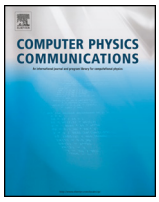
Computer Physics Communications 275 (2022) 108313

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc



Hybrid simulation of energetic particles interacting with magnetohydrodynamics using a slow manifold algorithm and GPU acceleration ☆

Chang Liu [a,*], Stephen C. Jardin [a], Hong Qin [a], Jianyuan Xiao [b], Nathaniel M. Ferraro [a], Joshua Breslau [a]

[a] Princeton Plasma Physics Laboratory, Princeton, NJ, 08540, USA
[b] University of Science and Technology of China, Hefei, 230026, China

ARTICLE INFO

Article history:
Received 2 August 2021
Received in revised form 22 January 2022
Accepted 10 February 2022
Available online 18 February 2022

Keywords:
Plasma physics
Magnetohydrodynamics
Energetic particle
Slow manifold
GPU acceleration

ABSTRACT

The hybrid method combining particle-in-cell and magnetohydrodynamics can be used to study the interaction between energetic particles and global plasma modes. In this paper we introduce the M3D-C1-K code, which is developed based on the M3D-C1 finite element code solving the magnetohydrodynamics equations, with a newly developed kinetic module simulating energetic particles. The particle pushing is done using a new algorithm by applying the Boris pusher to the classical Pauli particles to simulate the slow-manifold of particle orbits, with long-term accuracy and fidelity. The particle pushing can be accelerated using GPUs with a significant speedup. The moments of the particles are calculated using the δf method, and are coupled into the magnetohydrodynamics simulation through pressure or current coupling schemes. Several linear simulations of magnetohydrodynamics modes driven by energetic particles have been conducted using M3D-C1-K with the δf method, including fishbone, toroidal Alfvén eigenmodes and reversed shear Alfvén eigenmodes. Good agreement with previous results from other eigenvalue, kinetic and hybrid codes have been achieved.

1. Introduction

The physics of enegetic particles (EPs) is an important area of plasma physics and their confinement is critical to the success of International Thermonuclear Experimental Reactor (ITER) and future fusion reactors. EPs can interact with the bulk plasma and drive magnetohydrodynamics (MHD) instabilities, which can cause significant transport of EPs. These physics problems must be simulated comprehensively as there are strong kinetic effects associated with EPs. A widely used strategy to study EPs is the hybrid simulation, which combines the particle-in-cell (PIC) and the MHD simulations. In this method, EPs are described with markers carrying density and momentum, and are pushed following the equation of motion of the EPs with the electromagnetic fields from the MHD simulations. The moments of EPs are calculated using the obtained distribution function, where the δf method can be used to reduce the noise. The moments are then coupled into the MHD equations, which characterizes the energy and momentum exchange between the EPs and the bulk plasmas. With such a coupling scheme in the simulation, when the motion of EPs is in resonance with some MHD modes, the EP distribution can be significantly altered near the resonance region and can give strong feedback to the modes. Compared to fully kinetic simulations in which both the EPs and the bulk plasmas are described using particles, the hybrid approach can save substantial simulation time while still keeping the essential physics related to EP-MHD interaction.

In most of the previously developed hybrid simulation codes, including MEGA [1], M3D-K [2], NIMROD [3], JOREK [4], XTOR-K [5], HMGC [6], and CLT-K [7], particle pushing is done following the guiding center equations of motion in order to reduce the particle phase space dimension and allow the usage of timesteps larger than the gyro period. It has been observed [8] that advancing guiding center equations using explicit integration methods like the Runge-Kutta method can lead to breakdown of energy and momentum conservation and large deviation of particle orbits for long time simulations due to the accumulation of numerical error. Recently, a series of methods for pushing the slow manifold of magnetized particles have been developed [9]. In those methods, the mirror force was treated as an additional conservative force,

☆ The review of this paper was arranged by Prof. Z. Was.

* Corresponding author.
E-mail address: cliu@pppl.gov (C. Liu).

https://doi.org/10.1016/j.cpc.2022.108313
0010-4655/

which enabled people to use full orbit particle pushing algorithms like the Boris algorithm with timesteps larger than the gyro period while still keeping the simplicity and the structure preserving property of the algorithm.

In order to perform long time hybrid simulations to study the physics of EPs, we have developed a new hybrid code M3D-C1-K, in which we have implemented one of the slow manifold algorithms introduced in [9], whose essence is to use the Boris algorithm to push the slow manifold of classical Pauli particle orbits. The code is based on the M3D-C1 code [10], which solves the MHD equations as an initial value problem using high order 3D finite elements. The code can do both linear and nonlinear simulations, and the MHD equations can be integrated using fully implicit or semi-implicit methods [11]. The particle pushing is developed with particle based parallelization, and can run on graphics processing units (GPUs) with significant speedup compared to running on central processing units (CPUs). In addition to particle pushing, M3D-C1-K also includes the calculation of the particle distribution function evolution using the δf method, and the particle weight is used to calculate the perturbed moments. The moments of the particle distribution function are coupled with the MHD equations using one of two schemes, pressure coupling or current coupling, which utilize different orders of moments but are physically equivalent. This new code has been tested with a number of linear simulation problems including the excitation of Alfvén eigenmodes and fishbone modes, and the results agree well with those of other codes. Compares to the previous kinetic-MHD codes, M3D-C1-K is better optimized for heterogeneous CPU+GPU computing and is suitable for simulation of long-time nonlinear MHD phenomena involving kinetic effects.

This paper is organized as follows: in Sec. 2 we introduce the new slow manifold Boris algorithm used in this code, including a test run showing its conservation property. In Sec. 3 we introduce the δf method and how we calculate the particle weights that are used for deposition. In Sec. 4 we discuss the pressure coupling and current coupling schemes and how they are implemented in M3D-C1-K. In Sec. 5 we show how the code utilizes GPUs to realize particle based parallelization, and how the data is transferred between CPUs and GPUs. We also present a comparison of the particle pushing code running on CPUs and GPUs. In Sec. 6, we show a series of simulation results using this new code, and a comparison with results from other codes. In Sec. 7 we conclude.

2. Particle pushing with slow manifold algorithm

In M3D-C1-K, a hybrid model is utilized to simulate the physics of the bulk plasma and the EPs. The bulk plasma is described by the MHD equations which are solved using the finite element method. EPs are represented by markers and advanced using the particles' equations of motion, which are calculated using the electromagnetic field information obtained from the MHD equations. Then the EP information is coupled back into the MHD equations by depositing moment information onto the finite element mesh. This is similar to a PIC simulation. The difference between this and fully kinetic or gyrokinetic PIC simulation is that in a fully kinetic simulation, particle density and current are used in the Poisson's equation and the Ampere's law to calculate the electromagnetic fields. But in a hybrid model we use the pressure or current from the EPs and insert them into the MHD equations.

In previously developed hybrid codes like M3D-K [2] and NIMROD [3], the orbits of marker particles follow the drift or gyro kinetic equations. For example, the particles' equations of motion implemented in M3D-K can be written as

$$\frac{d\mathbf{X}}{dt} = \frac{1}{B^\star}\left[v_\parallel \mathbf{B}^\star - \mathbf{b}\times\left(\mathbf{E} - \frac{\mu}{q}\nabla B\right)\right], \tag{1}$$

$$m\frac{dv_\parallel}{dt} = \frac{1}{B^\star}\mathbf{B}^\star\cdot(q\mathbf{E} - \mu\nabla B), \tag{2}$$

where

$$\mathbf{B}^\star = \mathbf{B} + \frac{mv_\parallel}{q}\nabla\times\mathbf{b}, \tag{3}$$

$$B^\star = \mathbf{B}^\star\cdot\mathbf{b}. \tag{4}$$

Here $\mathbf{E}$ is the electric field, $\mathbf{B}$ is the magnetic field, $\mathbf{b} = \mathbf{B}/|B|$ is the unit vector in the direction of $\mathbf{B}$, $\mathbf{X}$ is the guiding center location, $v_\parallel$ is the parallel velocity, μ is the magnetic moment, and m and q are the mass and charge of particles.

The equations of motion are derived from a Lagrangian written in guiding center coordinates following the variational principle [12]. We can see that in this model, the gyro phase angle is an ignorable coordinate which reduces the explicit phase space from 6D to 5D. The timestep for calculating the equations of motion can be chosen based on the particles' drift motion, and can be much larger than the gyro period ($2\pi/\Omega$, Ω is the particle gyro frequency) and can thus save considerable computation time.

The guiding center equations of motion can be calculated using an explicit integration method like 4th order Runge-Kutta (RK4). Although RK4 minimizes the numerical error at every step, it has been shown that the error can accumulate and lead to nonphysical results in long time simulations [8]. For example, for a collisionless particle moving in a static magnetic field in tokamak geometry, the toroidal angular momentum ($P_\phi = q\psi + mv_\parallel RB_\phi/B$, ψ is the poloidal field flux and ϕ is the toroidal direction) and kinetic energy ($E = (1/2)mv_\parallel^2 + \mu B$) will not be conserved if using RK4 for particle pushing, leading to the particle orbit deviating from its original drift motion surface [13]. This problem can be more serious for particles with large parallel momentum such as energetic particles generated in fusion reactions or energetic electrons such as runaway electrons. To resolve this problem, symplectic algorithms [8,14] and structure-preserving methods [15] have been developed, which were designed to preserve physical Casimir invariants when integrating the equations of motion.

In this regard, in M3D-C1-K, in addition to RK4 integration of guiding center equations, we have implemented an alternative method for particle pushing, which is a volume-preserving slow manifold Boris algorithm. The Boris algorithm has been widely used for pushing particles in magnetic fields. It has been shown to have excellent long time accuracy [16]. Since it was developed for integration of full orbits of magnetized particles, the timestep is chosen to be much smaller than the gyroperiod. However, it has been shown [9] that by introducing a mirror force term which behaves like an effective electric force, one can use the Boris algorithm to calculate the slow manifold of a magnetized particle's orbit. Slow manifold provides a novel way to understand the concept of "guiding center" that does not involve complicated coordinate transformation or introduce non locality in phase space [17,18]. The mirror force term will give the effect of the gradient drift, while the curvature drift will be given by the Boris algorithm itself. The algorithm can be described as

$$\frac{\mathbf{X}_l - \mathbf{X}_{l-1}}{\Delta t} = \mathbf{V}_{l-1/2}, \tag{5}$$

$$\frac{m}{q}\frac{\mathbf{V}_{l+1/2} - \mathbf{V}_{l-1/2}}{\Delta t} = \mathbf{E}^\dagger(\mathbf{X}_l) + \frac{\mathbf{V}_{l+1/2} + \mathbf{V}_{l-1/2}}{2}\times\mathbf{B}(\mathbf{X}_l) \tag{6}$$

where $\mathbf{E}^\dagger = \mathbf{E} - \mu\nabla B$, and $\mathbf{X}_l$ and $\mathbf{V}_{l-1/2}$ characterize the location and velocity of the slow manifold at l and $l-1/2$ timestep. Though Eq. (6) looks like an implicit form with $\mathbf{V}_{l+1/2}$ appearing on both sides, it can be calculated explicitly as shown in [16].

As discussed in [9], by including the mirror force in $\mathbf{E}^\dagger$, the algorithm can be used to push particles with timesteps larger than

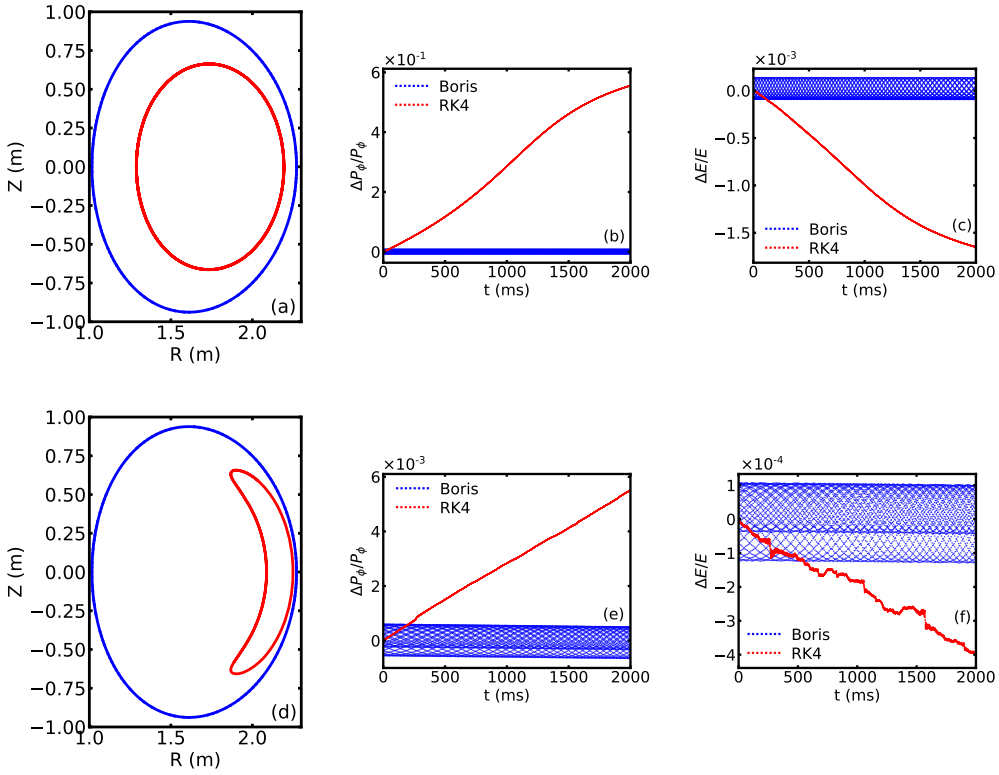


Fig. 1. The above plots show the simulation results of a passing particle with $v_\parallel = 2.4\times10^6$ m/s, $v_\perp = 7\times10^5$ m/s, including the particle orbit (red line in (a)), relative error of toroidal angular momentum P_ϕ (b) and energy E (c) using Boris and RK4 methods. The below plots show the simulation results of a trapped particle with $v_\parallel = 7\times10^5$ m/s, $v_\perp = 2.4\times10^6$ m/s, including the particle orbit (red line in (d)), relative error of P_ϕ (e) and E (f). The data points are plotted for every 100 timesteps. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$2\pi/\Omega$, as long as particles stay close to the slow manifold.[1] Note that in the Boris algorithm, $\mathbf{X}$ and $\mathbf{V}$ lie on different times with a difference of $1/2\Delta t$, which is like the leapfrog integration method. To bootstrap the Boris algorithm at the initial timestep, we use RK4 to advance the guiding center equations of motion (Eqs. (1)–(4) from $\mathbf{X}_0$ to $\mathbf{X}_1$, and then use $(\mathbf{X}_1-\mathbf{X}_0)/\Delta t$ as $\mathbf{V}_{1/2}$, to ensures that markers stay close to the slow manifold of particle motion.

To check the conservation property of the slow manifold Boris algorithm, we did a test run to push particles in a static magnetic field without an electric field. The simulation is set up in a DIII-D tokamak like geometry, with minor radius $a = 0.67$ m, major radius $R = 1.67$ m, and on-axis magnetic field $B = 2$ T. Two particles were tested. One is a passing particle with $v_\parallel = 2.4\times10^6$ m/s and $v_\perp = 7\times10^5$ m/s. The other is a trapped particle with $v_\parallel = 7\times10^5$ m/s and $v_\perp = 2.4\times10^6$ m/s. Both are initialized at the low field side. For the integration of the guiding center equations with RK4 we use a timestep $\Delta t = 3.2\times10^{-7}$ s $\approx 5/(2\pi/\Omega)$, and for the slow manifold Boris algorithm we use a smaller timestep $\Delta t' = 1/4\Delta t$ which gives a similar total computation time as RK4. Fig. 1 shows the error of the particles' toroidal angular momentum P_ϕ and energy E using the two methods. We can see that the numerical error of RK4 will accumulate and reach a significant level for long time simulation, while the error of the Boris algorithm is always bounded. The Boris algorithm shows a better long time conservation property for both P_ϕ and E, especially for passing particles with large $v_\parallel$, though the benefit is only significant for long time simulations ($t > 500$ ms). For short time simulations, the error of RK4 is smaller as it is derived from a higher order integration method.

[1] If not initialized accurately, markers may just jump back and forth across the slow manifold which leads to large errors.

The simulation result indicates that, though the conservative property of the slow manifold Boris algorithm is more attractive for long-time simulation, for most of the to-date kinetic-MHD simulations with fast ions, this advantage is not significant. Nevertheless, we have demonstrated [19] that for high-energy electron simulation, the benefit can be more significant, and we will present this using M3D-C1-K in future publications. In addition, the slow manifold Boris algorithm can give a speedup relative to RK4 when used in M3D-C1-K, which will be discussed in Sec. 5.

3. δf method and particle weight calculation

The moments of kinetic particles are calculated using their distribution function. In order to reduce the numerical noise, we use the δf method to calculate the change of the EP distribution function, meaning that for each marker, in addition to its coordinates, we also need to calculate the evolution of the value of $\delta f = f - f_0$ or particle weight $w = \delta f/f$ during the particle pushing. Here f_0 is the equilibrium EP distribution function. The δf method can be applied to linear simulations, or nonlinear simulations if the perturbed quantities are not far from their equilibrium values. This is the case, for example, in the Alfvén wave frequency chirping simulations. However, for nonlinear simulations with significant change of quantities, there are no benefits to using this and a full-f method should be used instead.

The evolution of δf can be written as

$$\frac{d\delta f}{dt} = -\frac{df_0}{dt}, \tag{7}$$

which is derived from the particle Vlasov equation $df/dt = 0$. Eq. (7) can also be written as the evolution of w as

$$\frac{dw}{dt} = -(1-w)\frac{1}{f_0}\frac{df_0}{dt}. \tag{8}$$

In the particle simulation the dw/dt term represents the change of particle weight following its trajectory, and can be calculated during particle pushing. When doing linear simulations, the particle trajectory is calculated using the equilibrium field only. In addition, the $(1-w)$ term in Eq. (8) will be replaced by 1, so that Eq. (8) only includes linear terms. For nonlinear simulations, the particle trajectory calculation includes both the equilibrium and the perturbed fields.

In the above equations, df_0/dt represents the change of the equilibrium distribution by the perturbed fields, since $df_0/dt = 0$ with the equilibrium fields only. Given that there is no electric field in the equilibrium, P_ϕ, E and μ are constants of motion in the absence of perturbations. The time derivative of f_0 can then be written as

$$\frac{df_0}{dt} = \frac{dP_\phi}{dt}\frac{\partial f_0}{\partial P_\phi} + \frac{dE}{dt}\frac{\partial f_0}{\partial E}, \tag{9}$$

and $\partial f_0/\partial P_\phi$ and $\partial f_0/\partial E$ can be calculated from the analytical expression of f_0 using the chain rule. Here we assume μ will not change with perturbed fields following the approximation of guiding center. Since P_ϕ and E will only be changed by the perturbed fields, their time derivatives can be expressed as

$$\frac{dP_\phi}{dt} = \left(\frac{d\mathbf{X}}{dt}\right)_1 \cdot \nabla\psi + \left(\frac{dv_\parallel}{dt}\right)_1 mRB_\phi/B, \tag{10}$$

$$\frac{dE}{dt} = q\mathbf{v}\cdot\mathbf{E}_1 + \mu\frac{\partial B_{1\parallel}}{\partial t}. \tag{11}$$

In a linear simulation, the $(\dots)_1$ terms can be expressed as

$$\left(\frac{d\mathbf{X}}{dt}\right)_1 = \frac{\mathbf{E}_1\times\mathbf{B}_0}{B_0^2} + v_\parallel\frac{\mathbf{B}_1}{B_0}, \tag{12}$$

$$\left(\frac{dv_\parallel}{dt}\right)_1 = q\mathbf{E}\cdot\mathbf{B}/B - \mathbf{b}_0\cdot\mu\nabla B_{1\parallel}, \tag{13}$$

where $\mathbf{E}_1$ and $\mathbf{B}_1$ are the perturbed electric and magnetic fields, and $B_{1,\parallel} = \mathbf{b}_0\cdot\mathbf{B}_1$. These equations are derived from guiding-center equations of motion, and are used in linear simulation to reduce numerical error. For a nonlinear simulation, $(d\mathbf{X}/dt)_1$ and $\left(dv_\parallel/dt\right)_1$ can be obtained by calculating the difference between $d\mathbf{X}/dt$ and $dv_\parallel/dt$ from the Boris algorithm including all the perturbed fields, with $(d\mathbf{X}/dt)_0$ and $(dv_\parallel/dt)_0$ using only the equilibrium fields following the guiding center equation, in order to include all the nonlinear contributions. $\partial B_{1\parallel}/\partial t$ and $\nabla B_{1\parallel}$ are calculated similarly by taking the difference of the results with and without perturbed fields.

To include the finite Larmor radius (FLR) effect related to physics on small spatial scales comparable to the gyroradius, one can use orbit-averaged fields $\langle\mathbf{E}_1\rangle$, $\langle\mathbf{B}_1\rangle$ in the above equations to replace the fields $\mathbf{E}_1$, $\mathbf{B}_1$ evaluated at the guiding center, like

$$\begin{aligned}\langle\mathbf{E}_1\rangle(\mathbf{X}) &= \frac{1}{2\pi}\int \mathbf{E}_1(x)\delta(\mathbf{x}-\mathbf{X}-\boldsymbol{\rho}_L)d\mathbf{x}d\theta,\\ &\approx \frac{1}{4}\sum_{j=1}^{4}\mathbf{E}_1\left(\mathbf{X}+\boldsymbol{\rho}_{L,j}\right).\end{aligned} \tag{14}$$

Here $\boldsymbol{\rho}_L = \mathbf{v}_\perp\times\mathbf{b}/\Omega$ is the gyro radius vector, $\mathbf{v}_\perp$ is the particle velocity perpendicular to the magnetic field calculated from μ, and θ is the gyro phase angle. The integration can be approximately calculated using the 4-point averaging scheme [20,21], where $\boldsymbol{\rho}_{L,j}$ are 4 vectors with length $|\rho_L|$ and are uniformly distributed in θ. In addition, for the $B_\parallel$ terms associated with the particles' magnetic moment, the gyroaverage should be taken at an effective Larmor radius $\boldsymbol{\rho}_L/\sqrt{2}$, as discussed in [22]. These terms include the second term in Eq. (11) and the second term in Eq. (13), and the mirror force term used in the calculation of the effective electric field in the Boris algorithm when doing nonlinear simulation.

The calculation of the change in particle weights needs particle's $\mathbf{v}$ and $\mathbf{x}$ (required for field evaluation) at the same time. When pushing particles using the Boris algorithm, we take $\mathbf{v}_l = \left(\mathbf{v}_{l-1/2}+\mathbf{v}_{l+1/2}\right)/2$, and use $\mathbf{x}_l$ and $\mathbf{v}_l$ in the integration of the weight equation.

After obtaining δf or w, the moments can be calculated from them. The parallel and perpendicular pressure can be calculated as

$$\begin{aligned}\delta P_\parallel(\mathbf{x}) &= \int mv_\parallel^2 wfB^* dv_\parallel d\mu d\theta,\\ &\approx \sum_k mv_\parallel^2 w_k\frac{f_k}{g_k}B^* S(\mathbf{x}-\mathbf{x}_k),\end{aligned} \tag{15}$$

$$\begin{aligned}\delta P_\perp(\mathbf{x}) &= \int \mu B\left(w+\frac{B_{1\parallel}}{B_0}\right)fB^* dv_\parallel d\mu d\theta,\\ &= \sum_k \mu B\left(w_k+\frac{B_{1\parallel}}{B_0}\right)\frac{f_k}{g_k}B^* S(\mathbf{x}-\mathbf{x}_k).\end{aligned} \tag{16}$$

Here $\sum_k$ is the summation of all the particle markers, B^* characterizes the phase space volume and is used as the Jacobian for the phase space integral, and S is the shape function used for particle deposition. In the calculation of $\delta P_\perp$ the change of perpendicular pressure due to the variation of $B_\parallel$ is taken into account. Here g represents the distribution of loaded makers, which depends on how the markers are initialized. If the makers are initialized uniformly in phase space, then $g = B^*$ and the summation should include f in the summation, or include an additional f/g term in the weight evolution equation like in M3D-K [2]. In M3D-C1-K, we initialize the markers following the same distribution function f_0 using the Monte Carlo method in order to reduce the total number of markers while keeping a low noise level. With this implementation, the marker distribution will then follow the evolution of B^*f during the simulation, and the f/g term in the summations of Eqs. (15) and (16) can be ignored.

Note that according to Eqs. (15) and (16) the change of integration Jacobian B^* can also affect the particle moments. For example, when affected by a compressing magnetic field ($\nabla\times\mathbf{E}\neq 0$), the particle distribution which is initially homogeneous in space and energy can be compressed by the $\mathbf{E}\times\mathbf{B}$ velocities and form a gradient. This effect can be captured by the particle pushing since it is equivalent to solving a continuity equation as pointed out by [20], so that $g = B^*f$ can be kept. However, it cannot be captured by the df_0/dt term as there is no gradient in the initial particle distribution function. To address this issue, we can follow the discussion in [23] and use $d = w + (1-w)B_1^*/B^*$ to replace w in the summation in Eqs. (15) and (16), which was also used in the M3D-K implementation. For a linear simulation, the definition of d is $d = w + B_1^*/B^*$ which only keeps the linear terms. Note that with this additional term and the $B_{1\parallel}/B_0$ term in Eq. (16), EPs can behave like plasma with heat capacity ratio $\gamma = 2$ in the perpendicular direction.

In a finite-element representation, the summations in Eqs. (15) and (16) can be calculated using the Galerkin method to obtain the particle pressure fields, by multiplying with a test function v_i and integrate in the elements. This can be written as

$$\int v_i\delta P_\parallel J d\mathbf{x} = \sum_k w_k mv_{k,\parallel}^2\int v_i(\mathbf{x})J(\mathbf{x})S(\mathbf{x}-\mathbf{x}_k)d\mathbf{x}, \tag{17}$$

$$\int v_i\delta P_\perp J d\mathbf{x} = \sum_k w_k\mu_k B(\mathbf{x}_k)\int v_i(\mathbf{x})J(\mathbf{x})S(\mathbf{x}-\mathbf{x}_k)d\mathbf{x}. \tag{18}$$

The polynomial coefficients can be obtained by solving the mass matrix. If we take S as a δ-function, the integral can be re-

duced and the whole calculation is significantly simplified. However, since in M3D-C1 high order polynomials are used for the test functions, the obtained pressure fields can be spiky. One can use a different S like a tent function with a finite width to get a smoother result, but this means that we also need to use a finite-width shape function when evaluating the field at the particle's location to make the whole scheme self-consistent, which can complicate the particle pushing and slow down the computation. For the linear simulations discussed in Sec. 6, we use δ-function as the particle shape function.

When performing simulations including the FLR effect, the pressure deposition should be calculated through pull-back transformation and follow the orbit average scheme with 4-point averaging. The $S(\mathbf{x}-\mathbf{x}_k)$ terms in Eqs. (17) and (18) should be replaced by $1/4\sum_{j=1}^{4} S\left(\mathbf{x}-\mathbf{X}_k-\boldsymbol{\rho}_j\right)$, which means that each particle will contribute to pressure deposition at 4 points along its gyro orbit. This implementation is consistent with the field evaluation in Eq. (14).

4. Coupling to MHD equations

In the calculation of the contribution of EPs to the MHD equations, we assume that the density of energetic particles (n_h) is small compared to the bulk ion density (n). In this case, the major contribution of EPs lies in the MHD momentum equation. Following different assumptions on the meaning of the MHD momentum equation, one can use either pressure coupling or current coupling schemes to represent this contribution.

If we assume that the MHD momentum equation describes the change of total momentum including both the energetic particles and the rest of the ions and electrons (bulk plasma), the terms related to the EP momentum change and forces should be included. In that case, the MHD momentum equation can be written as

$$\rho\left(\frac{\partial \mathbf{V}}{\partial t}\right)+\rho(\mathbf{V}\cdot\nabla\mathbf{V})+\frac{\partial \mathbf{K}_h}{\partial t}=\mathbf{J}\times\mathbf{B}-\nabla p-\nabla\cdot\mathbf{P}_h, \tag{19}$$

where ρ is the bulk plasma density, $\mathbf{V}$ is the bulk plasma velocity. $\mathbf{J}=\nabla\times\mathbf{B}$ is the total current, p is the bulk plasma pressure, and $\mathbf{P}_h=P_\parallel\mathbf{bb}+P_\perp(\mathbf{I}-\mathbf{bb})$ is the total EP pressure tensor. To use the result of the δf method, one can subtract the equilibrium force balance equation

$$\mathbf{J}_0\times\mathbf{B}_0=\nabla p_0+\nabla p_{h0}, \tag{20}$$

to only calculate the evolution of the perturbed field. Here we assume that the EP equilibrium pressure is isotropic.[2] The momentum equation then becomes

$$\rho\left(\frac{\partial \mathbf{V}}{\partial t}\right)+\rho(\mathbf{V}\cdot\nabla\mathbf{V})+\frac{\partial \mathbf{K}_h}{\partial t}=\mathbf{J}_0\times\mathbf{B}_1+\mathbf{J}_1\times\mathbf{B}_0+\mathbf{J}_1\times\mathbf{B}_1 -\nabla\delta p-\nabla\cdot\delta\mathbf{P}_h, \tag{21}$$

where $\mathbf{J}_1=\nabla\times\mathbf{B}_1$ and $\delta\mathbf{P}_h$ is calculated from δf like in Eqs. (15) and (16). This method is called "pressure coupling" and is implemented in M3D-K [2]. Note that in M3D-K, the $\partial\mathbf{K}_h/\partial t$ term is ignored assuming the EP momentum is small compared to the bulk momentum.

If we assume that the MHD momentum equation describes the momentum change of bulk plasma only and does not include the EPs, then it should be instead written as

$$\rho\left(\frac{\partial \mathbf{V}}{\partial t}\right)+(\mathbf{V}\cdot\nabla\mathbf{V})=(\mathbf{J}-\mathbf{J}_h)\times\mathbf{B}-\nabla p \tag{22}$$

where $\mathbf{J}_h$ is the EP current, and $\mathbf{J}-\mathbf{J}_h$ is the current from the bulk plasma. Here EP is coupled into the MHD equation through $\mathbf{J}_h$ rather than $\mathbf{P}_h$, therefore this method is called "current coupling". Note that in this equation we do not include the electric force on the bulk plasma $-qn_h\mathbf{E}$, which was present in the current coupling scheme in [24,25] due to the fact that the bulk plasma is non-neutral. The reason is that this term will cancel part of the $\mathbf{J}_h\times\mathbf{B}$ term which is due to the $\mathbf{E}\times\mathbf{B}$ drift of EPs, since the $\mathbf{E}\times\mathbf{B}$ drift will cause both ions and electron to move at the same velocity with their currents canceling [26].

$\mathbf{J}_h$ includes currents from the parallel motion ($J_{h,\parallel}$), the current due to the drift motion ($\mathbf{J}_{h,D}$), and the magnetization current which is due to the gyro motion of EPs ($\mathbf{J}_{h,M}$). The first two kinds of current can be calculated using the result of $d\mathbf{X}/dt$ from the guiding center equation of motion or the slow manifold Boris method after pull-back transformation [27]. Note that $J_{h,\parallel}$ will not contribute to the $\mathbf{J}\times\mathbf{B}$ force in the momentum equation. $\mathbf{J}_{h,M}$ can be calculated as [27]

$$\begin{aligned}\mathbf{J}_{h,M}(x)&=\int\dot{\boldsymbol{\rho}}\delta(\mathbf{X}+\boldsymbol{\rho}-\mathbf{x})fB^*d^3\mathbf{X}dv_\parallel d\mu d\theta,\\&=\nabla\times\mathbf{M},\end{aligned} \tag{23}$$

where $\mathbf{M}=P_\perp\mathbf{b}/B$. If we take the drift kinetic limit and choose a simple representation of drift velocity including the curvature and gradient drifts,

$$\mathbf{v}_D=\frac{mv_\parallel^2}{qB}\nabla\times\mathbf{b}+\frac{\mu}{qB}\mathbf{b}\times\nabla B, \tag{24}$$

then $\mathbf{J}_h\times\mathbf{B}$ can be simplified as

$$\begin{aligned}\mathbf{J}_h\times\mathbf{B}&=\left[q\int\mathbf{v}_D f d^3\mathbf{v}+\nabla\times\mathbf{M}\right]\times\mathbf{B},\\&=P_\parallel\mathbf{b}\cdot\nabla\mathbf{b}-P_\perp\nabla\ln B\times\mathbf{b}\times\mathbf{b}-\nabla\times\left(\frac{P_\perp}{B}\mathbf{b}\right)\times\mathbf{B},\\&=\left[\nabla P_\perp+\nabla\cdot\left[\left(P_\parallel-P_\perp\right)\mathbf{bb}\right]\right]\times\mathbf{b}\times\mathbf{b},\end{aligned} \tag{25}$$

which is close to the $\nabla\cdot\mathbf{P}_h$ term in the pressure coupling scheme, except that here the component parallel to $\mathbf{b}$ is eliminated by the $\times\mathbf{b}\times\mathbf{b}$ operator. This means that we can use the result of $\mathbf{P}_\parallel$ and $\mathbf{P}_\perp$ calculated from Eqs. (15) and (16) for both pressure and current coupling schemes, rather than calculating $\mathbf{J}_h$ separately. The strategy for current coupling is the same as the method used in MEGA [1]. For simulation including FLR, this relationship is still valid as both $\mathbf{J}_h$ and $\mathbf{P}_h$ should be calculated by doing pull-back transformation.

When doing a δf simulation, one should subtract the equilibrium force balance equation like in Eq. (21),

$$\rho\left(\frac{\partial \mathbf{V}}{\partial t}\right)+\rho(\mathbf{V}\cdot\nabla\mathbf{V})=(\mathbf{J}_0-\mathbf{J}_{h0})\times\mathbf{B}_1+(\mathbf{J}_1-\delta\mathbf{J}_h)\times(\mathbf{B}_0+\mathbf{B}_1) -\nabla\delta p. \tag{26}$$

The pressure terms in Eq. (25) should be replaced by $\delta P_\parallel$ and $\delta P_\perp$ to give the result of $\delta\mathbf{J}_h\times(\mathbf{B}_0+\mathbf{B}_1)$. Assuming that the equilibrium EP current $\mathbf{J}_{h0}$ is perpendicular to $\mathbf{B}_0$ and satisfies the force balance $\mathbf{J}_{h0}\times\mathbf{B}_0=\nabla p_{h0}$, this force of $\mathbf{J}_{h,0}\times\mathbf{B}_1$ can be written as

$$\mathbf{J}_{h,0}\times\mathbf{B}_1=\mathbf{b}_0\frac{\mathbf{B}_1}{B_0}\cdot\nabla p_{h0}. \tag{27}$$

This simplified current coupling scheme was also implemented in the MEGA code [1].

Note that in the pressure coupling scheme in [25], only the perpendicular part of $\nabla\cdot\mathbf{P}_h$ is added in the momentum equation,

[2] For anisotropic equilibrium EP pressure, there are additional terms that comes from the $\nabla\cdot\left[(P_{\parallel 0}-P_{\perp 0})\mathbf{bb}\right]$ which is associated with the perturbed $\mathbf{B}$ fields.

which is exactly the same as the result in Eq. (25) and is equivalent to the simplified current coupling scheme. The reason for only including the perpendicular part is that, assuming the perpendicular motion of both bulk plasma and EPs are dominated by $\mathbf{E}\times\mathbf{B}$ drifts, then $\mathbf{K}_h$ in the perpendicular direction is much smaller compared to $\rho\mathbf{V}$ as $n_h \ll n$ and can be safely neglected. However, in the parallel direction $\partial\mathbf{K}_h/\partial t$ cannot be ignored, and the momentum evolution of the bulk plasma and EPs should be calculated separately. In the pressure coupling scheme implemented in M3D-C1-K, we still include all the components of the $\nabla\cdot\mathbf{P}_h$ term and ignore the $\partial\mathbf{K}_h/\partial t$ in all directions to follow the implementation in M3D-K. We find that for all the simulations we have conducted, the two coupling schemes give almost the same results, with no difference on numerical stability, which will be discussed in Sec. 6. These results indicate that the EP parallel momentum term does not affect much the final results in those cases.

5. GPU acceleration of particle pushing

The M3D-C1 code was developed using the distributed memory parallelization model with Message Passing Interface (MPI). The whole 3D mesh is decomposed into the same number of subdomains as the number of CPU processes. Each process is responsible for calculating the elements of the MHD equation matrices for one subdomain, and only manages the memory of fields within it. This is called "domain-based parallelization". When developing the particle pushing code for M3D-C1-K, we used a hybrid parallelization model, which employs "particle-based parallelization" and "shared memory model" for particle pushing. We find that if we stuck with the domain-based model, the code would then need to take care of particles moving from one subdomain to another, which would involve frequent communication between different processes or threads that can significantly slow down the computation. In the particle-based parallelization, each parallel thread takes care of pushing one particle for several timesteps independent of other threads. The field information is duplicated across all the GPUs. Therefore this model is suitable for large-scale parallel computing using GPUs. This strategy of particle-based parallelization is also used in many gyrokinetic codes like GTC [28] and GTS.

In the development of M3D-C1-K, we utilized GPUs to accelerate particle pushing and particle weight calculation, which is the most time-consuming part of the kinetic module. The particle pushing code is developed using OpenACC. OpenACC is a coding standard similar to OpenMP, which provides a list of directives to help write parallel computing code and simplify data communication operations between hosts and accelerator devices (such as GPUs). We also implement the multi-thread parallelization of particle pushing on multi-core CPUs using OpenMP, so that the code can run on just CPUs or with GPUs by setting compilation directives. The calculation of the MHD equation finite element matrix and the matrix solving is still done by the M3D-C1 code using CPUs.

In the implementation of particle-based parallelization, each particle pushing thread must have access to the electromagnetic field information in the whole mesh, so that the particle can move to an arbitrary location in the mesh without performing extra communication. This means that the field information must be collected from each CPU processes after the MHD calculation and uploaded to the shared memory of each GPU. For most modern GPUs, the memory is large enough to store the field information of the whole 3D mesh. The data collection on CPU processes is done utilizing the MPI Shared Memory (SHM) model introduced in MPI-3, which can accelerate the communication between processes on the same computation node. For communication between different nodes, the classical message communication interface is used. After the pushing, the particle information needs to be downloaded from GPUs and distributed into the distributed memory of each MPI processes. The data distribution work and the calculation of $P_\parallel$ and $P_\perp$ for pressure or current coupling is done using CPUs.

The fields and particles are evolved separately in M3D-C1-K. The field is evolved according to the MHD equations and is integrated using the implicit or semi-implicit method [11]. In the simulation we find that the MHD timestep is not limited by the Courant–Friedrichs–Lewy (CFL) condition just like in semi-implicit method, even if the EP calculation is explicit. This means that the semi-implicit operator [29] is powerful enough to suppress the numerical instabilities brought by the explicit methods in particle pushing. The particle pushing and weight calculation are done between the integral of two adjacent MHD timesteps. It has subcycles for particle pushing in order to increase the accuracy of particle orbit calculation. The transfer of field information is done before the beginning of the particle pushing subcycles, which can save time for communication between CPUs and GPUs. During the subcycles, the fields are assumed to be static.

The performance benchmark of the particle pushing code in M3D-C1-K on CPUs and GPUs is shown in Fig. 2. After porting the code to GPUs without any modification to the algorithm, we get about 11 times speed up when pushing 16 million particles for 50 steps using RK4. The benchmark was done on the Summit cluster using four nodes. The CPU run utilizes 8 IBM POWER9 CPUs with 22 SIMD Multi-Core (SMC) on each processor. The GPU run utilizes 24 NVIDIA Tesla V100 GPUs. The simulation is set in a 3D mesh with a DIII-D like geometry, with 4 toroidal planes and 5679 elements per plane. The particles are uniformly distributed in the 3D mesh with a Maxwellian distribution.

It is also shown in Fig. 2 that there is a speedup when using the slow manifold Boris algorithm for 200 steps and timestep 1/4 of that of RK4 on GPUs. This speedup is brought by the simplification of the field evaluation in the Boris algorithm. In M3D-C1-K, the electromagnetic fields are represented using scalar and vector potentials $(\phi, \mathbf{A})$. When evaluating the fields $(\mathbf{E}, \mathbf{B})$ at a specific point during particle pushing, the derivatives of the polynomials are needed. Thus if calculating terms like the magnetic field curvature term in the guiding center equations, one needs to calculate the second order derivatives of the polynomials, which can be time-consuming when using a 3D mesh. After profiling the particle pushing code using RK4, it was found that most of the time is spent in the evaluation of the second order derivatives of polynomials. When using the Boris algorithm, the magnetic field curvature term is not needed, and the gradient term ∇B can be easily calculated by treating B as an additional scalar field, thus only the first order derivative of the polynomials is needed.

More speedup can be achieved by further optimization, for example, by improving the coalescence of the GPU memory access and using single-precision floating-point arithmetic. For simulations shown in this paper with energetic ions, it is found that the computation time for particle pushing is already close to the computation time spent on the CPU for the MHD calculation, thus further optimization on particle pushing is not critical to the overall performance. For future simulations with higher energy ions or runaway electrons, the optimization on particle pushing is still important.

6. Simulation results

In this section we show the linear simulation results of M3D-C1-K, including fishbone, toroidal Alfvén eigenmode (TAE), and reversed shear Alfvén eigenmode (RSAE). The results are compared with those from other codes, including the mode frequency, growth rate, and structure.

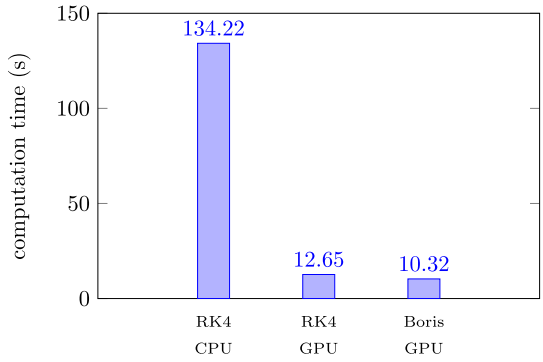


Fig. 2. Computation time for pushing 16 million particles in a 3D mesh with 4 toroidal planes (5679 elements per plane) using different methods and processors. For RK4 the particle-pushing was done for 50 steps and for the Boris algorithm the pushing was done for 200 steps with the timestep 1/4 of that used in RK4 simulation. Note that this is a comparsion for the computation time for particle pushing only and not including MHD calculation.

6.1. Linear fishbone simulation

For the linear fishbone simulation we followed the setup in [2], which includes a benchmark study of linear fishbone simulations between M3D-K and NOVA-K in a large aspect ratio circular tokamak. The setup was also used for a benchmark between NIMROD and M3D-K in [3]. A circular tokamak with $R = 1$ m and $a = 0.361925$ m was chosen for the test. The plasma consists of hydrogen ions whose density is uniform with $n_0 = 2.489 \times 10^{20}$ m^{-3}. The total pressure profile is $p(\psi) = p_0 \exp(-\psi/0.25)$, where ψ is the normalized poloidal flux ranging from 0 at the magnetic axis to 1 at the boundary. The central pressure $p_0 = 16335$ Pa and the central total plasma beta β_{total} is 8%. The toroidal field at the magnetic axis is $B_T = 1$ T. The safety factor (q) profile is given by an analytical expression,

$$q = q_0 + \psi \left[q_1 - q_0 + \left(q_1' - q_1 + q_0\right) \frac{(1-\psi_s)(\psi - 1)}{\psi - \psi_s} \right], \quad (28)$$

where $q_0 = 0.6$ and $q_0' = 0.78$ are the value and derivative of q at $\psi = 0$, $q_1 = 2.5$ and $q_1' = 5.0$ are the value and derivative of q at $\psi = 1$. $\psi_s = \left(q_1' - q_1 + q_0\right) / \left(q_0' + q_1' - 2q_1 + 2q_0\right)$.

The density profile of EPs has the same shape as the plasma pressure profile. In momentum space it follows an isotropic slowing down distribution given by

$$f(v) = \frac{H(v_0 - v)}{v^3 + v_c^3}, \quad (29)$$

where $v_0 = 3.9 \times 10^6$ m/s is the maximum velocity of EPs and $v_c = 0.58 v_0$ is the critical velocity. The same value of v_0 and v_c is used for all flux surfaces. Since the EP density and pressure follow the same spatial profile as the plasma pressure, we can vary the value of the EP density and the bulk plasma pressure to change the ratio of β_h/β_{total} (β_h is the EP pressure beta) while keeping the total pressure profile fixed. Note that when initializing the EP distribution we did not consider the average value of ψ for passing and trapped particles like the calculation in [2]. Instead, we just used the local value of ψ for EP initialization. In order to satisfy the very small value of normalized Larmor radius used in [2], $\rho_L = v_0/(\Omega a) = 0.0125$, we use a reduced EP ion mass $m_{EP} = 0.11 m_H$ (m_H is the hydrogen mass). This can help reduce the finite orbit width (FOW) effect of EPs.

The results of a linear simulation with toroidal mode number $n = 1$ are shown in Fig. 3, including simulations using pressure coupling and current coupling schemes. The FLR effect was not included in the simulation. We can see that both the growth rate (γ) and the real frequency (ω) agree well with the M3D-K and NIMROD results, except for the mode real frequency at large β_h/β_{total}. When β_h/β_{total} increases from 0 to 0.75, the mode changes from an ideal MHD kink mode to a fishbone mode with a finite real frequency due to the response of EPs. The mode growth rate decreases as β_h/β_{total} changes from 0 to 0.25, and then increases as β_h/β_{total} changes from 0.25 to 0.75. The real frequency is zero with $\beta_h = 0$ and increases almost linearly as β_h increases. The results of simulations using pressure coupling and current coupling schemes are almost identical.

The mode structure of the perturbed poloidal flux ($\delta\psi$), the perturbed EP parallel pressure ($\delta p_\parallel$) and the difference between the perturbed parallel and perpendicular EP pressure ($\delta p_\perp - \delta p_\parallel$) for a linear $n = 1$ simulation with $\beta_h/\beta_{total} = 0.5$ are shown in Fig. 4. Note that the non-adiabatic response of EP pressure ($\delta p_\perp - \delta p_\parallel$) is localized at the low-field-side, indicating that this pressure perturbation mostly comes from trapped particles through resonance with the fishbone mode. The particle pressure results have some noise because of the usage of the δ particle shape function and high-order polynomials as test functions. The mode structure results are consistent with the NIMROD simulation results in [3].

6.2. TAE simulation

For TAE linear simulation we used the setup in [30], which was also used for a NIMROD TAE simulation in [31]. The simulation was done in a large aspect ratio tokamak ($R = 10$ m, $a = 1$ m). The magnetic field on axis is $B_T = 3$ T. The bulk ions are hydrogen with a uniform density of $n_0 = 2 \times 10^{19}$ m^{-3}. The bulk plasma pressure is set to be constant to avoid pressure gradient driven modes, $p =$ 6408 Pa. The safety factor profile is $q(r) = 1.71 + 0.16(r/a)^2$. Note that at $r = 0.5a$ there is a rational surface $q = 1.75$.

The energetic ions are deuterium and have a density profile given by

$$n(s) = n_0 c_3 \exp\left(-\frac{c_2}{c_1} \tanh \frac{\sqrt{s} - c_0}{c_2}\right), \quad (30)$$

where $s = \psi_t/\psi_t(a)$ is the normalized toroidal flux. $n_0 = 1.4431 \times 10^{17}$ m^{-3} is the EP density at $s = 0$. The coefficients $c_0 = 0.49123$, $c_1 = 0.298228$, $c_2 = 0.198739$ and $c_3 = 0.521298$. This EP density profile has a large gradient at the rational surface $q = 1.75$, which can drive TAE. The EPs are initialized with a Maxwellian distribution in velocity space with a uniform temperature T_f.

The linear TAE simulation was done for $n = 6$. The growth rates and frequencies of TAEs as functions of EP temperature from the M3D-C1-K simulations are shown in Fig. 5, including the results in the zero Larmor radius (ZLR) limit, and the results including FLR effect by taking 4-point gyro-averages. The results are plotted along with the simulation results from other codes which were benchmarked in [30]. We can see that the M3D-C1-K results are close to the results from gyrokinetic, hybrid-MHD and eigenvalue codes. After including the FLR effect, the mode growth rates drop significantly for high T_f cases as the EP Larmor radius is large for those cases. The TAE frequencies also drop slightly with the FLR effect. We have done the simulations using both pressure and current coupling, and the results of mode growth rates and frequencies are equal.

The mode structure of the perturbed poloidal vorticity ($\delta\phi$) from the M3D-C1-K simulation for $T_f = 400$ keV including FLR effects is shown in Fig. 6. The radial structure indicates that the mode is localized near the $r = 0.5a$ rational surface and is dominated by $m = 10$ and $m = 11$ harmonics, which is consistent with the fact that the mode lies at the rational surface $q = 1.75 = 0.5 \times (10 + 11)/n$.

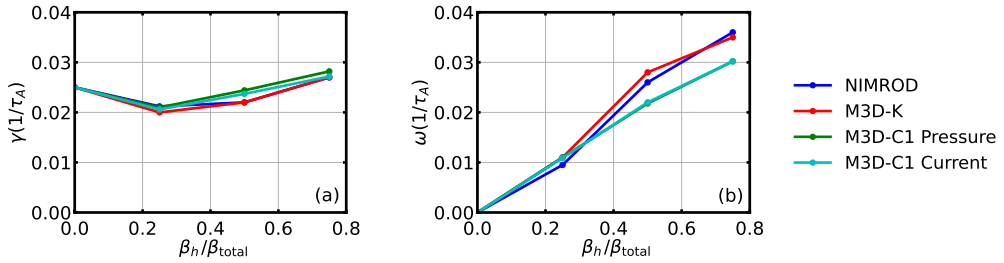


Fig. 3. Simulation results of mode growth rate (a) and real frequency (b) as functions of EP beta fraction of the $n = 1$ fishbone. Blue line is the result of NIMROD [3]. Red line is the result of M3D-K [2]. Green line is the result of M3D-C1-K using pressure coupling, and the cyan line is the result using current coupling.

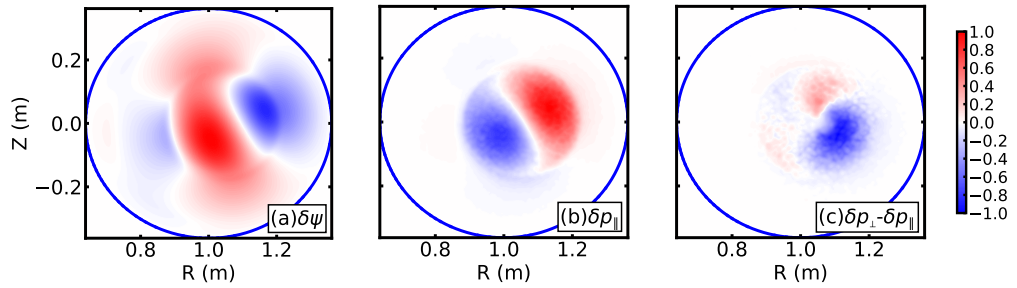


Fig. 4. Structure of the perturbed poloidal flux $\delta\psi$ (a), the perturbed EP parallel pressure $\delta p_{\parallel}$ (b) and the difference between the perturbed parallel and perpendicular EP pressure $\delta p_{\perp} - \delta p_{\parallel}$ (c) from the $n = 1$ linear fishbone simulation with $\beta_h/\beta_{total} = 0.5$ using M3D-C1-K. The values are normalized according to the maximum absolute value.

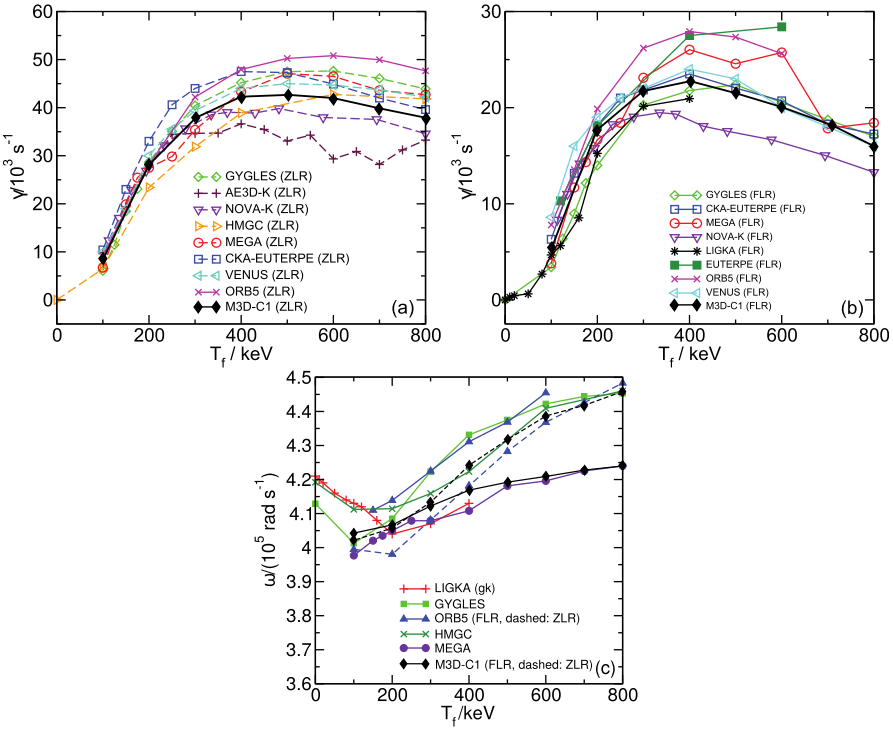


Fig. 5. Mode growth rates from calculations without FLR effects (a), with FLR effects (b) and the mode frequencies (c) as functions of T_f for the linear $n = 6$ TAE simulation. The black diamonds show the results from M3D-C1-K, on top of results from other codes presented in [30].

6.3. RSAE simulation

We also performed linear RSAE simulations in M3D-C1-K. For those simulations we used real tokamak geometry with plasma equilibrium and EP distribution from experimental diagnostics. The equilibrium is obtained from DIII-D shot #159243 at 805 ms, during which the deuterium NBI is activated and a series of RSAEs were excited and measured [32,33]. The simulation follows the setup in [34], in which a number of eigenvalue, gyrokinetic and hybrid-MHD codes participated in a linear benchmark. The equilibrium fields, including the pressure profile, were read from the result of the equilibrium code kinetic EFIT, which takes into account the kinetic ion contribution in calculating the Grad-Shafranov (G-S) equation. As shown in Fig. 3 in [34], the safety factor q profile has a minimum point ($q_{min} = 2.94$) at $\rho = 0.4$ (ρ is the normalized square root of toroidal flux). The EP distribution is approximated by an isotropic Maxwellian distribution. Here we used the EP density and temperature profile from kinetic EFIT, where the EP pressure was estimated by subtracting the measured thermal

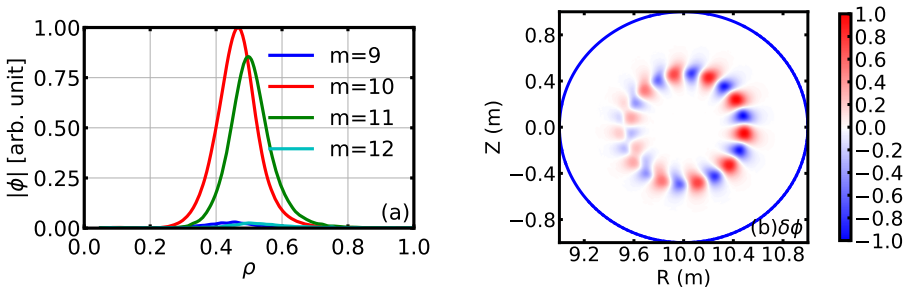


Fig. 6. (a) Poloidally averaged radial structure of perturbed poloidal vorticity $\delta\phi$ of different poloidal harmonics from the $n = 6$ TAE simulation using M3D-C1-K. (b) Poloidal structure of $\delta\phi$. The values are normalized according to the maximum absolution value.

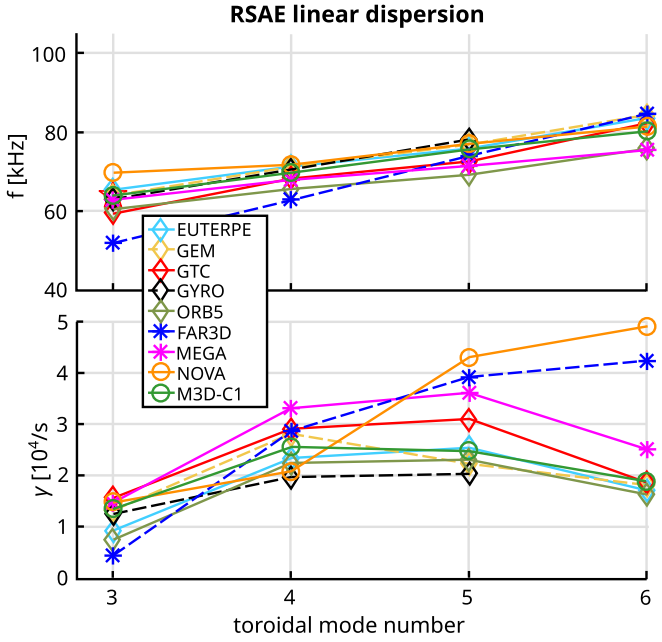


Fig. 7. Mode frequencies (f) and growth rates (γ) from calculations with FLR effects for different n numbers for the linear RSAE simulation. The green circles show the results from M3D-C1-K, on top of results from other codes presented in [34].

pressure from the computed total pressure using the equilibrium reconstruction. The density and temperature profiles of both bulk plasma and fast ions used in the M3D-C1-K simulation have been carefully compared with the data used in [34] to make sure they are in good agreement.

Using this equilibrium, we did linear simulations using M3D-C1-K for $n = 3 - 6$. The results of the RSAE real frequencies and growth rates are plotted in Fig. 7 along with the results from other codes presented in [34]. The M3D-C1-K results agree well with results from the other initial value MHD and gyrokinetic codes. The mode frequencies increase as the n number, while the growth rate is largest for $n = 4$ and 5. For those simulations we include the FLR effect, and we found that FLR can lead to a decrease of the mode growth rate similar to what we found for the TAE simulation. The mode structure of the $n = 4$ RSAE simulation is shown in Fig. 8, including the radial structure of different m harmonics of $\delta\phi$ and the 2D poloidal structure. The perturbed field is localized near the $q = q_{min}$ flux surface and is dominated by the $m = 12$ component, which is consistent with the RSAE physics ($q_{min} \approx m/n$) and in agreement with the results in [34] from the other codes.

7. Conclusions

In this paper we have introduced the new code M3D-C1-K, which was developed based on the M3D-C1 MHD code with particle simulation for the kinetic effects. The particles are described using markers, which are pushed using a new slow manifold Boris algorithm. This new algorithm can provide good conservation properties for long time simulations. In addition, it can simplify the field evaluation calculation and speed up the particle pushing. The particle simulation is interfaced with the MHD code by calculating the moments of the particles using the δf method, and then coupled into the MHD equations through pressure or current. The particle pushing code has been ported to run on GPUs, which gives a 11 times speed up compared to the CPU version. Both the linear fishbone simulations and the linear Alfvén mode simulations, including TAE and RSAE, have been conducted, and the results agree well with previous results from other codes.

M3D-C1-K is based on M3D-C1, which utilizes the semi-implicit method to do MHD calculations with large timesteps. To fit the kinetic part into this framework, we integrate the MHD and particle equations separately, and introduce subcycles for particle pushing. Given that the MHD equations are still evolved using a large timestep which is not limited by the CFL condition, and particle pushing on GPUs is very fast, we believe that M3D-C1-K is suitable for simulation of long-time MHD phenomena involving kinetic effects, including the nonlinear evolution of EP-driven Alfvén modes with frequency chirping and mode coupling, and kink or tearing modes interacting with EPs. For those simulations, the computation time spent on the MHD calculation on CPUs and particle pushing on GPUs are comparable. For phenomena involving wave-particle interaction over short timescales, such as global Alfvén eigenmodes (GAEs) or compressional Alfvén eigenmodes (CAEs), small MHD timesteps are required which can make the MHD calculation take most of the computation time. In order to better simulate these kinds of problems, we plan to further optimize the MHD calculation and have it utilize GPUs.

The new slow manifold Boris algorithm used in the code was originally developed to preserve physical structures and conserve constants of motion, which can improve the credibility of long time simulations. As discussed in Sec. 2, this advantage is not significant for a typical EP simulation of only hundreds of milliseconds as RK4 can provide similar order of absolute numerical error. For longer time simulations the benefit of the Boris algorithm can be more significant. In addition, this advantage can be more important for simulating particles with large parallel velocities such as high energy electrons. These electrons can be generated through inductive electric fields as runaway electrons, or through external current drive with plasma waves, and can interact with MHD modes. Given that the high-energy electrons can have velocities close to the speed of light, it is important to have a particle pushing algorithm that can conserve the toroidal momentum and keep the shape of the particle's orbit, as discussed in [35,19]. The slow manifold Boris algorithm therefore is a good candidate for doing nonlinear MHD simulation with energetic electrons. Nevertheless, the particle pushing algorithm can still be further optimized and

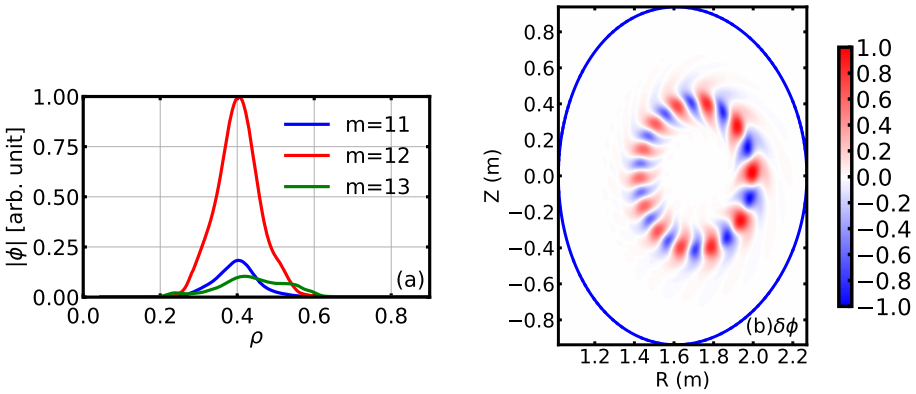


Fig. 8. (a) Poloidally averaged radial structure of perturbed poloidal vorticity $\delta\phi$ of different poloidal harmonics of the $n = 4$ RSAE simulation using M3D-C1-K. (b) Poloidal structure of $\delta\phi$. The values are normalized according to the maximum absolution value.

combined with the field evolution to improve numerical stability, like in [36], which will be done in future work.

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.

Acknowledgements

We would like to thank Yasushi Todo, Andreas Bierwage, Elena Belova, Nikolai Gorelenkov, Xin Wang, Roscoe White, Jin Chen, Zhihong Lin and Amitava Bhattacharjee for fruitful discussion. We would like to thank Seegyoung Seol and Mark Shephard of the Scientific Computation Research Center (SCOREC) group at Rensselaer Polytechnic Institute (RPI) for the implementation and support of unstructured meshing capabilities in M3D-C1. This work was supported by U.S. Department of Energy grant DE-AC02-09CH11466. This research used the Traverse cluster at Princeton University and AiMOS cluster of Center for Computational Innovations (CCI) at RPI. It also used the Summit cluster of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Y. Todo, T. Sato, Phys. Plasmas 5 (5) (1998) 1321–1327.
[2] G.Y. Fu, W. Park, H.R. Strauss, J. Breslau, J. Chen, S. Jardin, L.E. Sugiyama, Phys. Plasmas 13 (5) (2006) 052517.
[3] C.C. Kim the NIMROD Team, Phys. Plasmas 15 (7) (2008) 072507.
[4] G.T.A. Huysmans, O. Czarny, Nucl. Fusion 47 (7) (2007) 659–666.
[5] H. Lütjens, J.-F. Luciani, J. Comput. Phys. 227 (14) (2008) 6944–6966.
[6] S. Briguglio, G. Vlad, F. Zonca, C. Kar, Phys. Plasmas 2 (10) (1995) 3711–3723.
[7] J. Zhu, Z.W. Ma, S. Wang, Phys. Plasmas 23 (12) (2016) 122506.
[8] H. Qin, X. Guan, Phys. Rev. Lett. 100 (3) (2008) 035006.
[9] J. Xiao, H. Qin, Comput. Phys. Commun. 265 (2021) 107981.
[10] N.M. Ferraro, S.C. Jardin, J. Comput. Phys. 228 (20) (2009) 7742–7770.
[11] S.C. Jardin, N. Ferraro, J. Breslau, J. Chen, Comput. Sci. Discov. 5 (1) (2012) 014002.
[12] R.G. Littlejohn, J. Plasma Phys. 29 (01) (1983) 111–125.
[13] H. Qin, X. Guan, W.M. Tang, Phys. Plasmas 16 (4) (2009) 042510.
[14] C.L. Ellison, J.M. Finn, J.W. Burby, M. Kraus, H. Qin, W.M. Tang, Phys. Plasmas 25 (5) (2018) 052502.
[15] J. Liu, Y. Wang, H. Qin, Nucl. Fusion 56 (6) (2016) 064002.
[16] H. Qin, S. Zhang, J. Xiao, J. Liu, Y. Sun, W.M. Tang, Phys. Plasmas 20 (8) (2013) 084503.
[17] J.W. Burby, J. Math. Phys. 61 (1) (2020) 012703.
[18] J.W. Burby, E. Hirvijoki, J. Math. Phys. 62 (9) (2021) 093506.
[19] C. Liu, C. Zhao, S.C. Jardin, N. Ferraro, C. Paz-Soldan, Y. Liu, B.C. Lyons, Plasma Phys. Control. Fusion (2021).
[20] W.W. Lee, J. Comput. Phys. 72 (1) (1987) 243–269.
[21] W.X. Wang, Z. Lin, W.M. Tang, W.W. Lee, S. Ethier, J.L.V. Lewandowski, G. Rewoldt, T.S. Hahm, J. Manickam, Phys. Plasmas 13 (9) (2006) 092505.
[22] L. Chen, Y. Lin, X.Y. Wang, J. Bao, Plasma Phys. Control. Fusion 61 (3) (2019) 035004.
[23] E.V. Belova, R.E. Denton, A.A. Chan, J. Comput. Phys. 136 (2) (1997) 324–336.
[24] W. Park, S. Parker, H. Biglari, M. Chance, L. Chen, C.Z. Cheng, T.S. Hahm, W.W. Lee, R. Kulsrud, D. Monticello, L. Sugiyama, R. White, Phys. Fluids, B Plasma Phys. 4 (7) (1992) 2033–2037.
[25] W. Park, E.V. Belova, G.Y. Fu, X.Z. Tang, H.R. Strauss, L.E. Sugiyama, Phys. Plasmas 6 (5) (1999) 1796–1803.
[26] C. Zhao, C. Liu, S.C. Jardin, N.M. Ferraro, Nucl. Fusion 60 (12) (2020) 126017.
[27] H. Qin, W.M. Tang, Phys. Plasmas 11 (3) (2004) 1052–1063.
[28] W. Zhang, W. Joubert, P. Wang, B. Wang, W. Tang, M. Niemerg, L. Shi, S. Taimourzadeh, J. Bao, Z. Lin, in: International Workshop on Accelerator Programming Using Directives, Springer, 2018, pp. 3–21.
[29] D.D. Schnack, D.C. Barnes, Z. Mikic, D.S. Harned, E.J. Caramana, J. Comput. Phys. 70 (2) (1987) 330–354.
[30] A. Könies, S. Briguglio, N. Gorelenkov, T. Fehér, M. Isaev, P. Lauber, A. Mishchenko, D.A. Spong, Y. Todo, W.A. Cooper, R. Hatzky, R. Kleiber, M. Borchardt, G. Vlad, A. Biancalani, A. Bottino, ITPA EP TG, Nucl. Fusion 58 (12) (2018) 126027.
[31] Y. Hou, P. Zhu, C.C. Kim, Z. Hu, Z. Zou, Z. Wang, Phys. Plasmas 25 (1) (2018) 012501.
[32] C.S. Collins, W.W. Heidbrink, M.E. Austin, G.J. Kramer, D.C. Pace, C.C. Petty, L. Stagner, M.A. Van Zeeland, R.B. White, Y.B. Zhu, Phys. Rev. Lett. 116 (9) (2016) 095001.
[33] W.W. Heidbrink, C.S. Collins, M. Podestà, G.J. Kramer, D.C. Pace, C.C. Petty, L. Stagner, M.A. Van Zeeland, R.B. White, Y.B. Zhu, Phys. Plasmas 24 (5) (2017) 056109.
[34] S. Taimourzadeh, E.M. Bass, Y. Chen, C. Collins, N.N. Gorelenkov, A. Könies, Z.X. Lu, D.A. Spong, Y. Todo, M.E. Austin, J. Bao, A. Biancalani, M. Borchardt, A. Bottino, W.W. Heidbrink, R. Kleiber, Z. Lin, A. Mishchenko, L. Shi, J. Varela, R.E. Waltz, G. Yu, W.L. Zhang, Y. Zhu, Nucl. Fusion 59 (6) (2019) 066006.
[35] X. Guan, H. Qin, N.J. Fisch, Phys. Plasmas 17 (9) (2010) 092502.
[36] Z.X. Lu, G. Meng, M. Hoelzl, P. Lauber, J. Comput. Phys. 440 (2021) 110384.