Development of energetic particle module in M3D-C1

Chang Liu

Princeton Plasma Physics Laboratory

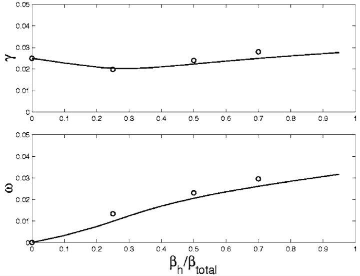
CTTS Meeting
Oct 20 2019

Motivation of developing particle module in M3D-C1

- We want to have the same capability of the kinetic module of M3D-K code in M3D-C1, to study the interaction between energetic ions and MHD activities (Alfven waves, kink/tearing modes etc).
- With more advanced finite-element representation and implicit time advance method, M3D-C1 can study the nonlinear problem with larger timestep and save computation time.
    - Explicit particle pushing can be accelerated using modern HPC with GPU, like in PIC codes.
- Near Goal: Reproduce the fishbone simulation result in Fu (2006) and Kim (2008).

Benchmark case: fishbone simulation with varying β_h/β_{total}

- The test case we are working on is a $n = 1$ fishbone mode linear simulation, which is based on an internal kink mode.
- By varying the energetic particle beta while fixed the total beta, the mode will have a real frequency (ω) that grows with β_h/β_{total}, while the growth rate changes little.
  - The mode will have resonance with precession motion of trapped ions.
- The result agrees with NOVA2 result based on zero orbit width limit assumption.



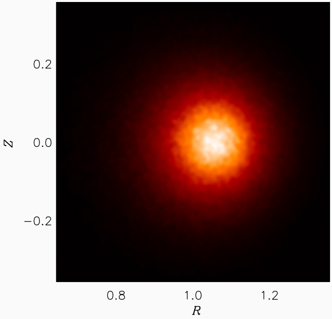
Particles loading and initialization

- Particles are loaded following the equilibrium distribution, with isotropic distribution in pitch angles.

$$f_0 = \frac{cH(v_0 - v)}{v^3 + v_c^3} \exp(-\psi/\psi_n)$$

- Two ways to load particles
  - In M3D-K, particles are loaded homogeneously in both real and momentum space. Each particle will then carry a f_0 that will appear in the weight equation.
  - In NIMROD, particles are loaded following f_0 through a Monte-Carlo sampling method.
  - We have implemented both methods in M3D-C1. The results obtained are close.

Loaded particle distribution using Monte-Carlo sampling



Particle pushing

$$\delta f \text{ method} \qquad \frac{\partial \delta f}{\partial t} + \dot{\mathbf{z}}_0 \cdot \frac{\partial \delta f}{\partial \mathbf{z}} = -\delta \dot{\mathbf{z}} \cdot \frac{\partial f_0}{\partial \mathbf{z}}$$

- In the linear run, the left part of equation is advanced by pushing markers following drift kinetic equations with equilibrium B fields only.

$$\frac{d\mathbf{X}}{dt} = \frac{1}{B^*}\left(v_\parallel \mathbf{B}^* + \mathbf{b}_0 \times \frac{\mu}{e}\nabla B_0\right)$$

$$m\frac{d\mathbf{v}_\parallel}{dt} = -\frac{1}{B^*}\mathbf{B}^* \cdot (\mu \nabla B_0)$$

$$\mathbf{B}^* = \mathbf{B}_0 + \frac{m v_\parallel}{e}\nabla \times \mathbf{b}_0, \qquad B^* = \mathbf{B}^* \cdot \mathbf{b}_0$$

Weight evolution

- We implement the same weight equation as in Kim (2008).

$$
\begin{aligned}
\frac{dw}{dt} = \frac{\delta \dot{f}}{f_0} &= -\delta \mathbf{v} \cdot \nabla f_0 - \dot{\epsilon} \partial_\epsilon f_0 \\
&= \frac{mF}{e\psi_n B^3} \left[\left(v_\parallel^2 + \frac{v_\perp^2}{2} \right) \delta \mathbf{B} \cdot \nabla B - v_\parallel \nabla \times \mathbf{B} \cdot \mathbf{E} \right] \\
&+ \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} + v_\parallel \frac{\delta \mathbf{B}}{B} \right) \cdot \frac{\nabla \psi - \rho_\parallel \nabla F}{\psi_n} \\
&+ \left[\frac{m}{B^3} \left(v_\parallel^2 + \frac{v_\perp^2}{2} \right) \mathbf{B} \times \nabla B + \frac{m v_\parallel^2}{B^2} \mu_0 \mathbf{J}_\perp \right] \cdot \mathbf{E} \frac{3v}{v^3 + v_0^3}
\end{aligned}
$$

- For homogeneous particle loading, dw/dt will be multiplied by equilibrium f_0.

Particle deposition

- Parallel and perpendicular pressure are calculated from the particles and coupled to momentum equation of MHD.
- We implement two different method for pressure deposition
  - δ-function deposition

$$\int \nu P_{\|} g d\mathbf{x} = \sum_i m v_{i,\|}^2 \nu(\mathbf{x}_i)$$

$$\int \nu P_{\perp} g d\mathbf{x} = \sum_i \mu_i B(\mathbf{x}_i) \nu(\mathbf{x}_i)$$

  - Shape function deposition

$$\int \nu P_{\|} g d\mathbf{x} = \sum_i \frac{1}{|S_i|} \int \nu S(\mathbf{x} - \mathbf{x}_i) m v_{i,\|}^2 g d\mathbf{x}$$

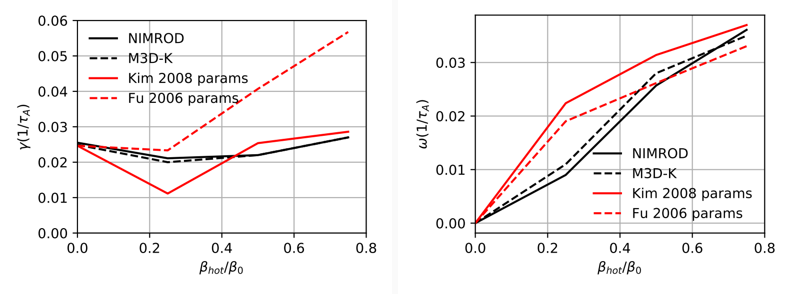
$$\int \nu P_{\perp} g d\mathbf{x} = \sum_i \frac{1}{|S_i|} \int \nu S(\mathbf{x} - \mathbf{x}_i) \mu_i B(\mathbf{x}_i) g d\mathbf{x}$$

Parameters for the fishbone test case

$$R/a = 2.8, \quad \beta_{total} = 0.08, \quad \psi_n = 0.25, \quad q_0 = 0.6, \quad q_a = 2.5$$

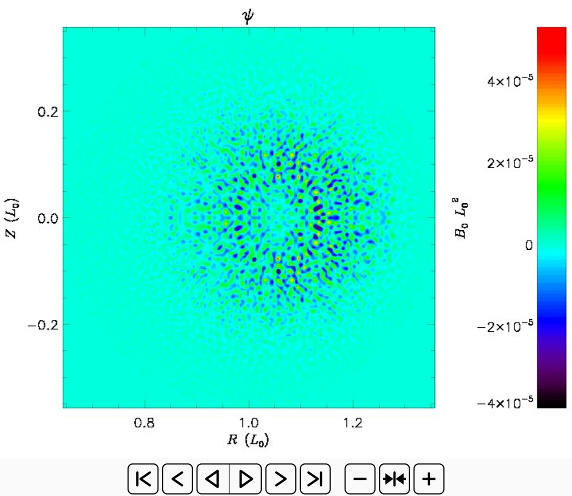
- Note that there are some differences between the hot particle parameters used in Fu (2006) and Kim (2008)
  - Fu (2006)
  $$\rho_h = v_0/(\Omega_h a) = 0.0125, \qquad v_0/v_A = 4$$
  - Kim (2008)
  $$\rho_h = v_0/(\Omega_h a) = 0.058, \qquad v_0/v_A = 1$$

Simulation result of the mode growth rate and real frequency

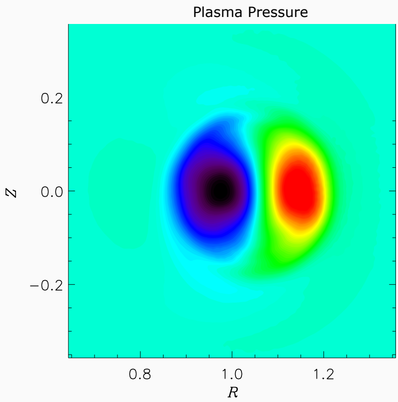


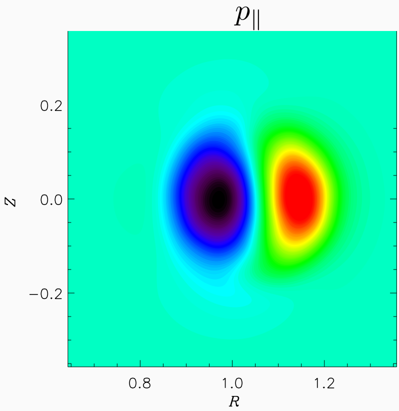
- Convergence study with varying *dt* and number of particles gives almost the same result.

Mode rotating due to the real frequency

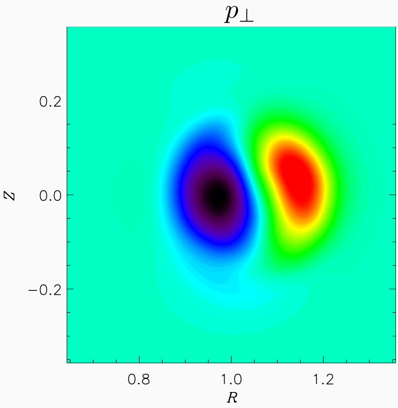


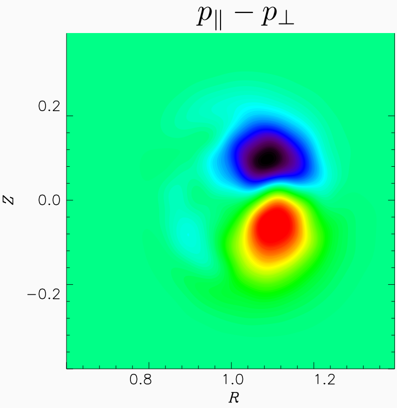
Pressure from plasma and energetic particles





Pressure from plasma and energetic particles (cont'd)





Accelerate particle pushing using GPU

- We have tried to use OpenACC to accelerate particle pushing based on current mesh-based particle structure.
- Initial result of optimization on Summit
  - On single thread CPU: 861.42s
  - On GPU (no data update between timestep): 2.03s
  - On GPU (data update between host and device at each timestep): 608.47s
- What we learned: It seems that mesh-based data structure of particles is a poor choice for GPU optimization due to particle adding and deleting.
  - There is no easy way to parallelize deletion of elements in array. Doing sequentially on GPU is extremely slow

Summary

- Basic framework of a kinetic module, including particle loading, particle pushing, weight calculation and pressure deposition, has been implemented in M3D-C1.
- Benchmark with M3D-K and NIMROD for the linear fishbone simulation shows qualitatively agreement, but there are some differences for both the growth rate and real frequencies.
    - We will try to use both M3D-K and NIMROD to redo the test case and do a more careful comparison.
- GPU accelerating of particle pushing is promising, but we need to change the particle data structure to optimize memory management and reduce communications.

Parallel pressure term in M3D-C1

$$\begin{aligned}\nabla \cdot (\alpha \mathbf{BB}) &= \mathbf{B} \cdot \nabla(\alpha \mathbf{B}) \\ &= \mathbf{BB} \cdot \nabla \alpha + \alpha \mathbf{B} \cdot \nabla \mathbf{B} \\ &= \mathbf{BB} \cdot \nabla \alpha + \frac{1}{2}\alpha \nabla B^2 - \alpha \mathbf{B} \times \nabla \times \mathbf{B}\end{aligned}$$

Parallel pressure term in M3D-C1

$$
\begin{aligned}
\nu\nabla\varphi\cdot\nabla\times R^2\nabla\cdot(\alpha\mathbf{BB}) &= R^2\nabla_\perp\nu\times\nabla\varphi\cdot\nabla\cdot(\alpha\mathbf{BB}) \\
&= [\alpha,\psi](\nu,\psi) + \alpha' R^{-2} F(\nu,\psi) \\
&+ \frac{1}{2}\alpha R^2 [B^2,\nu] \\
&+ \alpha\Delta^*\psi[\nu,\psi] + \alpha F[\nu,F]
\end{aligned}
$$

Parallel pressure term in M3D-C1

$$\nu R^2 \nabla\varphi \cdot \nabla \times R^2 \nabla \cdot (\alpha \mathbf{BB}) = \nu F[\alpha, \psi] + \nu F F \alpha' R^{-2} - \alpha\nu[\psi, F]$$

Parallel pressure term in M3D-C1

$$
\begin{aligned}
\nu\nabla_\perp \cdot \left[R^{-2}\nabla\cdot(\alpha\mathbf{BB})\right] &= -\nabla_\perp\nu\cdot\left[R^{-2}\nabla\cdot(\alpha\mathbf{BB})\right] \\
&= -R^{-2}[\alpha,\psi][\nu,\psi] - \alpha' R^{-4}F[\nu,\psi] \\
&- \frac{1}{2}\alpha R^{-2}(\nu, B^2) \\
&+ R^{-4}\alpha\Delta^*\psi(\nu,\psi) + FR^{-4}\alpha(\nu,F)
\end{aligned}
$$