Status of the M3D-C^1 hybrid kinetic energetic ion module

Josh Breslau

Acknowledgments:
Steve Jardin, Nate Ferraro, Guoyong Fu, Kaushik Kalyanaraman

CEMM Meeting
San Jose
October 30, 2016

Outline

- Specification of task
- Implementation
  - Particle loading
  - Particle push
  - Pressure deposition
  - Fluid coupling
  - I/O & Diagnostics
- Summary & next steps

Specification of task

- Goal is to add the option to advance an ensemble of fast particles on the M3D-C1 domain using particle-in-cell (PIC) techniques.
  - Start with beam ions.
  - Particle time step may be sub-cycled relative to fluid step.
  - Use high-order integration for accuracy.

- Support multiple physics models
  - Full orbit (Lorentz force, non-relativistic)
  - Drift-kinetic: advance guiding center equation of motion, conserving constants of motion

$$\varepsilon = \tfrac{1}{2} M_i U^2 + \mu B \tag{1}$$

$$\mu = \tfrac{1}{2} M_i v_\perp^2 / B \tag{2}$$

$$P_\varphi = e\psi + M_i R U B_\varphi / B \tag{3}$$

- No collisions.

Drift kinetic equations of motion

$$\left\{\begin{aligned}
\dot{\mathbf{X}} &= \frac{1}{B^{**}}\left[U\mathbf{B}^{*} + \hat{\mathbf{b}}\times\left(\mu\nabla B/q - \mathbf{E}\right)\right] && (4)\\
\dot{U} &= -\frac{q}{mB^{**}}\mathbf{B}^{*}\bullet\left(\mu\nabla B/q - \mathbf{E}\right) && (5)\\
\dot{\mu} &= 0 && (6)
\end{aligned}\right.$$

where

$$\mathbf{B}^{*} \equiv \mathbf{B} + \frac{mU}{q}\nabla\times\hat{\mathbf{b}} \tag{7}$$

and

$$B^{**} \equiv \mathbf{B}^{*}\bullet\hat{\mathbf{b}} \tag{8}$$

The δf method

To minimize noise for linear problems, represent the energetic population distribution function as $f(\mathbf{x},\mathbf{v},t)=f_0(\mathbf{x},\mathbf{v}) + \delta f(\mathbf{x},\mathbf{v},t)$, where the former is an analytic function and only the latter is constructed from the particle ensemble, with weights w_i evolving from zero according to

$$\dot{w}_i = -\frac{\left(1-w_i\right)}{f_0}\left(\mathbf{V}_1 \bullet \nabla f_0 - q\mathbf{V}_0 \bullet \mathbf{E}\frac{\partial f_0}{\partial \varepsilon}\right) \tag{9}$$

where

$$\mathbf{V}_0 \equiv U\hat{\mathbf{b}} + \mathbf{V}_D \tag{10}$$

$$\mathbf{V}_D \equiv \frac{1}{qB^3}\left(mU^2 + \mu B\right)\mathbf{B}\times\nabla B + \frac{mU^2}{qB^2}\mathbf{J}_\perp \tag{11}$$

$$\mathbf{V}_1 \equiv \frac{\mathbf{E}\times\mathbf{B}}{B^2} + \frac{U\delta\mathbf{B}}{B} \tag{12}$$

$$\mathbf{B} = \mathbf{B}_0 + \delta\mathbf{B} \tag{13}$$

Pressure coupling

- Assumes hot ion density is negligible but β is significant:

$$n\left(\frac{\partial \mathbf{V}}{\partial t}+\mathbf{V}\bullet\nabla\mathbf{V}\right)=\mathbf{J}\times\mathbf{B}-\nabla p-\nabla\bullet\Pi_{visc}-\nabla\bullet\Pi_{hot} \tag{14}$$

where

$$\Pi_{hot}\equiv p_{\|}\hat{\mathbf{b}}\hat{\mathbf{b}}+p_{\perp}\left(\mathbf{I}-\hat{\mathbf{b}}\hat{\mathbf{b}}\right)=\left(p_{\|}-p_{\perp}\right)\hat{\mathbf{b}}\hat{\mathbf{b}}+p_{\perp}\mathbf{I} \tag{15}$$

- Applying the Galerkin finite element method, construct scalar pressure fields by weighted integration over delta-function sources and solve:

$$\int \nu_i\left(R,\varphi,z\right)\nu_j\left(R,\varphi,z\right)p_{\|j}d^3V=m\sum_{k=1}^{N}w_kU_k^2\nu_i\left(R_k,\varphi_k,z_k\right) \tag{16}$$

$$\int \nu_i\left(R,\varphi,z\right)\nu_j\left(R,\varphi,z\right)p_{\perp j}d^3V=m\sum_{k=1}^{N}w_k\mu_kB\left(R_k,\varphi_k,z_k\right)\nu_i\left(R_k,\varphi_k,z_k\right) \tag{17}$$

Outline

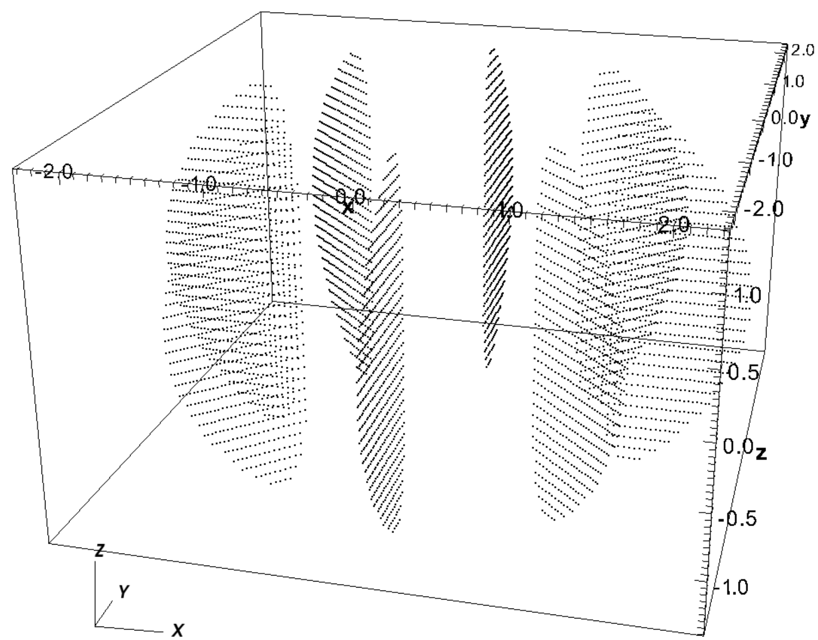
- Specification of task
- Implementation
  - Particle loading
  - Particle push
  - Pressure deposition
  - Fluid coupling
  - I/O & Diagnostics
- Summary & next steps

Particle loading

- Implemented in subroutine init_particles().

- Physical space initialization: uniform over (R,φ,z) cube with Jacobian to ensure uniformity over d^3x. Particles outside mesh rejected.

Sample spatial distribution over four-partition KSTAR mesh:
5840 / 8192 = 32 x 8 x 32 *particles deposited.*



Velocity space initialization

- Original implementation: uniform on 2D grid of $0<E\leq E_{max}=10$ keV; $0\leq\lambda\leq\pi$.

- Coordinates transformed to (v_R, v_φ, v_z) (full-orbit) or ($v_{||}$, μ/q) (drift-kinetic).

- New implementation: use Jacobian to initialize distribution uniformly on d^3v, with $0<|\mathbf{v}|<\mathrm{sqrt}(2E_{max}/m)$.

Equilibrium particle distribution

- Maxwellian implemented:

$$f_0(\mathbf{x},\mathbf{v}) = \left(\frac{1}{v_{th}\sqrt{2\pi}}\right)^3 e^{-\frac{v^2}{2v_{th}^2}} \tag{18}$$

- Slowing-down beam distribution planned:

$$f_0(\mathbf{x},\mathbf{v}) = \frac{P_0 \exp\left(P_\varsigma / \psi_0\right)}{\varepsilon^{3/2} + \varepsilon_0^{3/2}}, \tag{19}$$

where

$$P_\varsigma = g(\psi)\,\rho_{\|} - \psi$$

is the canonical poloidal momentum.

Notes on particle loading

- Particle module initialization loads a two-layer ghost mesh for MPI particle handoff bookkeeping.
  - Requires reallocation and redefinition of intermediate coefficient arrays.
  - Ghosts must be destroyed and arrays redefined again before resumption of fluid advance.
- Electric field components must be explicitly computed prior to particle advance.
- Global particle time step is a predetermined fraction (full orbit) or multiple (drift-kinetic) of the minimum gyroperiod.

Particle advance overview

- Subroutine advance_particles() advances all particle positions, velocities by a specified time increment, using given 2D (real or complex) or 3D fields, subcycling as necessary.

- Hierarchical organization of particles by element, element ensemble, OMP thread, and MPI/mesh partition allows good optimization.

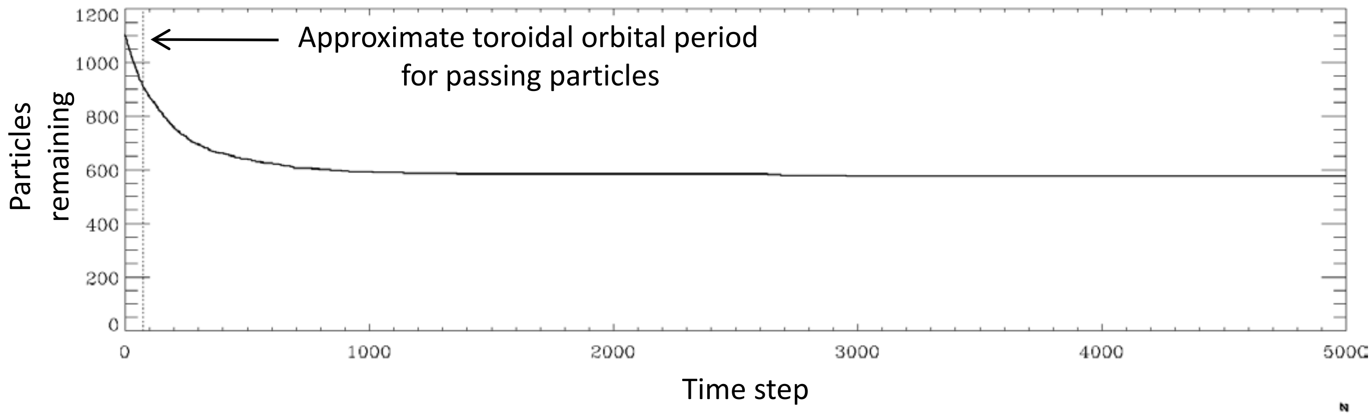
- 4^{th}- and 5^{th}-order Runge-Kutta ODE integration are available; both show good energy, P_φ conservation over many time steps.

Particle advance example

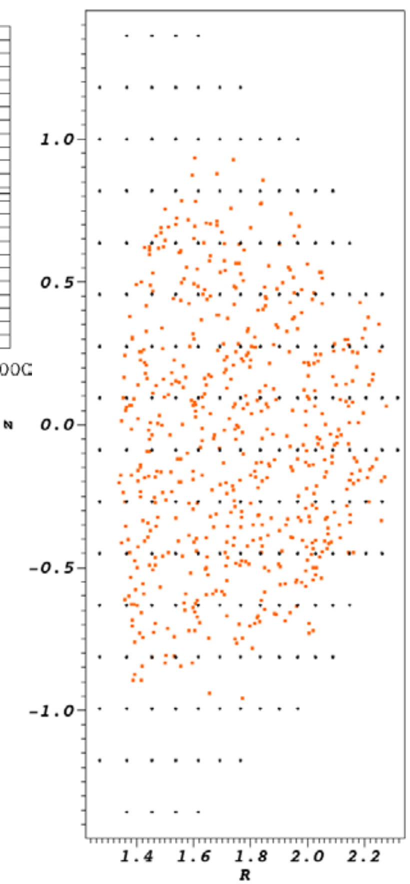
- Coarse KSTAR mesh (776 elements in four domains).
- Initial distribution: 16 x 1 x 16 x 1 x 6 = 1536 candidates.
- 1104 candidates accepted, 432 rejected.
- Particles/cell range from 1.24 to 1.70; overall avg = 1.42.
- Drift-kinetic formulation, 4th-order RK stepping, 2D complex fields.
- 5000 steps, dt (drift-kinetic)=10^{-7} s $\approx$ 5 gyroperiods.
- Execution time: 123.4 s on four PEs (one thread/PE).
- 557 particles remain by end of run.
- Max $\delta KE/KE_0 = 9.4\times10^{-4}$; mean = 7.8×10^{-7}; rms = 1.15×10^{-4}.
- Max $\delta P_\varphi / P_{\varphi 0} = 2.7\times10^{-4}$; mean = -1.3×10^{-6}; rms = 4.60×10^{-5}.

- 5th-order RK: 136 s. Max $\delta KE/KE_0 = 3.7\times10^{-4}$; rms = 6.05×10^{-5}.
  Max $\delta P_\varphi / P_{\varphi 0} = 1.9\times10^{-4}$; rms = 2.20×10^{-5}.

Particles on open field lines exit promptly



M3D-C1 mesh bounding box



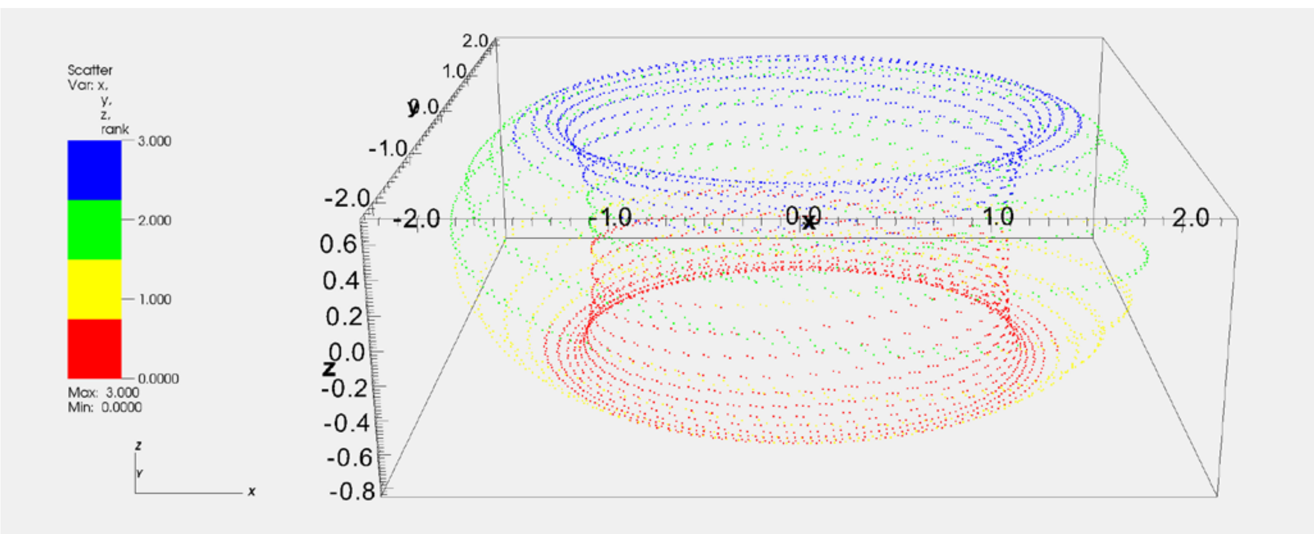
Initial 1104 particles

Final 557 particles

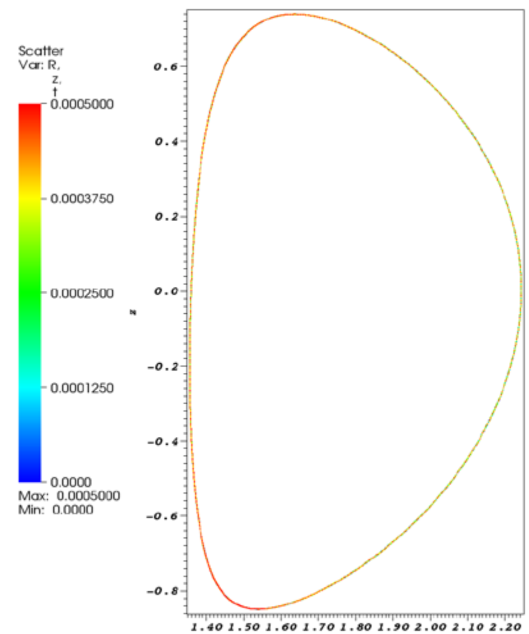
This phenomenon can exacerbate the load imbalance for a domain-decomposed mesh!

Sample passing orbit ($\lambda_0 = 10^{-5}$)

- Initial KE=9.9995466e+03 eV; final=9.9995466e+03.
- Initial P_φ=-0.52531 eV s; final=-0.52530.



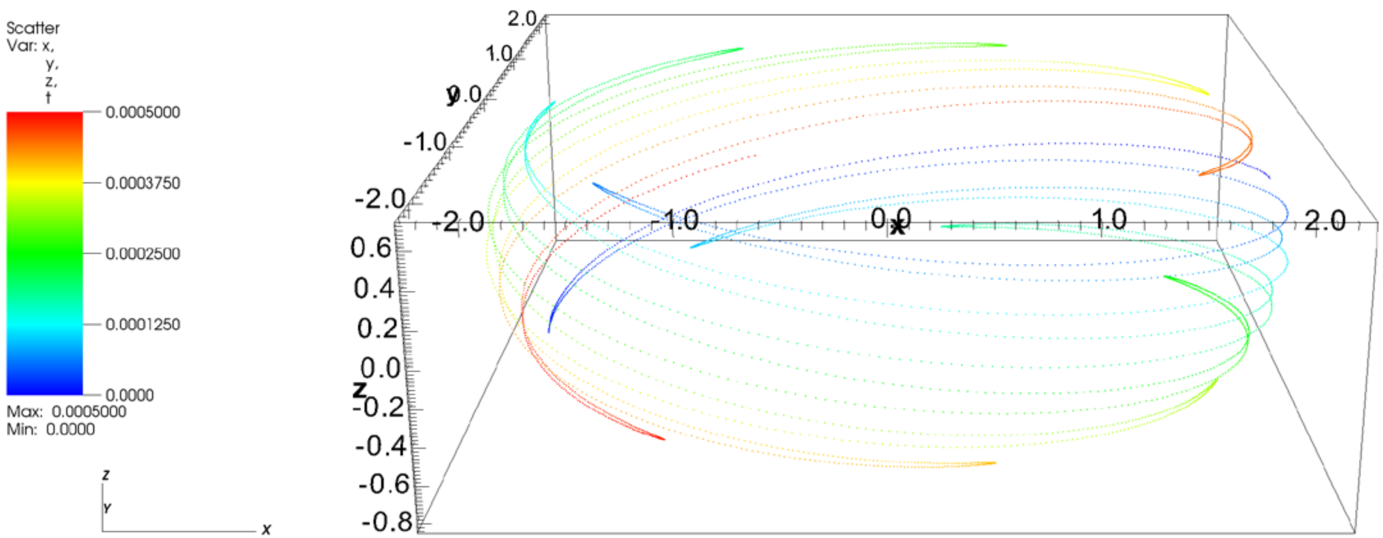
R-z plane projection



Colors indicate MPI rank.

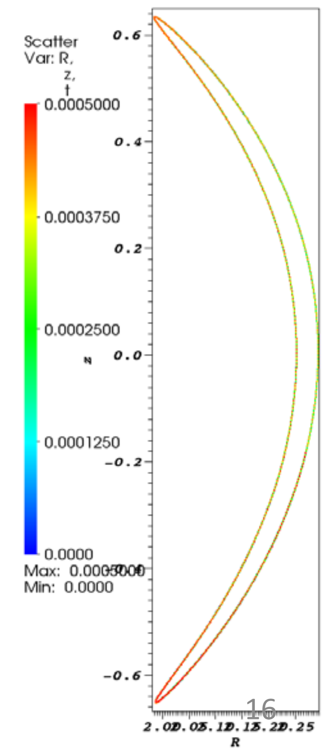
Sample trapped orbit ($\lambda_0 = 3\pi/5$)

- Initial KE=9.9995e+03 eV; final=9.9990e+03.
- Initial P_φ=-0.476633; final=-0.476630.



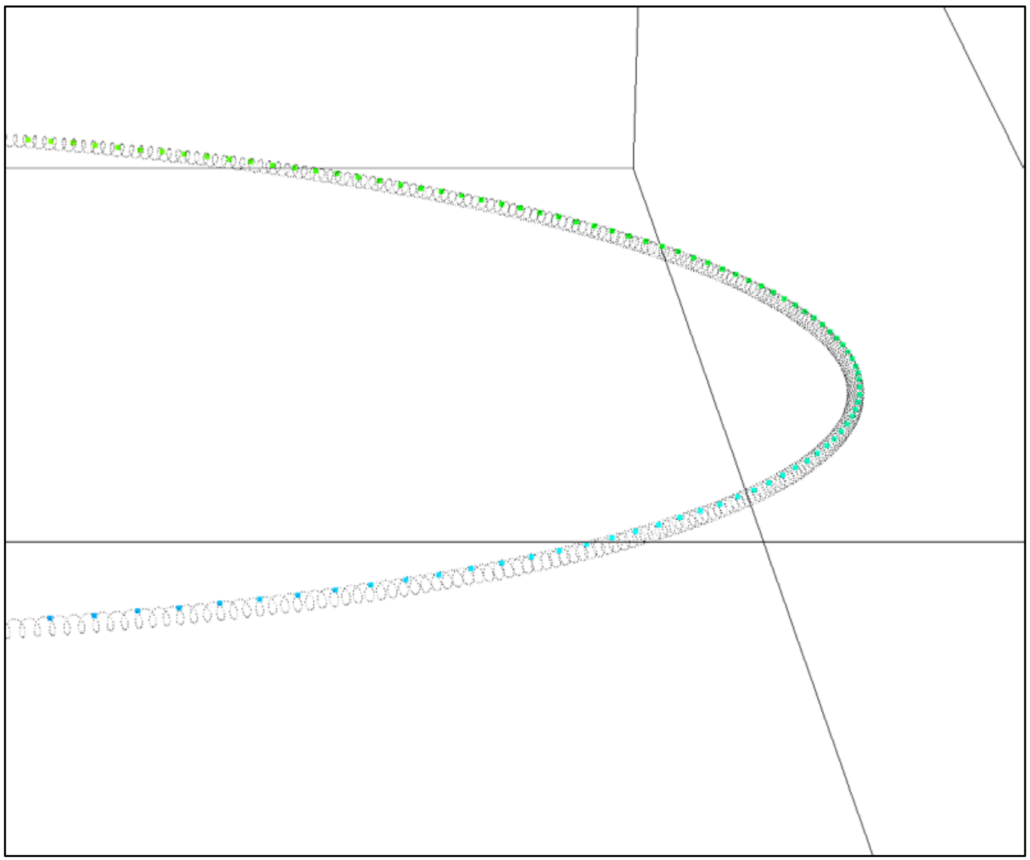
Colors indicate advancing time.

R-z plane projection



Drift-kinetic/full-orbit comparison

- Full-orbit: 20,480 steps, pdt (drift-kinetic)=10^{-10} s ≈ 0.005 gyroperiods.
- Execution time: 19:50.6 s for 1104 particles on four kruskal PEs.



Detail

- KE conservation for full-orbit is good, but angular momentum conservation is relatively poor; consider alternate integrators.
- A drift-kinetic step is about twice as fast as a full-orbit step, and can be around 1600x larger for comparable accuracy.

Preparing to scale up

- With sufficient particles for accurate phase space resolution, the particle advance could dominate the overall M3D-C1 execution time. If the particle advance scales well to 10,000+ cores, M3D-C^1 should scale well too.

- The current bottleneck is load imbalance: particle number differs dynamically between cores.

- Two possible solutions are under consideration:
  - Maintain a copy of global mesh and relevant global field data on every shared-memory node.
    - Pros: Eliminates need for ghost layers, memory needs are not excessive.
    - Cons: Redundancy or all-to-all communication; will not scale to arbitrary mesh size.

  - Use differing domain decompositions for fluid/particle sections, dynamic load balancing.
    - Pros: All communications are nearest-neighbor, should win at large enough scale.
    - Cons: Complexity, rebalancing overhead.

Pressure deposition

- RHS vectors for $p_{||}$, $p_\perp$ are computed by integrating over particle delta functions within each element.
  - Makes use of SCOREC routine vector_insert_block().
  - Element loop could be multithreaded, but race conditions could be tricky, and time saved appears to be small compared to particle push.

- LHS vectors computed by subroutine solve_pi_tensor(), which inverts mass matrix to solve for each component.
  - Very fast (time is independent of particle count).
  - Requires deletion of ghost mesh, recalculation of coefficient arrays, which is awkward.

Fluid coupling: velocity projections

- All terms in (14) must be projected to the M3D-C^1 velocity representation:

$$\mathbf{V} = R^2\nabla U \times \nabla\varphi + \omega R^2 \nabla\varphi + R^{-2}\nabla_\perp \chi$$

- The appropriate projection operators to extract the scalar components of the momentum equation are, respectively

$$U : \iint d^2R\nu_i \nabla\varphi \bullet \nabla_\perp \times R^2$$

$$\omega : \iint d^2R\nu_i R^2 \nabla\varphi \bullet$$

$$\chi : -\iint d^2R\nu_i \nabla_\perp \bullet R^{-2}$$

Preliminary definitions

In terms of the M3D-C1 scalar fields,

$$\mathbf{B} = \nabla\psi \times \nabla\varphi - \nabla_\perp f' + F\nabla\varphi = \nabla\psi \times \nabla\varphi - \nabla f' + F^*\nabla\varphi$$

$$\mathbf{J} = \nabla \times \mathbf{B} = \nabla F^* \times \nabla\varphi + \frac{1}{R^2}\nabla_\perp \psi' - \Delta^*\psi\nabla\varphi$$

Also define

$$\alpha \equiv \frac{p_\| - p_\perp}{B^2}, \quad \beta \equiv p_\perp$$

with Poisson bracket $\left[f, g \right] \equiv \nabla\varphi \bullet \left(\nabla f \times \nabla g \right)$

and inner bracket $\left(f, g \right) \equiv \nabla_\perp f \bullet \nabla_\perp g$

Poloidal velocity stream function U

$$\nu_i \nabla\varphi \bullet \nabla_\perp \times \left\{ R^2 \nabla \bullet \left[\alpha \mathbf{BB} + \beta \mathbf{I} \right] \right\} = R^2 \nabla_\perp \nu_i \times \nabla\varphi \bullet \left\{ \nabla \bullet \left[\alpha \mathbf{BB} \right] \right\} + R^2 \left[\beta, \nu_i \right]$$

$$= R^2 \left[\beta, \nu_i \right] + \tfrac{1}{2} \alpha R^2 \left[B^2, \nu_i \right] + \left[\alpha, \psi \right] \left(\nu_i, \psi \right) + \alpha \Delta^* \psi \left[\nu_i, \psi \right]$$

$$+ R^2 \left[\alpha, \psi \right] \left[\nu_i, f' \right] - \left(\alpha, f' \right) \left(\nu_i, \psi \right) - \alpha \Delta^* \psi \left(\nu_i, f' \right)$$

$$- \left(\alpha, f' \right) R^2 \left[\nu_i, f' \right] + \alpha' R^{-2} F \left(\nu_i, \psi \right) + \alpha R^{-2} F \left(\nu_i, \psi' \right)$$

$$+ \alpha' F \left[\nu_i, f' \right] + \alpha F \left[\nu_i, F \right]$$

Toroidal angular velocity ω

$$\nu_i R^2 \nabla\varphi \bullet \nabla \bullet \left[\alpha \mathbf{BB} + \beta \mathbf{I}\right] = \nu_i R^2 \nabla\varphi \bullet \nabla \bullet \left[\alpha \mathbf{BB}\right] + \nu_i \beta'$$

$$= \nu_i \beta' + \alpha \nu_i B B' - \frac{1}{R^2} \alpha \nu_i \left(\psi, \psi'\right) + \nu_i F\left[\alpha, \psi\right] - \alpha \nu_i \left[\psi, F^*\right]$$

$$-\nu_i F\left(\alpha, f'\right) - \alpha \nu_i \left(F^*, f'\right) + \nu_i F F \alpha' R^{-2} - \alpha \nu_i \left[\psi', f'\right]$$

Poloidal compressible velocity potential χ

$$\nu_i \nabla_\perp \bullet \left\{ R^{-2} \nabla \bullet \left[\alpha \mathbf{B}\mathbf{B} + \beta \mathbf{I} \right] \right\} = -\nabla_\perp \nu_i \bullet \left\{ R^{-2} \nabla \bullet \left[\alpha \mathbf{B}\mathbf{B} \right] \right\} - R^{-2} \left(\nu_i, \beta \right)$$

$$= -R^{-2} \left(\nu_i, \beta \right) - \tfrac{1}{2} \alpha R^{-2} \nabla_\perp \nu_i \bullet \nabla B^2 - R^{-2} \left[\alpha, \psi \right] \left[\nu_i, \psi \right] + R^{-4} \alpha \Delta^* \psi \left(\nu_i, \psi \right)$$

$$+ R^{-2} \left(\alpha, f' \right) \left[\nu_i, \psi \right] + R^{-2} \left[\alpha, \psi \right] \left(\nu_i, f' \right) + R^{-2} \alpha \Delta^* \psi \left[\nu_i, f' \right]$$

$$- \alpha' R^{-4} F \left[\nu_i, \psi \right] - R^{-4} \alpha F \left[\nu_i, \psi' \right] - R^{-2} \left(\alpha, f' \right) \left(\nu_i, f' \right)$$

$$+ F \alpha' R^{-4} \left(\nu_i, f' \right) + F R^{-4} \alpha \left(\nu_i, F^* \right)$$

I/O & Diagnostics

- The particle_test() subroutine writes out the entire trajectory of a predetermined subset of particles, tracking KE and P_φ.
  - Trajectory text files are compatible with VisIt Point3D format for scatter plotting.

- Parallel HDF5 is used to dump the entire particle distribution at a given time to an output file, including positions, velocities, and weights.
  - Utilities exist to extract position data from these to a text file, enabling comparisons and plotting with VisIt.
  - Utilities to visualize velocity distributions, pressure tensor components are still under development.

- Checkpointing of particle distribution will be based on HDF5.

Outline

- Specification of task
- Implementation
  - Particle loading
  - Particle push
  - Pressure deposition
  - Fluid coupling
  - I/O & Diagnostics
- Summary & next steps

Summary

- Particle initialization, full-*f* push, and I/O now working, tested and highly optimized for 2D complex version.
- Pressure deposition, δf push implemented; need testing.
- Fluid coupling in progress.

Next steps

- Verify hot ion pressure tensor component deposition algorithm, adjust normalization as necessary.
- Implement explicit $\nabla \bullet \Pi_{\text{ion}}$ term in fluid momentum equation.
- Implement particle checkpoint restart.
- Verify 2D complex version of kinetic code with fishbone test case.
- Develop visualization tools for velocity distributions.
- Generalize to 3D, nonlinear cases.