Hugoniot jump conditions for an ideal gas with adiabatic coefficient $\gamma = 5/3$. The ratio of the magnetic pressure, $P_{B0} = B_0^2/2/\mu_0$, to the upstream plasma pressure, $P_0 = 2n_0T_0$, is $P_{B0}/P_0 = 88.3$. This gives a factor of 3.84 for the density jump across the shock and a factor of 23 for the temperature jump. Estimates of the parameters at the time of stagnation are made using the $P_{B0}/P_0 \to \infty$ limit [36] and considering the skin layer to be thin, in which case the density jump is 10 with respect to the initial value and the plasma pressure is $6P_{B0}$, which gives $T_e \approx T_i \approx 53\,\text{eV}$. Using these values, estimates

Table 1
Characteristic physical parameters, spatial scales, and time scales for the dynamic pinch simulations.

	physical parameters			spatial scales [μm]			time scales [ps]		
	n [1/cm^3]	T [eV]	B_y [T]	λ_{De}	δ_e	λ_e	$1/\omega_{pe}$	τ_e	$1/\Omega_{ce}$
initial	1.0×10^{17}	1.0	0.0	2.4×10^{-2}	17	0.14	5.6×10^{-2}	0.34	0.0
compression	3.84×10^{17}	23.0	2.67	5.8×10^{-2}	8.6	20.0	2.9×10^{-2}	9.9	2.1
stagnation	10.0×10^{17}	53.0	2.67	5.4×10^{-2}	5.3	40.5	1.8×10^{-2}	13.3	2.1

of various electron spatial scales and time scales during these stages are given in Table 1. The spatial scales include the Debye length, $\lambda_{De} = V_e/\omega_{pe}$, the skin depth, $\delta_e = c/\omega_{pe}$, and the mean free path, $\lambda_e = V_e\tau_e$ with $V_e = \sqrt{T_e/m_e}$. The time scales include the plasma period, $1/\omega_{pe}$, the collision time, τ_e, and the cyclotron period, $1/\Omega_{ce}$. Examination of the values in Table 1 show the multi-scale nature of this problem. λ_{De} is on the order of 10 nanometers, δ_e and λ_e are on the order of 10 microns, while the plasma column varies between 0.15−1.5 cm. The shortest time scale is the plasma period, which is on the order of 10 femtoseconds and is seven orders of magnitude smaller than the implosion time, $\tau_{im} = R_{p0}/U_{im} = 163$ ns where $U_{im} = \sqrt{P_{B0}/\rho_0} = 9.2$ μm/s is the characteristic implosion speed.

One additional length scale not given in Table. 1 is the skin depth associated with the current layer. In the collisional limit ($\nu_e t \gg 1$) this is given as $\delta_{skin} = \delta_e\sqrt{\nu_e t}$. Using $t = \tau_{im}$, $\delta_{skin} \approx 0.1$ cm, which is not insignificant compared to the estimated size of the plasma column at this time, $R_{p0}/4 = 0.375$ cm. This parameter is important to mention because the estimates given in Table. 1 are based on an ideal piston where the skin layer is infinitesimally thin compared to the plasma column width. Thus, small deviations from these estimates are expected.

In the discussion below, an estimate of the plasma frequency corresponding to the initial density, $\omega_{pe0} = 18.0$ /ps, is used to normalize the simulation time step Δt (e.g., $\omega_{pe0}\Delta t = 18.0 \Rightarrow \Delta t = 1.0$ ps). In terms of the cyclotron frequency associated with the B_0, $\omega_{pe0}\Delta t = 38.3 \Rightarrow \Omega_{ce0}\Delta t = 1.0$. For all simulations, the particles each have equal weights and are initialized on the grid by uniformly spacing them in each cell for a given value of the initial number of particles per cell, N_{ppc}. The initial velocities are obtained by randomly sampling from a Maxwellian distribution. The temperature profiles shown are computed in the conventional way, which is the average of the variance of the distribution function in each of the three velocity space directions. Unless stated otherwise, the relative tolerance for the Newton solver, Picard solver, and GMRES are set to 10^{-8}, 10^{-10}, and 10^{-4}, respectively.

6.1. Dynamic pinch simulations: numerical parameter study

Results from a spatial resolution scan using $N_{ppc} = 1000$ and $\omega_{pe0}\Delta t = 18$ are presented in the top row of Fig. 2. Here, spatial profiles of the plasma density and species temperatures at a time during the run-in stage for different grid resolutions are shown. The number of grid points is varied from $N_x = 56$ to $N_x = 432$. The corresponding cell spacing range is $\Delta x = 275$ μm to $\Delta x = 35.6$ μm. The spatial resolution is set by the need to resolve the non-local ion and electron heat layers at the shock front. The grid is too coarse using $N_x = 56$, but diminishing returns are observed for $N_x \geq 108$. $N_x = 108$ corresponds to $\Delta x = 143$ μm, which gives about 8-9 cells across the analytic estimate for the non-local electron heat layer width, $\sqrt{M_i/m_e}\lambda_e = 1240$ μm [9].

Results from a time step scan using $N_{ppc} = 1000$ and $N_x = 216$ are presented in the bottom row of Fig. 2. The time step is varied from $\omega_{pe0}\Delta t = 9$ to $\omega_{pe0}\Delta t = 72$. Recall that $\Omega_{ce0}\Delta t = 1$ for $\omega_{pe0}\Delta t = 38.3$; the electron cyclotron motion at the outer edge of the plasma is not resolved for the two largest time steps considered. Moreover, the characteristic electron mean free time is $\tau_e = 0.01$ ns, and thus electron scattering is also not well resolved for time steps in this range. Despite this, only small deviations are observed for time steps as large as $\omega_{pe0}\Delta t = 72$. The mass matrix used for these simulations is such that a particle is allowed to cross two cells in a time step. Time steps larger than $\omega_{pe0}\Delta t = 72$ are not considered because some electrons in the tail of the distribution would violate this constraint.

The simulation results shown in Fig. 2 are obtained using $N_{ppc} = 1000$. However, quantitatively accurate results can also be obtained using values for N_{ppc} as low as 100. Spatial profiles of the plasma density and species temperatures at a time during the run-in and at the time of stagnation are shown in the top row of Fig. 3 using $N_{ppc} = 100$ and $N_{ppc} = 1000$. While the results from the lower particle resolution simulations have more noise, they are still in excellent agreement with the results from the higher particle resolution simulations. Time profiles of the energy partition from these simulations starting at $t = 0$ and going through the time of stagnation and into the expansion stage ($t_{max} = 220$ ns) are shown in the bottom row of Fig. 3. Again, the results from the low-particle resolution simulations are in excellent agreement with those from the high-particle resolution simulations. Time traces of the measure of energy conservation and charge conservation from these simulations are also shown in this figure. The conservation of energy is measured by computing the change of total energy in the system and subtracting the time-integrated Poynting flux associated with the magnetic piston coming in from the upper boundary of the simulation domain. Charge conservation is measured using the expression for $< \delta\rho >_{rms}$ from Eq. (36). Global energy and local charge density are both conserved well during the entire simulation using both $N_{ppc} = 100$ and $N_{ppc} = 1000$.

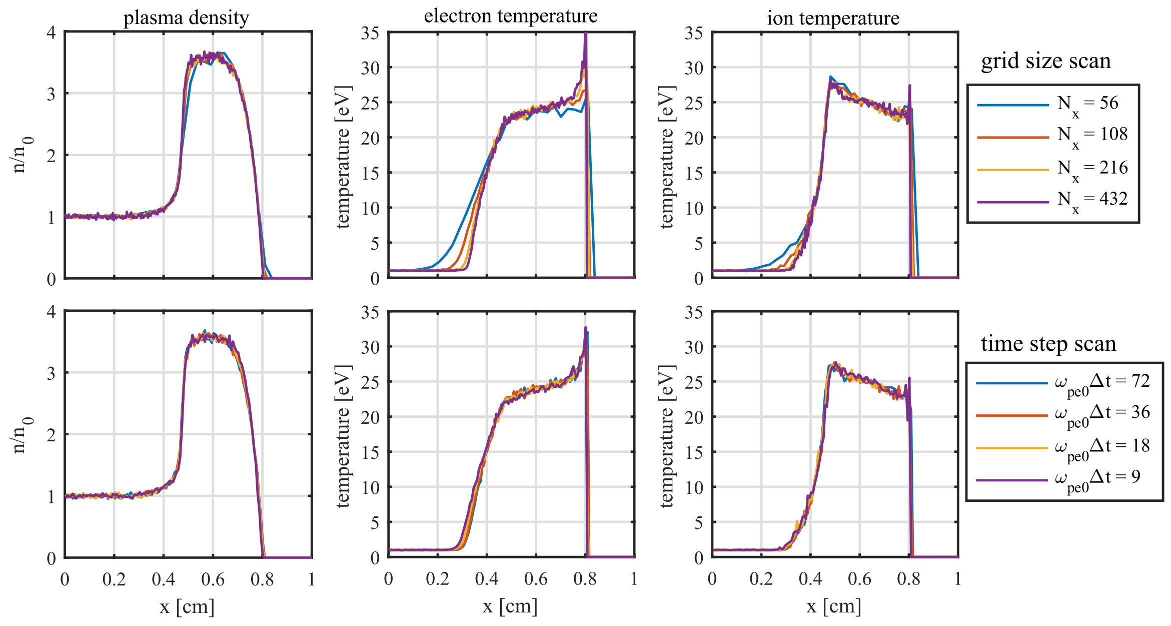

Fig. 2. Spatial profiles of the plasma density (left panels), electron temperature (middle panels) and ion temperature (right panels) at a time during the run-in stage ($t = 101.2$ ns) from dynamic pinch simulations for different grid resolutions (top row) and different time steps (bottom row). The grid resolution scan is performed using $\omega_{pe0}\Delta t = 18$ and the time step scan is performed using $N_x = 216$. $N_{ppc} = 1000$ is used for both scans.

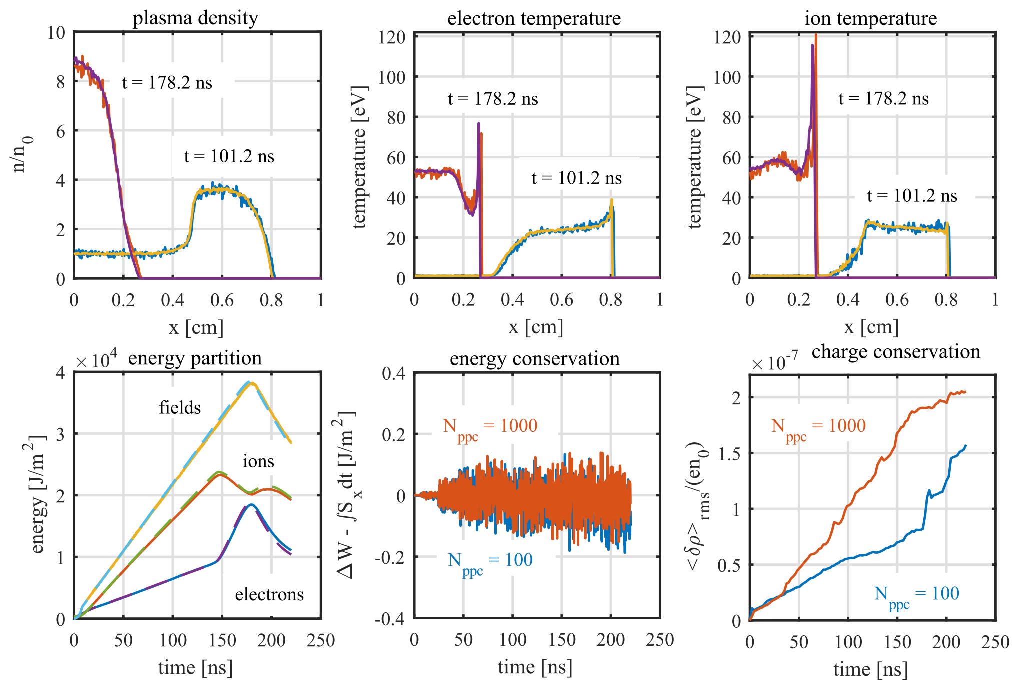

Fig. 3. Results from dynamic pinch simulations using a moderate and high number of particles per cell using $N_x = 432$ and $\omega_{pe0}\Delta t = 18$. In the top row, the noisy curves are obtained using $N_{ppc} = 100$ and the smooth ones are obtained using $N_{ppc} = 1000$. Top row: Spatial profiles of the plasma density (left panel), electron temperature (middle panel) and ion temperature (right panel) at a time during the implosion ($t = 101.2$ ns) and at the time of stagnation ($t = 178.2$ ns). Bottom row: Time traces of the energy in the fields and species (left panel), energy conservation (middle panel), and charge conservation (right panel). The dashed curves in the bottom left panel correspond to $N_{ppc} = 1000$.

6.2. Dynamic pinch simulations: solver performance

The performance of the PS-JFNK method for the dynamic pinch simulations is quantified by looking at the number of Newton iterations per time step, the number of GMRES iterations per Newton iteration, and the number of Picard iterations

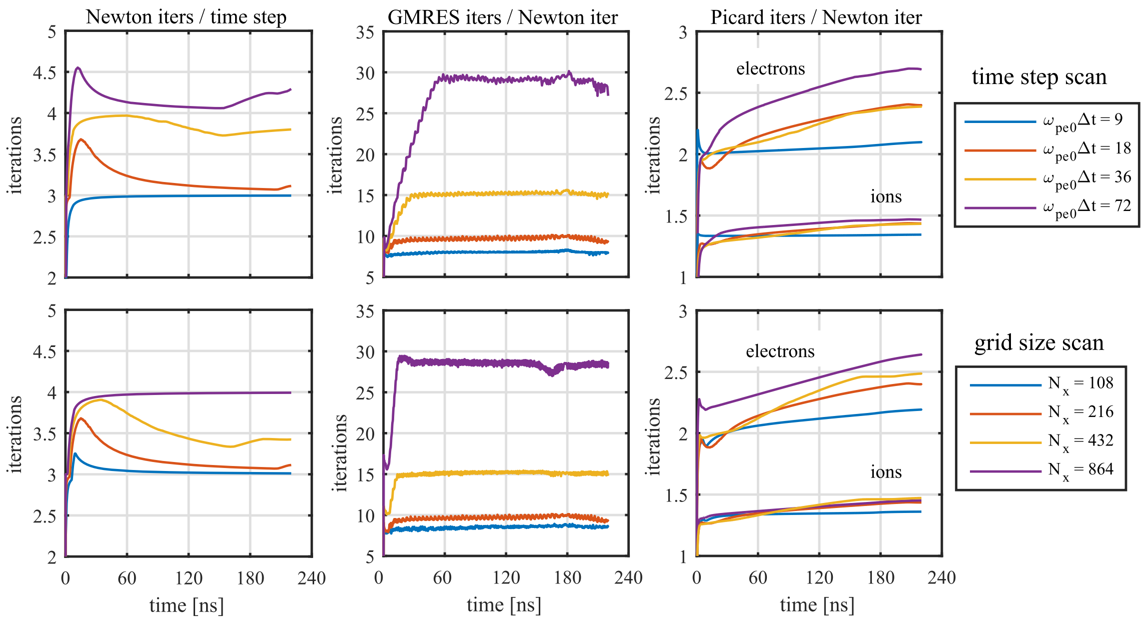

Fig. 4. Time profiles of the PS-JFNK solver performance from dynamic pinch simulations for different times steps using $N_x = 216$ (top row) and for different grid resolutions using $\omega_{pe0}\Delta t = 18$ (bottom row). Each of these simulations uses $N_{ppc} = 1000$. The running average number of Newton iterations per time step is shown in the left panels. The number of linear iterations per Newton iteration (averaged over 200 step) is shown in the middle panels. The running average number of Picard iterations per Newton iteration for both electrons and ions is shown in the right panels.

per Newton iteration. These quantities are shown in Fig. 4 for varying values of the simulation time step and the number of grid points using $N_{ppc} = 1000$. For both the time step and grid resolution scan, lowering the particle resolution by 10× does not significantly change the results shown. The running time-average number of Newton iterations per time step is shown in the left panels of Fig. 4. This value varies modestly from 3 and 4.5 for $\omega_{pe0}\Delta t$ going from 9 to 72 at $N_x = 216$, and varies between 3 and 4 for N_x going from 56 to 864 at $\omega_{pe0}\Delta t = 18$. The total number of Newton iterations taken to reach a certain point in time for these simulations is found by multiplying the values shown in these figures at that time by the total number of steps taken to reach that time (i.e., running time-average). Similarly, the running time-average number of Picard iterations per Newton iteration for both electrons and ions is shown in the right panels of Fig. 4. For all parameters considered, this value for the ions is a little more than 1.5 and for the electrons it is between 2 and about 2.5.

The number of GMRES iterations per Newton iteration is shown in the middle panels of Fig. 4. These values are averaged over 200 time steps. This number appears to depend on the CFL number, $c\Delta t/\Delta x$. For the results shown in the top row of Fig. 4, $c\Delta t/\Delta x = 2.1, 4.2, 8.4, 16.8$ for $\omega_{pe0}\Delta t = 9, 18, 36, 72$, respectively. The corresponding colors in the bottom row of Fig, 4 have the exact same CFL values as those in the top row. The number of GMRES iterations per Newton iteration is between 6 and 8 for $c\Delta t/\Delta x \leq 4$ and increases approximately linearly with the CFL number for larger values. Since light waves are captured in the preconditioner, what we believe is going on here is a CFL constraint associated with high energy electron particles in the tail of the distribution function. The off-diagonal terms of the mass matrix become relatively larger with respect to the diagonal term as $\Delta t/\Delta x$ increases. Since we only include the diagonal term of the mass matrix in the preconditioner, the increased number of GMRES iterations with this value is expected. However, it is important to re-emphasize that the GMRES iterations are purely a grid-based operation in our implementation of the PS-JFNK method (**Algorithm** 2). So, while the GMRES iterations are increasing, computational efficiency in terms of wall time can still be achieved when using many particles per cell, $N_p \gg N_g$, in which case grid-based operations are computationally cheap relative to particle operations. Nevertheless, a linear increase in the number of GMRES iterations with CFL number is not ideal and a more sophisticated preconditioner will be considered in future work.

The dependence of the wall time associated with the PS-JFNK solve with the time step is shown in Fig. 5 using a grid with $N_x = 216$. The wall times shown in this figure correspond to those needed to reach a physical time of 220 ns. These simulations were done on the Ruby cluster at LLNL. Each node on Ruby has 2 28-core Intel Xeon CLX-8276L processors (2.2 GHz). The algorithm is written in C++/Fortran using the Chombo framework [32] and uses a standard grid-based MPI decomposition. For the results in Fig. 5, 54 processors are used with 4 grid cells per processor. There may be better ways to handle the particles using MPI for this problem, as all the particles get swept into a small region during the compression, but the point here is to show the decrease in wall time with increasing time step and not to present the most efficient possible utilization of MPI.

The results in Fig. 5 show an improvement with wall time for the full range of Δt considered here, $0.9 \leq \omega_{pe0}\Delta t \leq 72$, for both $N_{ppc} = 100$ and $N_{ppc} = 1000$. The scaling is close to ideal ($\sim 1/\Delta t$) in the intermediate range, $4 \leq \omega_{pe0}\Delta t \leq 20$. The non-ideal behavior when going from $\omega_{pe0}\Delta t = 0.9$ to $\omega_{pe0}\Delta t = 4$ is associated with an increase in Newton iterations

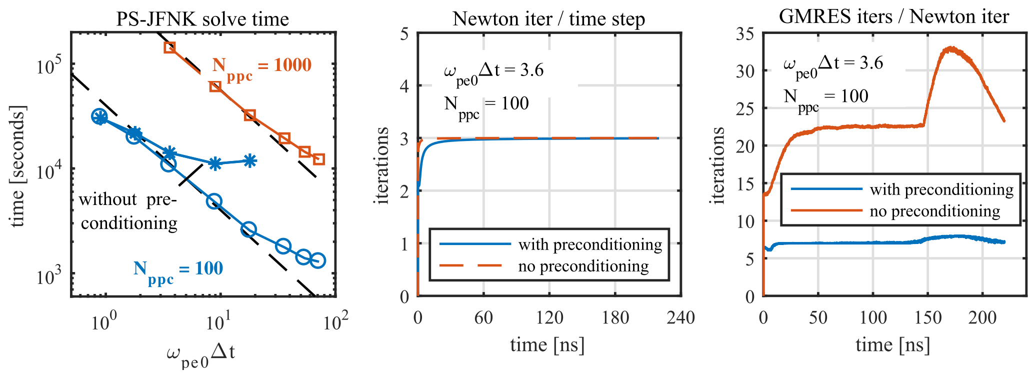

Fig. 5. Solver wall time (steps 1-10 of **Algorithm** 1) scaling with time step for dynamic pinch simulations using $N_x = 216$ (left panel). The times presented are that needed to reach 220 ns. The blue circles are for $N_{ppc} = 100$ and the red squares are for $N_{ppc} = 1000$. The blue asterisks are for $N_{ppc} = 100$ without using a preconditioner. The lower and upper dashed black lines scale with $1/\omega_{pe0}\Delta t$. The coefficient is 4.0×10^4 for the lower one and is 5.6×10^5 for the upper one. Time profiles of the running average number of Newton iterations per time step and the 200 time-step average number of GMRES iterations per Newton iteration for $\omega_{pe0}\Delta t = 3.6$ and $N_{ppc} = 100$, with and without preconditioning, are shown in the middle and right panels, respectively.

($2 \rightarrow 3$). The non-ideal behavior at the largest values of $\omega_{pe0}\Delta t$ is due to the relative increase in the cost associated with the increase in the number of GMRES iterations.

The wall time associated with the PS-JFNK solve when not using preconditioning is also shown in Fig. 5 using $N_x = 216$ and $N_{ppc} = 100$. For these parameters, preconditioning is essential for wall time improvements for $\omega_{pe0}\Delta t \geq 0.9$. The running-average number of Newton iterations per time step and the 200 time-step average number of GMRES iterations per Newton iteration for $\omega_{pe0}\Delta t = 3.6$ and $N_{ppc} = 100$, with and without preconditioning, are shown in the middle and right panels of Fig. 5. Preconditioning has little-to-no effect on the number of Newton iterations, as expected. But it has a large effect on the number of GMRES iterations. Without preconditioning, the number of GMRES iterations scales with the condition number of the matrix, which here is the maximum value of $\omega_{pe}\Delta t$ on the grid, which in turn scales with the square root of the maximum plasma density. The effectiveness of the preconditioner used in this work is seen in the right panel of Fig. 5 where the number of GMRES iterations per Newton iteration remains roughly constant as the plasma compresses.

6.3. *Dynamic pinch simulations: effects of damping*

Compared to the leap-frog method for **E** and **B**, aliasing errors are relatively large for poorly resolved modes in the fully implicit PIC method [12,26]. Damping of high-k modes is often needed to mitigate potential issues arising from interactions with these unphysically dispersive modes, such as numerical Cerenkov radiation. Damping is introduced by using $\theta > 1/2$. However, this comes at the expense of *exact* energy conservation and results in the plasma artificially cooling over time (see Eq. (19)). The effects of damping on the dynamic pinch simulations are shown in Fig. 6. Even a modest change to θ from the *exact* energy-conserving value, $\theta = 0.5 \rightarrow 0.501$, is seen to cause a significant effect on the physical accuracy. The effects of damping on the physical accuracy are mitigated as the number of particles per cell is increased [14] but note that the damping is cumulative in time. The ability to conserve energy *exactly* in the fully implicit PIC method is a significant advantage over conventional explicit and semi-implicit methods for long time scale simulations of collisional plasmas, where enhanced numerical heating [5] and/or cooling [8] can be a problem.

7. A predictor-corrector method for solving the implicit θ-PIC-MCC equations

There is often a trade-off between physical accuracy and computational efficiency when choosing between different numerical schemes. Two of the central features of **Algorithm** 1 are its ability to *exactly* conserve global energy and local charge. The importance of *exact* energy conservation for simulating the dynamic compression of collisional plasma is illustrated in Sec. 6.3. Assuming that *exact* local charge conservation is not critical, we show in this section how one can reformulate **Algorithm** 1 as a two-step predictor-corrector method that still *exactly* conserves global energy, but only approximately conserves local charge. The advantage is that this algorithm has a computational efficiency that is within a factor of two of the ECSIM PIC method [19]. Before presenting this algorithm, it is instructive to first show how the modified PS-JFNK method outline in **Algorithm** 1 can be viewed as a modified fixed-point iteration method for the particle positions.

7.1. *Reformulation of modified PS-JFNK as a modified fixed-point iteration method*

The PS-JFNK method presented in **Algorithm** 1 is considered to be modified because a modified system Jacobian is used to obtain the Newton correction to the fields. These modifications, which are that the particle positions and the magnetic field used in the Lorentz force are held fixed when computing the linearized current density, are motivated by the fact that

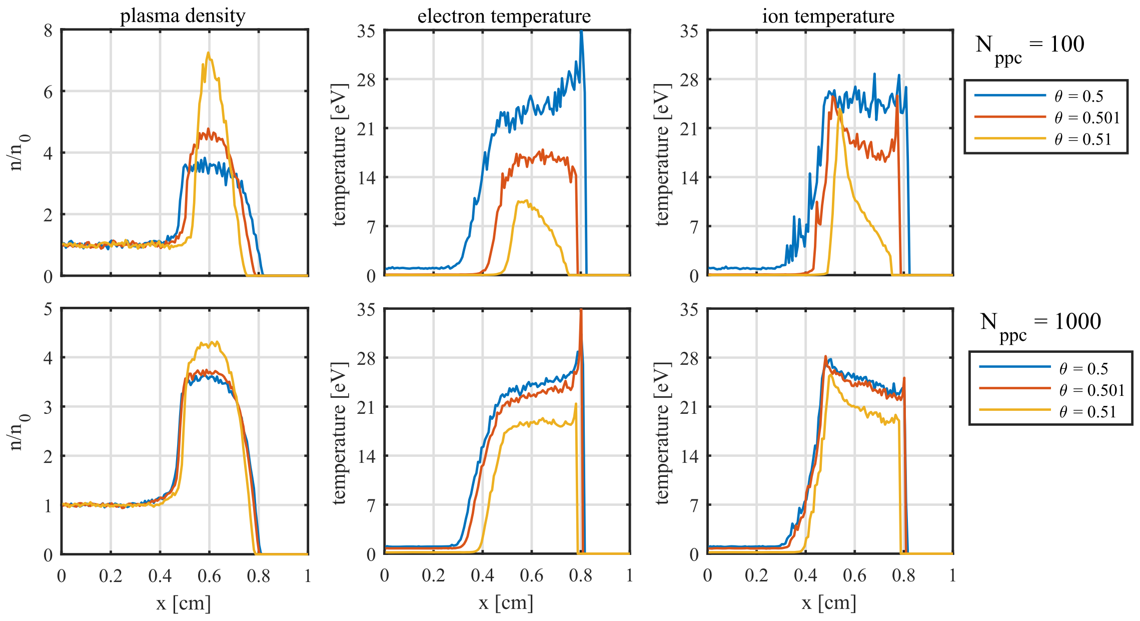

Fig. 6. Spatial profiles at $t = 101.2\,\text{ns}$ from 1D dynamic pinch simulations for different values of the time-biasing parameter θ to illustrate the effects of damping. $N_x = 216$ and $\omega_{pe0}\Delta t = 18$ are used for these simulations. The results in the top row are obtained using a low particle resolution, $N_{ppc} = 100$. Those in the bottom row are obtained using $N_{ppc} = 1000$.

neither of these assumptions affect the global energy conservation and they permit the ability to directly compute $\bar{\mathbf{J}}_g$ during the linear iterations via a grid-based operation rather than a particle-based operation. The system residual is not altered since these quantities that are held fixed during the linear stage are updated at each Newton iteration.

In the PS-JFNK method, the particle quantities are nonlinearly eliminated from the field equations. However, mathematically speaking, a closer inspection of the method outlined in **Algorithm** 1 reveals that this approach can be viewed as a modified fixed-point iteration method for the particle positions, where the fields are nonlinearly eliminated from the particle equations. A fixed-point iteration method is used to solve equations of the form $\mathbf{y} = \mathbf{f}(\mathbf{y})$. Starting with an initial guess $\mathbf{y} = \mathbf{y}^{(0)}$, the method proceeds as

$$\mathbf{y}^{(m+1)} = \mathbf{f}\left(\mathbf{y}^{(m)}\right), \quad \text{for } m = 0, 1, \ldots \tag{37}$$

until a certain convergence criterion is met. Similar to how one writes $\mathbf{X}_p = \mathbf{f}_p\left(\mathbf{X}_f\right)$ in the PS-JFNK method to obtain Eq. (28), the particle positions can be written as $\bar{\mathbf{x}}_p = \mathbf{g}_p\left(\mathbf{X}_f\right)$. Additionally, one can also write $\mathbf{X}_f = \mathbf{f}_f\left(\bar{\mathbf{x}}_p\right)$, in which case $\bar{\mathbf{x}}_p = \mathbf{g}_p\left(\mathbf{f}_p\left(\bar{\mathbf{x}}_p\right)\right) = \mathbf{h}_p\left(\bar{\mathbf{x}}_p\right)$. In other words, given the particle positions, the electromagnetic fields on the grid can readily be obtained. This is because the equations governing the fields are weakly nonlinear for fixed particle positions. The only source of nonlinearity comes through the magnetic field in the Lorentz force, which is also held fixed for the field solve in **Algorithm** 1. Using a fixed magnetic field in the Lorentz force during the field solve is what makes **Algorithm** 1 a *modified* fixed-point iteration method for the particle positions.

7.2. The predictor-corrector algorithm

With the concept of viewing **Algorithm** 1 as a modified fixed-point iteration method for the particle positions, we can reformulate this algorithm as a predictor-corrector method. This reformulation is outlined in **Algorithm** 3. This method involves two GMRES solves, one for the predictor stage and one for the corrector stage, and one call to **Algorithm** 2 for a self-consistent update of the particle quantities. This algorithm is efficient, second order accurate in time with respect to both particle positions and the Lorentz force, and conserves global energy *exactly*, but only approximately conserves local charge density.

Note that $\bar{\mathbf{v}}_p$ in step 7 of **Algorithm** 3 is computed using the magnetic field and the particle position from the predictor stage; consistent with what is used for computing the mass matrix quantities used in step 6. This is required to maintain *exact* energy conservation, which demands that $\bar{\mathbf{v}}_p$ be consistent with that used to compute $\bar{\mathbf{J}}_g$ when advancing the fields. However, this algorithm does not *exactly* conserve local charge density, which requires that $\bar{\mathbf{v}}_p$ used in the field solve to be consistent with the change in particle positions. When using a charge-conserving current deposition, such as the CC1 scheme considered in this work, local charge density can be expected to be approximately conserved. One could apply the

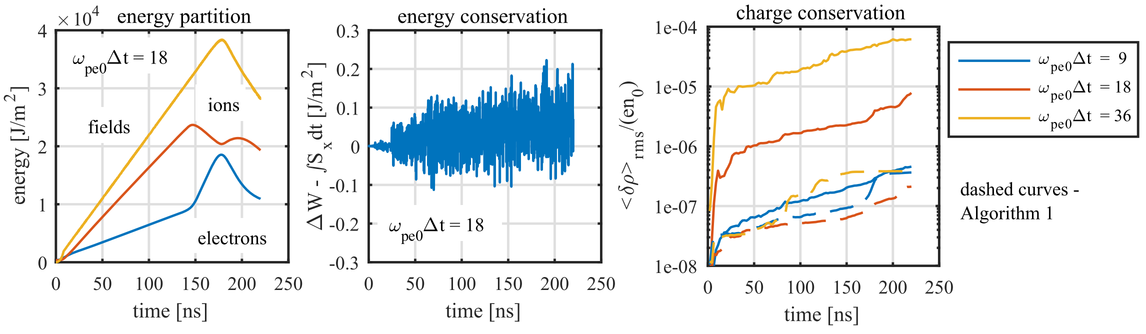

Fig. 7. Time profiles of energy partition (left panel), energy conservation (middle panel), and charge conservation (right panel) from dynamic pinch simulations using the predictor-corrector method (**Algorithm** 3). The results in the right panel are compared with those obtained using the fully implicit solution using **Algorithm** 1 (depicted with dashed curves). Each of these simulations uses $N_x = 216$ and $N_{ppc} = 1000$.

techniques from Ref. [29] to adjust the particle positions after the time advance in a such a way as to preserve local charge density, but that is not attempted here.

Algorithm 3 Predictor-corrector method for advancing the θ-PIC-MCC system from t_n to $t_{n+1} = t_n + \Delta t$.

```
   Predictor steps:
1:    Initial half time step advance of particle positions: $\bar{\mathbf{x}}_p^{(0)} = \mathbf{x}_p^n + \mathbf{v}_p^n \Delta t/2$.
2:    Compute mass matrix quantities, $\hat{\mathbf{J}}_g$ and $\mathbb{M}_{gg'}$, from Eqs. (33) using $\bar{\mathbf{x}}_p^{(0)}$ and $\mathbf{B}_g^{(0)} = \mathbf{B}_g^n$.
3:    With $\bar{\mathbf{J}}_g = \hat{\mathbf{J}}_g + \sum_{g'} \mathbb{M}_{gg'} \cdot \mathbf{E}_{g'}^{(1)}$, use GMRES to solve Eqs. (25) for $\mathbf{E}_g^{(1)}$ and $\mathbf{B}_g^{(1)}$.
4:    With $\mathbf{E}_g^{(1)}$, $\mathbf{B}_g^{(1)}$, and $\bar{\mathbf{x}}_p^{(0)}$, use Algorithm 2 to update the particle quantities, $\bar{\mathbf{x}}_p^{(1)}$ and $\bar{\mathbf{v}}_p^{(1)}$.
   Corrector steps:
5:    Compute mass matrix quantities, $\hat{\mathbf{J}}_g$ and $\mathbb{M}_{gg'}$, from Eqs. (33) using $\bar{\mathbf{x}}_p^{(1)}$ and $\mathbf{B}_g^{(1)}$.
6:    With $\bar{\mathbf{J}}_g = \hat{\mathbf{J}}_g + \sum_{g'} \mathbb{M}_{gg'} \cdot \mathbf{E}_{g'}^{n+\theta}$, use GMRES to solve Eqs. (25) for $\mathbf{E}_g^{n+\theta}$ and $\mathbf{B}_g^{n+\theta}$.
7:    With $\mathbf{E}_g^{n+\theta}$, $\mathbf{B}_g^{(1)}$, and $\bar{\mathbf{x}}_p^{(1)}$, use Eq. (29) to compute $\bar{\mathbf{v}}_p$, then set $\bar{\mathbf{x}}_p = \mathbf{x}_p^n + \bar{\mathbf{v}}_p \Delta t/2$.
8: Use Eq. (7) to transform the system variables from $t_{n+\theta}$ and $t_{n+1/2}$ to $t_{n+1}$.
9: Modify the particle velocities, $\mathbf{v}_p^{n+1}$, using the binary MCC algorithm for Coulomb collisions.
```

7.3. Dynamic pinch simulations with the predictor-corrector algorithm

The accuracy and efficiency of the predictor-corrector method given **Algorithm** 3 is examined in this section by using it to do the same dynamic pinch simulations presented previously using **Algorithm** 1 in Sec. 6. The relative tolerance for the GMRES solver is set to 10^{-6} for these simulations. Time profiles of the energy partition and energy conservation are shown in the left and middle panels, respectively, of Fig. 7 using $N_x = 216$, $N_{ppc} = 1000$, and $\omega_{pe0}\Delta t = 18$. These curves are in good agreement with the same results obtained using **Algorithm** 1 shown in the bottom left and bottom middle panels of Fig. 3 - illustrating accuracy and energy conservation with the predictor-corrector method. Time profiles for the measure of charge conservation (see Eq. (36)) from these simulations are shown in the right panel of Fig. 7 for several different time steps. These results show that local charge density is conserved well when using $\omega_{pe}\Delta t \leq 9$ but becomes increasingly poorly conserved as the time step increases. It is observed that the solution becomes inaccurate when using time steps larger than those considered for Fig. 7. The time step needed to maintain accuracy when using **Algorithm** 3 is about a factor of two smaller than that needed for **Algorithm** 1, where physically accurate results can be obtained using $\omega_{pe0}\Delta t = 72$. Furthermore, we have observed a similar lack of physical accuracy for $\omega_{pe0}\Delta t \geq 18$ when using **Algorithm** 1 with the non-charge-conserving CIC shape function. We speculate that local charge conservation may be important for maintaining physical accuracy when using large time steps.

While we are not able to use as large of a time step with **Algorithm** 3 compared to **Algorithm** 1 for the dynamic pinch simulations, the advantage is that the predictor-corrector algorithm is more computationally efficient for time steps where it does yield accurate results. The improvement in performance of this algorithm over **Algorithm** 1 is shown in Fig. 8 where the solver wall times as a function of $\omega_{pe0}\Delta t$ from **Algorithm** 3 are compared with those from **Algorithm** 1. For $N_{ppc} = 1000$, there is a 1.5–2.0 reduction in wall time for $3.6 \leq \omega_{pe0}\Delta t \leq 36$. For $N_{ppc} = 100$, there is a 1.2–1.8 reduction in wall time for $0.9 \leq \omega_{pe0}\Delta t \leq 36$. In fact, the solver wall time is slightly better when using **Algorithm** 3 with $\omega_{pe0}\Delta t = 36$ than it is when using **Algorithm** 1 with $\omega_{pe0}\Delta t = 72$. It may be advantageous to use such a method if performance is needed and rigorous local charge conservation is not necessary.

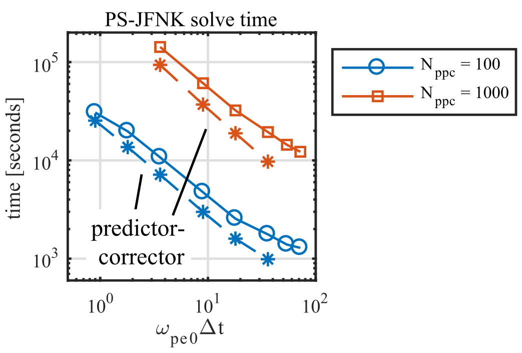

Fig. 8. Solver wall time scaling with time step for dynamic pinch simulations using $N_x = 216$. The times presented are that needed to reach 220 ns. The solid curves are obtained using the fully implicit method (**Algorithm** 1). The dashed star curves are obtained using the predictor-corrector method (**Algorithm** 3).

8. Further discussion and mathematical analysis of Algorithms 1 and 3

It is shown above in Sec. 7 how **Algorithm** 1 can be viewed as a modified fixed-point iteration method for the particle positions. Some computational aspects concerning the use of the mass matrices with a charge-conserving shape function are provided in Appendix B. A further discussion and mathematical analysis concerning the use of the mass matrices in **Algorithms** 1 and 3 is provided here. First, it is instructive to analyze the assumptions made to remove the need to re-advance the particles during the linear stage from the point of view of the standard PS-JFNK method. This is done by looking at how these modifications alter the system Jacobian. This is followed by a discussion on how this approach relates to fluid-based preconditioning methods and estimates of the computational cost associated with computing/using the mass matrices.

8.1. Mathematical analysis of the modified system Jacobian

In GMRES, one needs the action of the system Jacobian on some given field system vector $\mathbf{V}_f$. The only nonlinear term in the system Jacobian comes from the current density. At each GMRES iteration during Newton iteration $nl+1$, one needs to compute

$$\bar{\mathbf{J}}_g'\left(\mathbf{X}_f^{(nl)}\right)\mathbf{V}_f = \frac{\bar{\mathbf{J}}_g\left(\mathbf{X}_f^{(nl)}+\epsilon\mathbf{V}_f\right)-\bar{\mathbf{J}}_g\left(\mathbf{X}_f^{(nl)}\right)}{\epsilon}, \tag{38}$$

where $\mathbf{X}_f^{(nl)} = \left[\mathbf{E}_g^{(nl)}, \mathbf{B}_g^{(nl)}\right]$ is the field solution from Newton iteration nl and $\epsilon\mathbf{V}_f = \left[\delta\mathbf{E}_g, \delta\mathbf{B}_g\right]$ is a small correction. (The field corrections here are not to be confused with the Newton correction for the fields.) Considering Eq. (11) and using a Taylor series expansion, the linearized current density can be written as

$$\bar{\mathbf{J}}_g\left(\mathbf{X}_f^{(nl)}+\epsilon\mathbf{V}_f\right) \approx \bar{\mathbf{J}}_g\left(\mathbf{X}_f^{(nl)}\right) + \sum_p \frac{q_p}{\Delta V_g}\left[\delta\bar{\mathbf{v}}_p S_{gp}^{\mathbf{J}} + \bar{\mathbf{v}}_p^{(nl)}\frac{\partial S_{gp}^{\mathbf{J}}}{\partial\bar{\mathbf{x}}_p}\cdot\delta\bar{\mathbf{x}}_p\right]. \tag{39}$$

The perturbed particle values are

$$\delta\bar{\mathbf{x}}_p = \frac{\delta\bar{\mathbf{v}}_p\Delta t}{2} \quad\text{and}\quad \delta\bar{\mathbf{v}}_p = \frac{\Delta t}{2}\frac{q_s}{m_s}\left(\delta\mathbf{E}_p + \delta\bar{\mathbf{v}}_p\times\mathbf{B}_p^{(nl)} + \bar{\mathbf{v}}_p^{(nl)}\times\delta\mathbf{B}_p\right), \tag{40}$$

and the perturbed field quantities at the particles are

$$\delta\mathbf{E}_p = \sum_g\left[\delta\mathbf{E}_g S_{gp}^{\mathbf{E}} + \mathbf{E}_g^{(nl)}\frac{\partial S_{gp}^{\mathbf{E}}}{\partial\bar{\mathbf{x}}_p}\cdot\delta\bar{\mathbf{x}}_p\right], \quad\text{and}\quad \delta\mathbf{B}_p = \sum_g\left[\delta\mathbf{B}_g S_{gp}^{\mathbf{B}} + \mathbf{B}_g^{(nl)}\frac{\partial S_{gp}^{\mathbf{B}}}{\partial\bar{\mathbf{x}}_p}\cdot\delta\bar{\mathbf{x}}_p\right]. \tag{41}$$

The shape functions and their derivatives in Eqs. (39) and (41) are computed using the particle positions from the previous Newton iteration, $\bar{\mathbf{x}}_p^{(nl)} = \mathbf{x}_p^n + \bar{\mathbf{v}}_p^{(nl)}\Delta t/2$.

Holding the magnetic field in the Lorentz force fixed during the linear stage amounts to neglecting the last term in the expression for $\delta\bar{\mathbf{v}}_p$ in Eq. (40), in which case $\delta\mathbf{B}_g$ does not enter the perturbed current density. A rigorous mathematical explanation to justify ignoring this term is not given here. Instead, we emphasize that the magnetic field used in the Lorentz force does not affect global energy conservation and it does not affect local charge conservation. In other words, the field-particle system will converge to an energy- and charge-conserving solution with any reasonable choice for $\mathbf{B}_g$ in the Lorentz force. The magnetic field from the previous time step, $\mathbf{B}_g^n$, is used in the ECSIM methods [19]. The value from the previous Newton iteration is used in this work. We have not experienced any solver convergence issues when doing this, but should they arise then one could simply stop updating the magnetic field used in the Lorentz force. Any value of $\mathbf{B}_g$ after the first Newton iteration should be second order accurate.

Holding the particle positions fixed during the linear stage amounts to neglecting the terms in Eq. (39) and Eq. (41) that depend on the derivative of the shape functions. Magnetic field perturbations are neglected for an analysis of this assumption, in which case $\delta\bar{\mathbf{v}}_p = \alpha_s \mathbb{R}_p \delta\mathbf{E}_p$, where $\mathbb{R}_p$ is the magnetic field rotation matrix (Eq. (32)). The linearized current density and mean particle velocity can be expressed as

$$\delta\bar{\mathbf{J}}_g = \sum_p \frac{q_p}{\Delta V_g}\left[\delta\bar{\mathbf{v}}_p S_{gp}^{\mathbf{J}} + \frac{\Delta t}{2}\bar{\mathbf{v}}_p^{(nl)}\frac{\partial S_{gp}^{\mathbf{J}}}{\partial \bar{\mathbf{x}}_p}\cdot\delta\bar{\mathbf{v}}_p\right], \tag{42}$$

$$\delta\bar{\mathbf{v}}_p = \alpha_s \sum_g \left[\mathbb{R}_p\delta\mathbf{E}_g S_{gp}^{\mathbf{E}} + \frac{\Delta t}{2}\mathbb{R}_p\mathbf{E}_g^{(nl)}\frac{\partial S_{gp}^{\mathbf{E}}}{\partial \bar{\mathbf{x}}_p}\cdot\delta\bar{\mathbf{v}}_p\right]. \tag{43}$$

The recursive nature of Eqs. (42)-(43) with respect to $\delta\bar{\mathbf{v}}_p$ makes it difficult to formulate a closed form expression for the general multi-dimensional case. However, we can make semi-quantitative estimates of when the latter terms in these two expressions can be neglected. For a shape function of order n, call it S_{gp}^n, the derivative in direction i scales approximately as $\partial S_{gp}^n/\partial\bar{x}_p \sim S_{gp}^{n-1}/\Delta x_i$. It is readily shown in 1D that the last term in Eq. (42) has a magnitude that scales with $\bar{v}_{p,x}^{(nl)}\Delta t/\Delta x/2$ compared to the first term. This suggests that holding the particle positions fixed during the linear stage has little effect on the system Jacobian when most of the particles contributing to the current density are Courant limited. An analysis of Eq. (43) yields a similar result when comparing the last term in the sum over grid points directly with $\delta\bar{\mathbf{v}}_p$.

For the multi-dimensional case, the full vector nature of $\delta\bar{\mathbf{v}}_p$ needs to be considered, but one will similarly obtain a Courant-like condition. However, even if the last terms in Eqs. (42)-(43) are not strictly negligible, this doesn't imply that the method will fail. It just means that capturing the nonlinearities of the system with respect to the particle positions is left to the Picard solver at the Newton stage instead of through a nonlinear elimination process during the linear stage. Additionally, since the terms involving the change in particle positions scale with the derivative of the particle shape factor, which is antisymmetric about the particle positions, there can be multiple cancelling terms when integrating over the particles.

From a practical implementation point of view, it is worth commenting that it is possible that the linearized contribution to the current density arising from the linearized particle position may not be captured correctly in the conventional implementation of the PS-JFNK method. This is because both ϵ and $\mathbf{V}_f$ in Eq. (38) are small parameters, making the product even smaller. Recall that the self-consistent particle update is done using an iterative method with some relative tolerance. It is certainly possible that the value of $\epsilon\mathbf{V}_f$ is smaller than the relative tolerance for the iterative particle solver, in which case linearized corrections to the particles associated with the linearized particle positions may not be captured.

8.2. Comparing with fluid-based preconditioning methods

A discussion on the memory requirements for using the mass matrices with a charge-conserving shape function is given in Appendix B. There are several other aspects of the modified PS-JFNK method presented here concerning the use of the mass matrices that merit further discussion. First off, it should be emphasized that using the mass matrices in **Algorithms** 1 and 3 is not mandatory. One could instead choose to compute the current density directly from the particles during the linear iterations. The modifications to the particle advance made to use the mass matrices could still be advantageous since the linear system to be solved is simplified and an iterative method is not required when re-advancing the particles - only their velocities need to be updated. For the 1D and 2D problems considered here, the computational overhead associated with computing the mass matrices at each Newton iteration is far less than that associated with re-advancing the particles at each GMRES iteration, resulting in wall time reductions by factors on the order of 10.

Whether using the mass matrices results in a reduction of wall time depends on many things, such as the number of particles per cell, the number of spatial dimensions, and how many iterations are needed for GMRES to converge. Fluid-based preconditioning has been successfully used in fully implicit electrostatic [17] and Darwin electromagnetic [18] PIC models to reduce the number of GMRES iterations to values as low as 0—3 per Newton iteration on average when applied to certain test problems. The value of zero is made possible by using the solution of the preconditioned system as an initial guess for GMRES [18] and using 10^{-2} for the relative tolerance. For situations where GMRES converges with only 0—3 iterations, using the mass matrices may not lead to a significant improvement in the wall time. For the fully electromagnetic equations considered in this paper, the number of GMRES iterations per Newton iteration is consistently between 8—30 when using the diagonal elements of the mass matrices for the preconditioner.

Strictly speaking, since the method considered here uses a modified Jacobian to compute the Newton corrections, one could say that we are just solving a preconditioner system (i.e., it is a quasi-Newton method). Similar to what is done in Refs. [17,18], one could use this linear solution as an initial guess for the linear system with the full system Jacobian. But then the method involves using GMRES inside of GMRES and preconditioning the preconditioner - the algorithmic complexity is increased significantly. The net result may be a reduction of one or two Newton iterations for large time steps at the expense of one or two extra linear iterations with the full system Jacobian. However, since the residual function evaluations at these linear iterations are just as expensive as those at a Newton iteration, the work spent doing particle-based operations is just shifted from one place to another and may not result in any net computational savings.

The computational expense and memory associated with computing and storing the mass matrices is not an issue in 1D and 2D, but it may be in 3D (See Appendix B). If that is the case, then one could use a further approximate form of the system Jacobian that has a reduced size and hence cost. For example, one could use a truncated form of the exact mass matrices that neglects the elements furthest from the grid point where the current density is being computed, or one could use the mass matrix computed using a reduced-order shape function (i.e., CIC). These approaches would reduce the number of mass matrix elements by a factor of 3 to 6 in 3D. The drawback of this approach would most likely come through a limit on the time step for accuracy. It is worth commenting that an approximation to the mass matrices is used in place of the fluid-based preconditioner for the Darwin system considered in Ref. [18] in situations where the grid size is larger than the electron skin depth. Optimization of the PS-JFNK solver for arbitrary plasma studies in 3D will most likely involve some trade-offs among the computational expense associated with re-advancing the particles, memory associated with storing the mass matrices, and the algorithmic complexities associated with formulating a sufficiently accurate preconditioner that can also be efficiently inverted. Further consideration of these ideas is left for future work.

8.3. Computational cost of computing/using the mass matrices

One attractive feature of using the mass matrices is that the algorithm for computing them is similar to that for computing the current density in a standard PIC algorithm - by looping over the particles, computing weighting coefficients, and depositing their contribution to a container for the grid quantity [20]. Some estimates of the computational cost associated with computing/using the mass matrices are given here. It is most instructive to quantify this in terms of the operations that are being replaced during the linear stage, which is a self-consistent particle update via **Algorithm** 2 (which requires a field gather and velocity update at each Picard iteration) plus a current deposition. Note that computing the mass matrices is essentially the same set of operations as these three individual ones.

A direct counting of the number of floating-point operations is arduous, especially when using a charge-conserving interpolation method. Furthermore, it can be subject to specific implementation optimizations, and doesn't account for things like MPI exchange. A better way is to run each set of operations simultaneously many times and look at the average cost of each. This is done here using the thermalization problem from Sec. 5 and calling a field gather plus velocity update twice each (under the assumption that two Picard iterations are needed) plus a current deposition at each call to computing the mass matrices and tracking the time spent in each set of operations. The cost associated with the former set of operations is denoted as C_{DFP} (directly from particles). Using $N_{ppc} = 256$ in 2D, the cost of computing the mass matrices (averaged over 4×10^4 calls) is found to be $1.44C_{DFP}$. The cost associated with computing $\bar{\mathbf{J}}_g$ from the mass matrices at each GMRES iteration is $0.015C_{DFP}$. In other words, the overhead cost is equal to $1-2$ un-preconditioned GMRES iterations in the standard PS-JFNK method, while the cost of computing $\bar{\mathbf{J}}_g$ at each GMRES iteration is 67× less expensive. Similar, but more optimistic, numbers are obtained in 1D. The algorithm presented here has not yet been implemented in 3D. A more detailed quantification of the computational costs discussed here will be given in future work.

9. Summary

An implicit, electromagnetic PIC-MCC code is presented and shown to be capable of accurately and efficiently simulating dense plasmas. The algorithm uses a combination of techniques from the relatively new energy-conserving fully implicit [12,13,17,18] and semi-implicit [19,29,37] methods. The algorithm can *exactly* conserve global energy and local charge density and can efficiently use time steps that under-resolve the plasma period and the CFL condition with respect to the speed of light. The algorithm is fully implicit but leverages ideas from the ECSIM methods to enhance the efficiency of the PS-JFNK solver by removing the need to re-advance the particles during the linear iterations. It is shown that the modified PS-JFNK method presented here can be viewed as a modified fixed-point iteration method for the particle positions, and that this method can be reformulated as a predictor-corrector method for the particle positions that still *exactly* conserves global energy, but only approximately conserves local charge. The lack of full nonlinear consistency in this method puts a limit on how large of a time step can be used and still maintain physical accuracy, but the advantage of this method is that it is more computationally efficient at time steps where it does yield accurate results.

A first-order charge-conserving current deposition is used in this work. An orbit-average is used to maintain charge conservation for particles crossing multiple cells in a single time step. An inherent assumption here is that the particle's velocity is constant over a time step, which can potentially be a source of inaccuracies for large time steps. An alternative to the orbit-average method is the sub-cycling strategy [13,18]. This method prevents particle tunneling, which can improve both momentum conservation and long-time accuracy, but it is more computationally expensive than the orbit-average approach. In the future we will explore using a sub-cycling strategy to improve long-time accuracy.

The dynamic pinch in 1D planar geometry is used as a surrogate to quantify the algorithm's ability to simulate the dynamic compression of a plasma - a process that is typical in both DZP and ICF experiments. The algorithm is shown to be capable of accurately and efficiently modeling such systems with large grid cells ($\Delta x \gg \lambda_{De}$), large time steps ($\omega_{pe}\Delta t \gg 1$), and using a modest number of particles ($N_{ppc}\approx 100$). The ability to use this relatively low value for N_{ppc} is a direct consequence of *exact* energy conservation, which is shown to be crucial for accurate simulations of such systems. This in particular is a major advantage of the algorithm here over collisional particle codes using conventional PIC methods for modeling such systems, which often require an excessively large number of particles to mitigate effects associated with the inability to conserve energy [5,8,14].

A logical next step is to extend the work presented here to multi-dimensional curvilinear geometries [20,38]. Curvilinear geometries introduce complications not faced in planar geometries. The increase of the plasma density during compression is enhanced by $1/r$ in cylindrical geometries and by $1/r^2$ in spherical geometries. Aside from the relatively larger increase in ω_{pe} during compression, particles with variable weights are often needed to avoid using an unreasonably large number of particles at cells far from the axis. There is also a $1/r$ increase in the magnetic field during the compression in a pinch. For sufficiently large currents or small radii, the time step associated with the cyclotron period can become shorter than that associated with the plasma period. While the algorithm here is numerically stable when under-resolving the cyclotron period, it will not capture potentially important drift physics. The coupling of the method here with a hybrid, asymptotic-preserving particle pusher [39] for capturing drift physics when under-resolving the cyclotron period will be considered in future work. Additionally, the physics associated with a dynamic pinch in 2D cylindrical geometry is considerably more complex than that in 1D planar geometry [40]. This includes anomalous skin layer effects via current-driven micro instabilities [41], and the transport and deposition of high-energy particle beams generated during the nonlinear stage of $m=0$ sausage instabilities [10].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 21-FS-048 and No. 23-ERD-007.

Appendix A. Surface integral of discrete Poynting flux in 2D and 3D

The discrete energy law for the electromagnetic fields, Eq. (12), includes a volume integral of the divergence of the Poynting flux. The discrete definition for the Poynting flux for $D=1$, and the corresponding surface integral, is presented in Eq. (15). Here, we explicitly show how the Poynting flux, and its surface integral are defined for $D>1$.

Consider a 2D x-y geometry with N_x grid cells in the x-direction with uniform spacing Δx and N_y grid cells in the y-direction with uniform spacing Δy. The differential area vector is $\Delta\mathbf{A}=\pm\Delta y\hat{\mathbf{x}}$ for the upper/lower surfaces in the x-direction and $\Delta\mathbf{A}=\pm\Delta x\hat{\mathbf{y}}$ for the upper/lower surfaces in the y-direction. As mentioned in the text, $\boldsymbol{\mathcal{S}}_g$ is defined on cell faces (same as $\mathbf{B}_g$). The in-plane Poynting flux in 2D is

$$\mathcal{S}^{n+\theta}_{x,ij+\frac{1}{2}}=\frac{1}{\mu_0}\left[E^{n+\theta}_{y,ij+\frac{1}{2}}B^{n+\theta}_{z,ij+\frac{1}{2}}-E^{n+\theta}_{z,ij+\frac{1}{2}}B^{n+\theta}_{y,ij+\frac{1}{2}}\right], \tag{A.1}$$

$$\mathcal{S}^{n+\theta}_{y,i+\frac{1}{2}j}=\frac{1}{\mu_0}\left[E^{n+\theta}_{z,i+\frac{1}{2}j}B^{n+\theta}_{x,i+\frac{1}{2}j}-E^{n+\theta}_{x,i+\frac{1}{2}j}B^{n+\theta}_{z,i+\frac{1}{2}j}\right]. \tag{A.2}$$

The $E^{n+\theta}_y$ term in Eq. (A.1) and the $E^{n+\theta}_x$ term in Eq. (A.2) are properly located. The remaining terms are not and are found by averaging their values from the closest neighboring locations. That is,

$$B^{n+\theta}_{z,ij+\frac{1}{2}}=\frac{1}{2}\left(B^{n+\theta}_{z,i+\frac{1}{2}j+\frac{1}{2}}+B^{n+\theta}_{z,i-\frac{1}{2}j+\frac{1}{2}}\right),\quad E^{n+\theta}_{z,ij+\frac{1}{2}}=\frac{1}{2}\left(E^{n+\theta}_{z,ij+1}+E^{n+\theta}_{z,ij}\right), \tag{A.3}$$

$$B^{n+\theta}_{z,i+\frac{1}{2}j}=\frac{1}{2}\left(B^{n+\theta}_{z,i+\frac{1}{2}j+\frac{1}{2}}+B^{n+\theta}_{z,i+\frac{1}{2}j-\frac{1}{2}}\right),\quad E^{n+\theta}_{z,i+\frac{1}{2}j}=\frac{1}{2}\left(E^{n+\theta}_{z,i+1j}+E^{n+\theta}_{z,ij}\right), \tag{A.4}$$

$$B^{n+\theta}_{y,ij+\frac{1}{2}}=\frac{1}{4}\left(B^{n+\theta}_{y,i+\frac{1}{2}j+1}+B^{n+\theta}_{y,i+\frac{1}{2}j}+B^{n+\theta}_{y,i-\frac{1}{2}j+1}+B^{n+\theta}_{y,i-\frac{1}{2}j}\right), \tag{A.5}$$

$$B^{n+\theta}_{x,i+\frac{1}{2}j}=\frac{1}{4}\left(B^{n+\theta}_{x,i+1j+\frac{1}{2}}+B^{n+\theta}_{x,i+1j-\frac{1}{2}}+B^{n+\theta}_{x,ij+\frac{1}{2}}+B^{n+\theta}_{x,ij-\frac{1}{2}}\right). \tag{A.6}$$

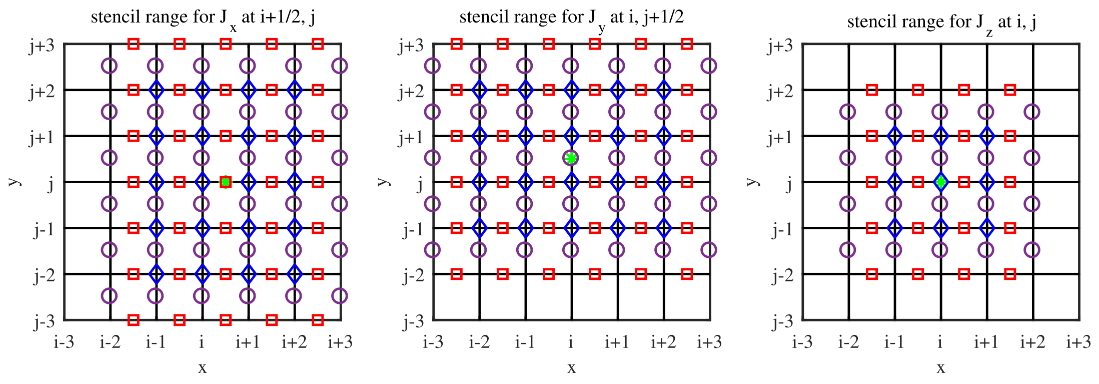
stencil range for J_x at i+1/2, j
stencil range for J_y at i, j+1/2
stencil range for J_z at i, j
j+3
j+2
j+1
j
j-1
j-2
j-3
i-3
i-2
i-1
i
i+1
i+2
i+3
x
y

Fig. B.9. Diagrams depicting the stencil range for the mass matrices when using the CC1 interpolation scheme in 2D with one cell crossing permitted in each direction. For each figure, the red boxes, purple circles, and blue diamonds represent the E_x, E_y and E_z locations, respectively, that enter the current density calculation at the location of the green asterisk.

Each of the terms on the right hand side of Eqs. (A.3)-(A.6) is taken from its native location on the Yee grid. The boundary integral in 2D is

$$\sum_{g\in\mathbf{A}} \mathcal{S}_g^{n+\theta}\cdot\Delta\mathbf{A} = \sum_{j=0}^{N_y-1} \mathcal{S}_{x,ij+\frac{1}{2}}^{n+\theta}\Big|_{i=0}^{N_x}\Delta y + \sum_{i=0}^{N_x-1} \mathcal{S}_{y,i+\frac{1}{2}j}^{n+\theta}\Big|_{j=0}^{N_y}\Delta x. \tag{A.7}$$

Extension of the results above from 2D to 3D is straightforward.

Appendix B. Mass matrices for charge-conserving current depositions

Assuming one cell crossing is permitted in each direction (i.e., particles are CFL limited), the stencil range for each component of the mass matrices when using the CC1 scheme in 2D is illustrated in Fig. B.9. The number of components associated with the mass matrices at each grid position for $D = 1, 2, 3$ are

$$N_{mm}^{D=1} = \begin{bmatrix} 5 & 4 & 4 \\ 4 & 3 & 3 \\ 4 & 3 & 3 \end{bmatrix}, \quad N_{mm}^{D=2} = \begin{bmatrix} 5\times 7 & 6^2 & 4\times 5 \\ 6^2 & 7\times 5 & 5\times 4 \\ 4\times 5 & 5\times 4 & 3\times 3 \end{bmatrix}, \quad N_{mm}^{D=3} = \begin{bmatrix} 5\times 7\times 7 & 6^3 & 6^3 \\ 6^3 & 7\times 5\times 7 & 6^3 \\ 6^3 & 6^3 & 7\times 7\times 5 \end{bmatrix}. \tag{B.1}$$

The number of components associated with each element of the mass matrices in Eq. (B.1) is expressed in terms of the product of the stencil length in each direction. For example, in 2D there the xx stencil has a range of 5 in the x-direction and 7 in the y-direction for a total of $5 \times 7 = 35$. There are a total of 33 components in 1D, 231 components in 2D, and 2031 components in 3D. If two cell crossings are permitted, then the stencil range for each in-plane direction of each element in Eq. (B.1) (except for the purely virtual terms, such as zz in 2D) increases by two. For two cell crossings, there are a total of 43, 431, and 4773 components for 1D, 2D, and 3D, respectively.

It is instructive to quantify the memory requirements in terms of simulation particles. For fully implicit PIC, each particle contains a minimum of $7 + 2D$ doubles. The 7 comes from 1 for the particle weight, 3 for the particle velocity, and another 3 for the old particle velocity. The $2D$ corresponds to the number of components needed to store the particle's current and old position vectors. The total memory for the mass matrices at each cell when allowing one cell crossing is equivalent to 3.67 simulation particles in 1D, 21 simulation particles in 2D, and 156.2 simulation particles in 3D. While these numbers are more than reasonable for 1D and 2D, the advantages of using the mass matrix formulation may diminish in 3D [37] for simulations with a moderate-to-low number of particles.

Allowing several cell crossings does not significantly increase the computational cost/memory in 1D and 2D, but it can in 3D. If using a Courant-limiting time step based on particles is too restraining, then we suggest using a hybrid method where the current density during the linear stage is written as the sum of two parts. One part is computed using the mass matrices where only Courant limited particles are included, while the current density associated with high-energy particles is computed in the conventional way (i.e., update particle velocities + current deposition). If the population of high-energy particles is relatively small, it should not affect the overall performance. Moreover, if there is a need for accurate transport of high-energy particles, such as energy deposition and transport of a high-energy beam traversing a cold background plasma, then one may want to use a CFL limited time step anyhow. Further exploration of this current density splitting concept is left to future work.

Appendix C. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcp.2023.112383.

References

[1] J.M. Dawson, Particle simulation of plasmas, Rev. Mod. Phys. 55 (1983) 403–447.
[2] C.K. Birdsall, A.B. Langdon, Plasma Physics via Computer Simulation, McGraw-Hill, New York, 1985.
[3] C.K. Birdsall, Particle-in-cell charged-particle simulations, plus Monte Carlo collisions with neutral atoms, pic-mcc, IEEE Trans. Plasma Sci. 19 (1991) 65–85.
[4] K. Nanbu, Probability theory of electron-molecule, ion-molecule, molecule-molecule, and Coulomb collisions for particle modeling of materials processing plasmas and cases, IEEE Trans. Plasma Sci. 28 (2000) 971–990.
[5] E.P. Alves, W.B. Mori, F. Fiuza, Numerical heating in particle-in-cell simulations with Monte Carlo binary collisions, Phys. Rev. E 103 (2021) 013306.
[6] A. Friedman, A. Langdon, B. Cohen, A direct method for implicit particle-in-cell simulation, Comments Plasma Phys. Control. Fusion 6 (1981) 225–236.
[7] B.I. Cohen, A. Langdon, D.W. Hewett, R.J. Procassini, Performance and optimization of direct implicit particle simulation, J. Comput. Phys. 81 (1989) 151–168.
[8] J.R. Angus, A.J. Link, A.E. Schmidt, 1d kinetic study of pinch formation in a dense plasma focus: transition from collisional to collisionless regimes, Phys. Plasmas 28 (2021) 010701.
[9] S.E. Anderson, L. Chacón, W.T. Taitano, A.N. Simakov, B.D. Keenan, Fully kinetic simulations of strong steady-state collisional planar plasma shocks, Phys. Rev. E 104 (2021) 055205.
[10] D. Klir, A.V. Shishlov, V.A. Kokshenev, P. Kubes, A.Y. Labetsky, K. Rezac, R.K. Cherdizov, J. Cikhardt, B. Cikhardtova, G.N. Dudkin, F.I. Fursov, A.A. Garapatsky, B.M. Kovalchuk, J. Krasa, J. Kravarik, N.E. Kurmaev, H. Orcikova, V.N. Padalko, N.A. Ratakhin, O. Sila, K. Turek, V.A. Varlachev, A. Velyhan, R. Wagner, Deuterium z-pinch as a powerful source of multi-mev ions and neutrons for advanced applications, Phys. Plasmas 23 (2016) 032702.
[11] E.P. Hartouni, A.S. Moore, A.J. Crilly, B.D. Appelbe, P.A. Amendt, K.L. Baker, D.T. Casey, D.S. Clark, T. Döppner, M.J. Eckart, J.E. Field, M. Gatu-Johnson, G.P. Grim, R. Hatarik, J. Jeet, S.M. Kerr, J. Kilkenny, A.L. Kritcher, K.D. Meaney, J.L. Milovich, D.H. Munro, R.C. Nora, A.E. Pak, J.E. Ralph, H.F. Robey, J.S. Ross, D.J. Schlossberg, S.M. Sepke, B.K. Spears, C.V. Young, A.B. Zylstra, Evidence for suprathermal ion distribution in burning plasmas, Nat. Phys. 19 (2023) 72–77.
[12] S. Markidis, G. Lapenta, The energy conserving particle-in-cell method, J. Comput. Phys. 230 (2011) 7037–7052.
[13] G. Chen, L. Chacón, D. Barnes, An energy- and charge-conserving, implicit, electrostatic particle-in-cell algorithm, J. Comput. Phys. 230 (2011) 7018–7036.
[14] J.R. Angus, A. Link, A. Friedman, D. Ghosh, J.D. Johnson, On numerical energy conservation for an implicit particle-in-cell method coupled with a binary Monte-Carlo algorithm for Coulomb collisions, J. Comput. Phys. (2022) 111030.
[15] J. Brackbill, D. Forslund, An implicit method for electromagnetic plasma simulation in two dimensions, J. Comput. Phys. 46 (1982) 271–308.
[16] A. Langdon, B.I. Cohen, A. Friedman, Direct implicit large time-step particle simulation of plasmas, J. Comput. Phys. 51 (1983) 107–138.
[17] G. Chen, L. Chacón, C. Leibs, D. Knoll, W. Taitano, Fluid preconditioning for Newton–Krylov-based, fully implicit, electrostatic particle-in-cell simulations, J. Comput. Phys. 258 (2014) 555–567.
[18] G. Chen, L. Chacón, A multi-dimensional, energy- and charge-conserving, nonlinearly implicit, electromagnetic Vlasov–Darwin particle-in-cell algorithm, Comput. Phys. Commun. 197 (2015) 73–87.
[19] G. Lapenta, Exactly energy conserving semi-implicit particle in cell formulation, J. Comput. Phys. 334 (2017) 349–366.
[20] D. Gonzalez-Herrero, A. Micera, E. Boella, J. Park, G. Lapenta, Ecsim-cyl: energy conserving semi-implicit particle in cell simulation in axially symmetric cylindrical coordinates, Comput. Phys. Commun. 236 (2019) 153–163.
[21] S. Mattei, K. Nishida, M. Onai, J. Lettry, M. Tran, A. Hatayama, A fully-implicit particle-in-cell Monte Carlo collision code for the simulation of inductively coupled plasmas, J. Comput. Phys. 350 (2017) 891–906.
[22] T. Takizuka, H. Abe, A binary collision model for plasma simulation with a particle code, J. Comput. Phys. 25 (1977) 205–219.
[23] K. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Trans. Antennas Propag. 14 (1966) 302–307.
[24] J. Villasenor, O. Buneman, Rigorous charge conservation for local electromagnetic field solvers, Comput. Phys. Commun. 69 (1992) 306–316.
[25] T. Esirkepov, Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor, Comput. Phys. Commun. 135 (2001) 144–153.
[26] G. Chen, L. Chacón, L. Yin, B. Albright, D. Stark, R. Bird, A semi-implicit, energy- and charge-conserving particle-in-cell algorithm for the relativistic Vlasov-Maxwell equations, J. Comput. Phys. 407 (2020) 109228.
[27] D. Knoll, D. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[28] O. Koshkarov, L. Chacón, G. Chen, L.F. Ricketson, Fast nonlinear iterative solver for an implicit, energy-conserving, asymptotic-preserving charged-particle orbit integrator, J. Comput. Phys. 459 (2022) 111146.
[29] Y. Chen, G. Tóth, Gauss's law satisfying energy-conserving semi-implicit particle-in-cell method, J. Comput. Phys. 386 (2019) 632–652.
[30] F. Bacchini, J. Amaya, G. Lapenta, The relativistic implicit particle-in-cell method, J. Phys. Conf. Ser. 1225 (2019) 012011.
[31] M. Campos Pinto, V. Pagès, A semi-implicit electromagnetic fem-pic scheme with exact energy and charge conservation, J. Comput. Phys. 453 (2022) 110912.
[32] M. Adams, P. Colella, D.T. Graves, J. Johnson, N. Keen, T.J. Ligocki, D.F. Martin, P. McCorquodale, D. Modiano, P. Schwartz, T. Sternberg, B.V. Straalen, Chombo Software Package for AMR Applications - Design Document, Technical report lbnl-6616e ed., Lawrence Berkeley National Laboratory, 2019.
[33] S. Balay, S. Abhyankar, M.F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W.D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M.G. Knepley, F. Kong, S. Kruger, D.A. May, L.C. McInnes, R.T. Mills, L. Mitchell, T. Munson, J.E. Roman, K. Rupp, P. Sanan, J. Sarich, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, J. Zhang, PETSc/TAO Users Manual, Technical Report ANL-21/39 - Revision 3.18, Argonne National Laboratory, 2022.
[34] A.E. Turrell, Processes driving non-Maxwellian distributions in high energy density plasmas, Ph.D. thesis, Imperial College London, 2013.
[35] J.E. Allen, An elementary theory of the transient pinched discharge, Proc. Phys. Soc. B 70 (1957) 24.
[36] J.R. Angus, A.J. Link, A.E.W. Schmidt, One-dimensional theory and simulations of the dynamic z-pinch, Phys. Plasmas 27 (2020) 012108.
[37] D. Gonzalez-Herrero, E. Boella, G. Lapenta, Performance analysis and implementation details of the energy conserving semi-implicit method code (ecsim), Comput. Phys. Commun. 229 (2018) 162–169.
[38] L. Chacón, G. Chen, A curvilinear, fully implicit, conservative electromagnetic pic algorithm in multiple dimensions, J. Comput. Phys. 316 (2016) 578–597.
[39] L. Ricketson, L. Chacón, An energy-conserving and asymptotic-preserving charged-particle orbit implicit time integrator for arbitrary electromagnetic fields, J. Comput. Phys. (2020) 109639.
[40] M.G. Haines, A review of the dense z-pinch, Plasma Phys. Control. Fusion 53 (2011) 093001.
[41] R. Davidson, N. Krall, Anomalous transport in high-temperature plasmas with applications to solenoidal fusion systems, Nucl. Fusion 17 (1977) 1313.