Improving the PIC method: Moment-preserving particle resampling and higher order velocity-space particle-grid mapping in the global total-f gyrokinetic particle-in-cell code XGC

A. Mollén[1], V. Carey[2], M. F. Adams[3], M. G. Knepley[4], R. Hager[1], J. Dominski[1] and C. S. Chang[1] + XGC/WDMApp team

[1] Princeton Plasma Physics Laboratory, Princeton, NJ, USA
[2] Theodon Consulting LLC Superior, CO, USA
[3] Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[4] State University of New York at Buffalo, NY, USA

Outline

- Motivation
- Brief introduction to the XGC code
- Particle resampling in a PIC code
- Updated velocity-space interpolation using a pseudo-inverse
- Summary

Motivation

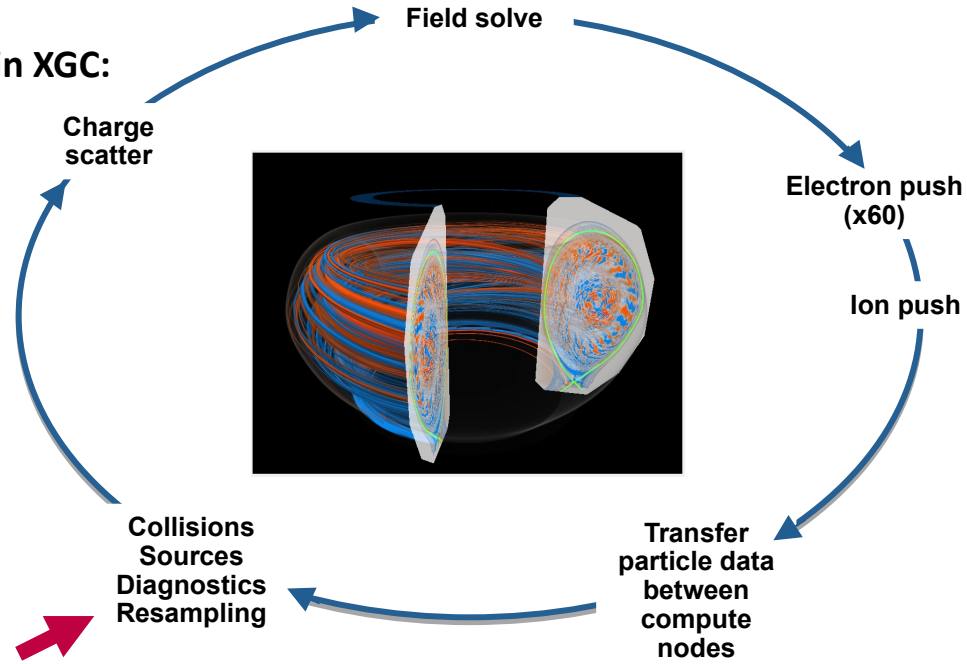
- Main target of the XGC code is to study boundary plasmas that may contain a steep edge pedestal.
- XGC is being coupled to the core codes GENE and GEM in the ECP-WDMApp project.
- Here we discuss two significant mathematical progresses that allow pedestal simulation and code coupling to be more accurate and efficient:
  - Moment-preserving particle resampling technique to reduce the load-imbalance and enhance simulation accuracy.
  - A novel velocity-space interpolation using the pseudo-inverse and particle resampling methods to remove the numerical heating in the steep edge pedestal.

Outline

- Motivation
- Brief introduction to the XGC code
- Particle resampling in a PIC code
- Updated velocity-space interpolation using a pseudo-inverse
- Summary

Introduction to the X-point gyrokinetic code (XGC)

- The global total-f gyrokinetic particle-in-cell (PIC) code XGC xgc.pppl.gov
  - Study transport in magnetic fusion plasmas.
  - Neoclassical and turbulent dynamics + neutral particle transport.
  - Treats full plasma volume including scrape-off layer across separatrix.
  - Code has been converted from Fortran to C++ with Kokkos/Cabana for GPUs.
  - **Time stepping in XGC:**



Focus of this presentation

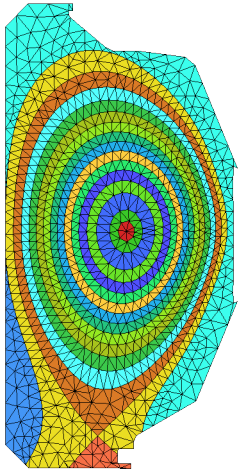
5D phase-space grid in XGC

- Typical size for XGC turbulence simulations
  - 0.1M- 1M configuration-space vertices per poloidal plane.
  - $N_{plane} \sim 16 - 128$.
  - 10 000 particles/vertex: trillions of total particles for ITER.
  - Velocity space grid: $N_{v_{||}} \times N_{v_\perp} \sim 40 \times 40$.

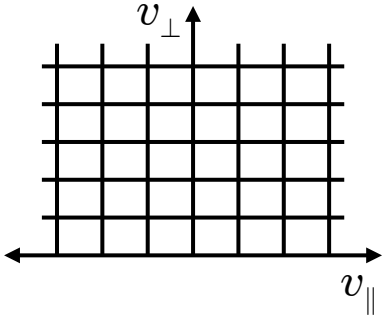
- Why Coulomb collisions on 5D grid?
  - Cost of particle-particle collision scheme $\geq O(N_{particles}$ or $N^2_{particles})$.
  - Cost of grid-based scheme $\sim O(N_{vertices})$ + interpolation cost.
    $N_{vertices} << N_{particles}$ → much cheaper.
  - The grid is also used for coupling to other codes in the Exascale Computing Project.

- Exact energy conservation (avoiding numerical heating) is difficult to obtain in particle-velocity grid coupling (interpolation). Other schemes can face similar issues [Alves et al, Phys. Rev. E (2021)].

Unstructured triangular mesh in 3D configuration space



Uniform rectangular grid in 2D velocity space



Distribution function in XGC total-f

- The total distribution function of species s is divided as [Ku et al, PoP (2018)]

$$f_s = f_0 + f_p = f_a + f_g + f_p$$

- Slowly varying part $f_0 = f_a + f_g$
  - f_a analytic function (Maxwellian)
  - f_g represented on the 5D phase-space continuum grid
- Fast-varying physics (particle orbital, instability and turbulent dynamics)
  - f_p represented by marker particles, with two weights:
    time-independent full-f weight and time dependent weight
- No restriction on relative magnitude of f_p compared with f_0.
- A fraction of f_p can be transferred over to f_g and f_a during a time step to control particle weight growth.

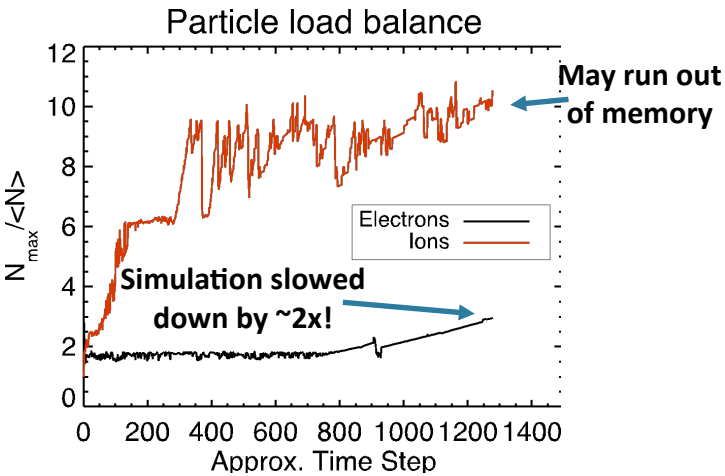
Outline

- Motivation
- Brief introduction to the XGC code
- Particle resampling in a PIC code
- Updated velocity-space interpolation using a pseudo-inverse
- Summary

Particle resampling - motivation

- Particle-in-cell codes have advantages, e.g.
  - Easily parallelizable and extremely scalable.
  - Good for high-dimensional problems and complicated boundary conditions.
- There are disadvantages for a long-time simulation
  - Time dependent sampling noise → error accumulation.
  - Complicated load-balancing in parallel simulation.
- Particle resampling can improve both accuracy and load balancing
  - Ensure good particle coverage in phase-space.
  - Homogenize the particle workload over compute nodes.

  → Allows longer time simulation.

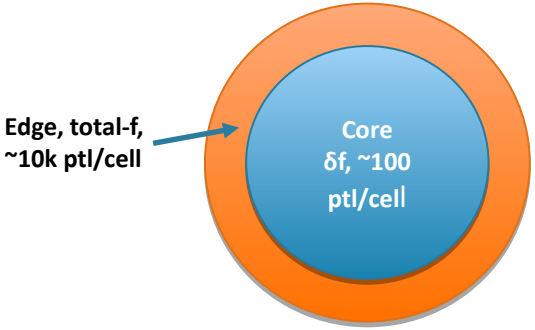


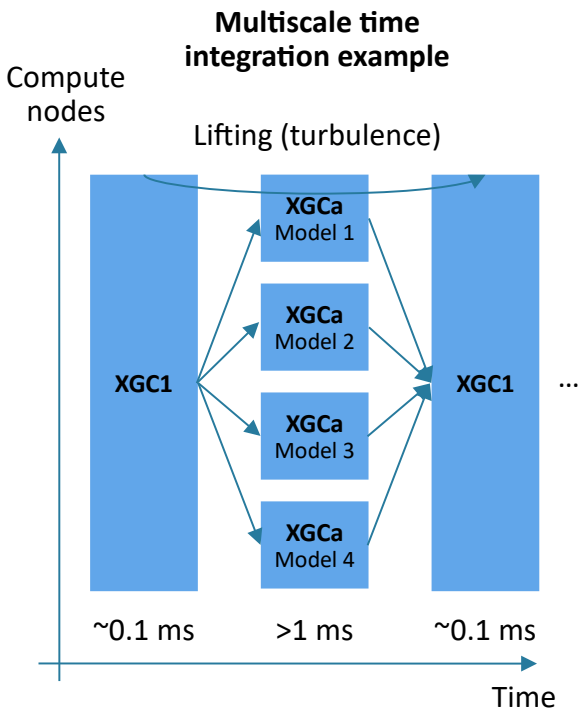
Particle resampling has many applications in XGC

- Load balancing
- Noise reduction (noise = variance in marker particle weights)
- Enable long time-scale simulation
- Quiet start (mitigate initial transient from particle loading)

- Multiscale time integration
  combining full-scale turbulence simulation with
  axisymmetric simulation and anomalous transport model

- Core-edge coupling of δf (core) and total-f (edge) code
  [Dominski et al, PoP (2021)]





- **Velocity space mapping with pseudo-inverse**

How the particle resampling in XGC works

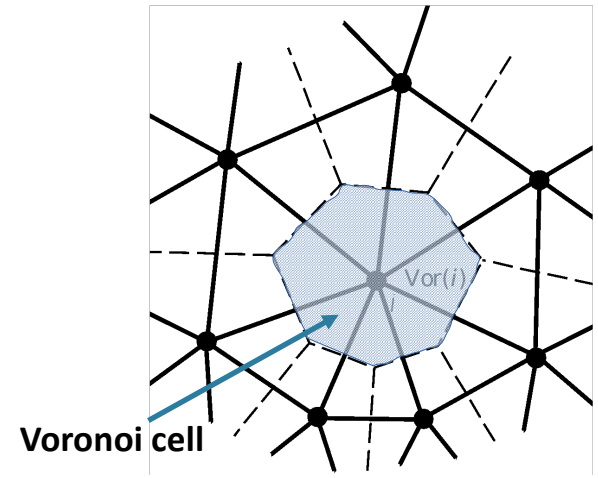
Moment-preserving particle resampling [Faghihi et al, JCP (2020)]

- Sort marker particles into phase-space bins
  - Voronoi cells from 3D mesh nodes + halfway between poloidal planes.
  - 2D velocity space cells.
- Up/down/re-sample particles in each bin
  - Add, remove or maintain particles in the bin.
  - Make adjustments to ensure that the total phase-space volume of the particles in every bin is preserved.
- Compute new particle weights through variance minimization subject to a set of constraints e.g. mass, momentum, kinetic energy conservation.
- Optimization is solved using constrained quadratic programming

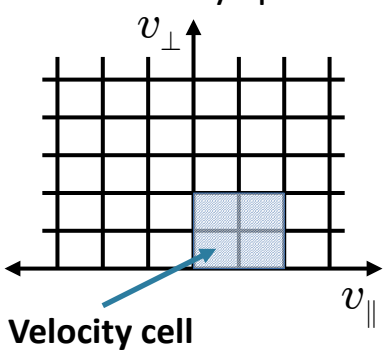
$$\text{minimize} \quad \frac{1}{2}\boldsymbol{w}^T \underline{\boldsymbol{Q}} \boldsymbol{w} + \boldsymbol{c}^T \boldsymbol{w}$$

$$\text{subject to} \quad \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} \boldsymbol{w} \begin{cases} = \boldsymbol{b}_1 \\ \leq \boldsymbol{b}_2 \end{cases}$$

3D configuration space



2D velocity space



How the particle resampling in XGC works

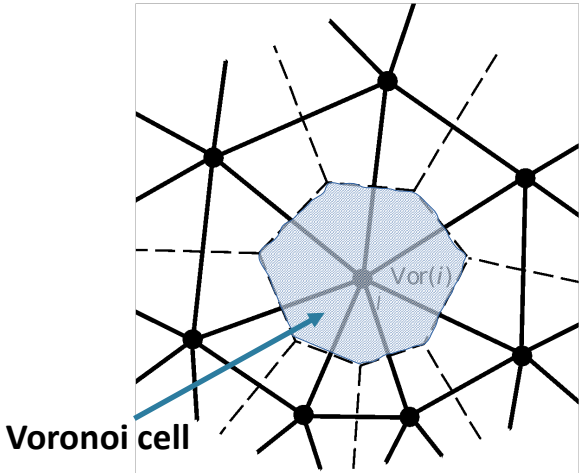
Moment-preserving particle resampling [Faghihi et al, JCP (2020)]

- There are several degrees of freedom (room for optimization)
  - Definition of phase-space bins.
  - Choice of moments to preserve, e.g. avoid ill-conditioned matrices from similar constraints.
  - How to sample marker particles, e.g. from which distribution.
  - How to add/remove marker particles in up/down-sampling.

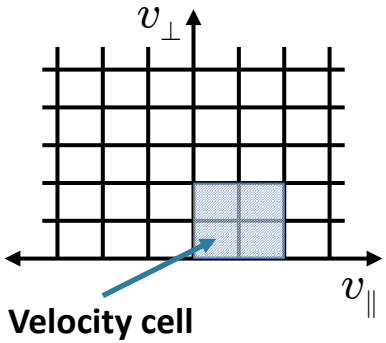
- The resampling works well for the typical use case of every 10-20 time step
  - All bins are independent so perfectly parallelizable.
  - Good performance with combined process and thread parallelism (MPI + OpenMP).

- Future work
  - Convert module from Fortran to C++.
  - Put on GPUs.
  - Machine learning.

3D configuration space



2D velocity space



Working well, but not perfect

- In some special cases we want to perform aggressive upsampling, but this seems to cause a numerical instability.

Show animation of time evolution of perturbed potential without vs with aggressive resampling

- Possible solutions
  - Marker particle placement in bins.
  - Choice of moments to preserve and how to sample marker particles.
  - Nonlinear programming (expensive).

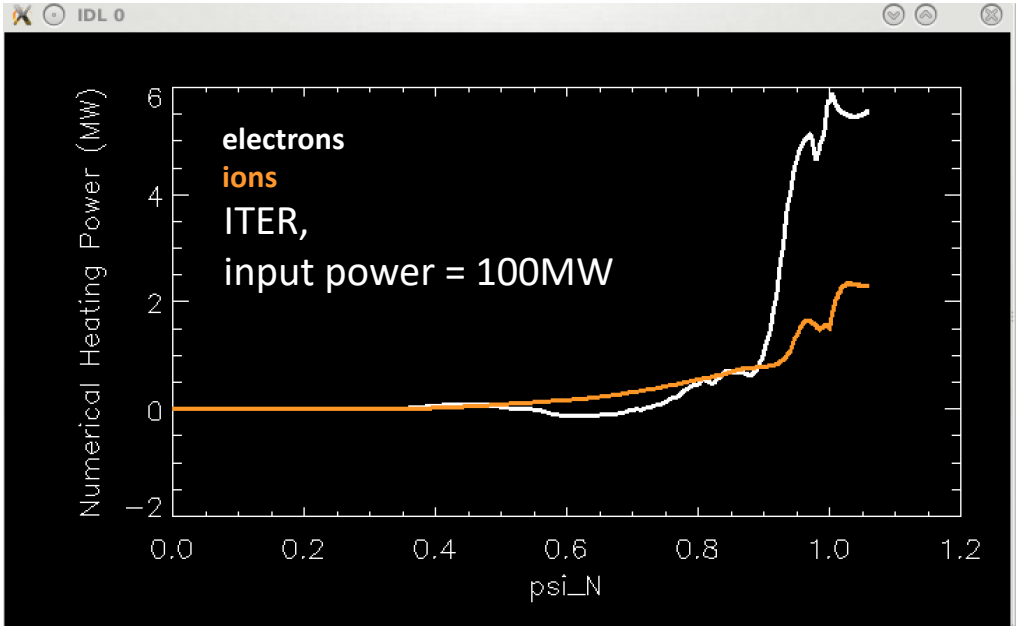
Outline

- Motivation
- Brief introduction to the XGC code
- Particle resampling in a PIC code
- Updated velocity-space interpolation using a pseudo-inverse
- Summary

Energy conservation in velocity-space interpolation

- Dissipative operations (collisions) and sources on 5D grid can have arbitrarily small conservation errors by stricter tolerances in iterative solvers.
- Bilinear interpolation error can be controlled by increasing the number of marker particles [Hager et al, JCP (2016)].
- Sometimes exact energy conservation could be important, e.g. mitigate numerical heating in edge pedestal in long simulations: Example case, numerical electron heating up to 6 % of experimental input power.

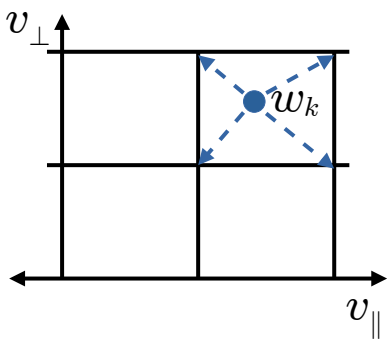
Radially integrated numerical heating power in the full-current ITER simulation

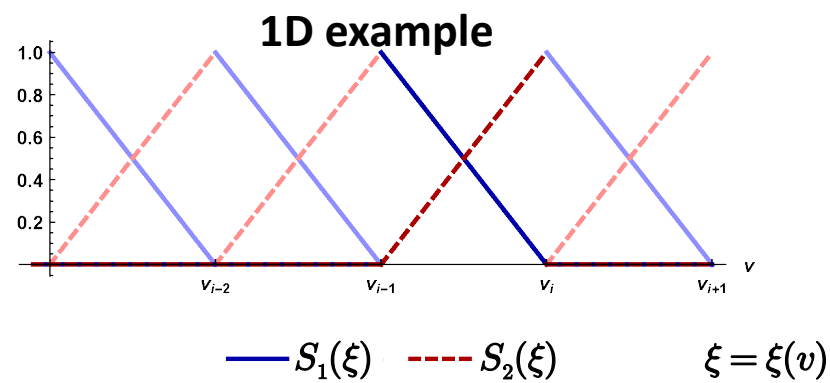


Mapping marker particles ↔ grid

- XGC implements a bilinear mapping in velocity space [Yoon, Chang, PPCF (2014)].
  - Marker particle weights are transferred to the grid using linear shape functions. ⇒ Fraction ∝ inverse linear distance between particle and surrounding 4 grid points.
  - Conserves particle number and momentum but not energy at the same level.

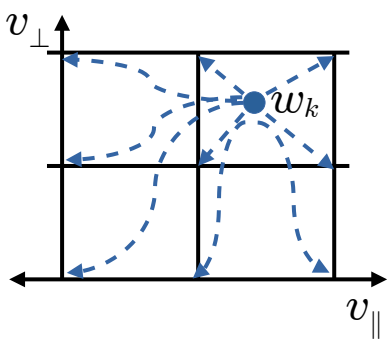
1st order

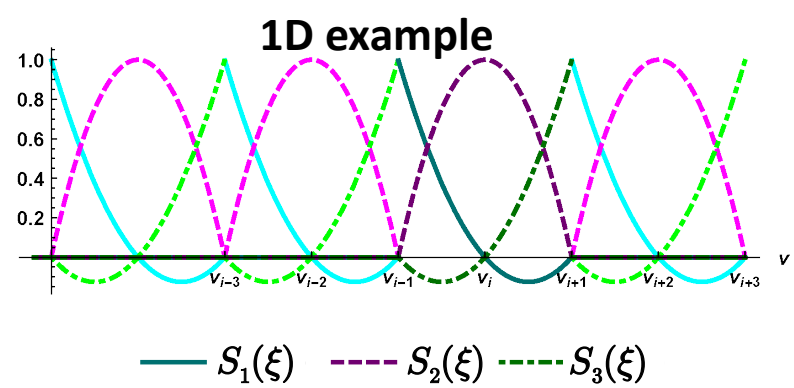




- We need quadratic mapping (or higher order) to also conserve energy.

2nd order





Quadratic mapping marker particles ↔ grid

- **For particles → grid mapping:**
  Quadratic Lagrange P_2 shape functions

$$S_1(\xi) = 1 - 3\,\xi + 2\,\xi^2$$
$$S_2(\xi) = 4\,\xi - 4\,\xi^2$$
$$S_3(\xi) = -\,\xi + 2\,\xi^2$$

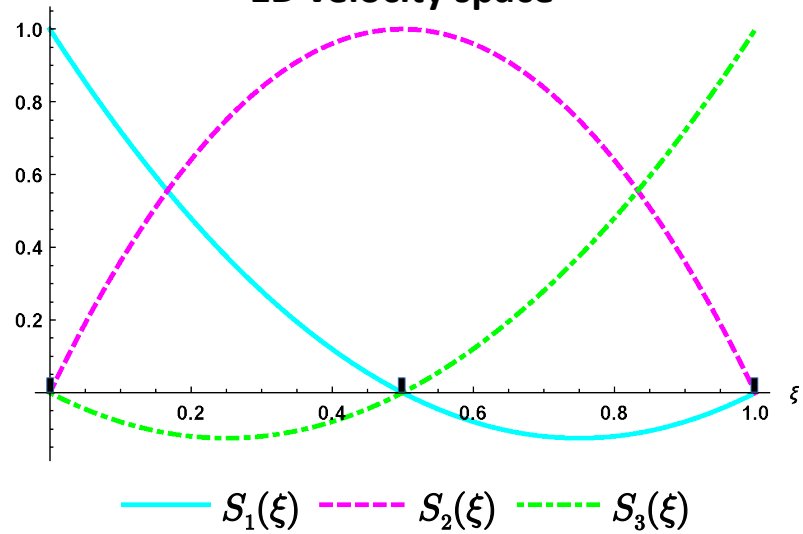
- 1D velocity space: One element consists of 3 grid points.
  Let $\xi(v) = (v - v_{i-1})/(v_{i+1} - v_{i-1}) \in [0, 1]$.

$S_1(\xi)$, $S_2(\xi)$, $S_3(\xi)$ should fulfill:

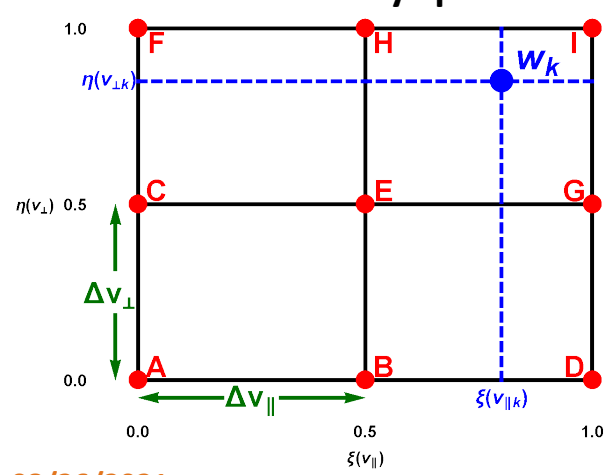
Particle conservation	$S_1(\xi) + S_2(\xi) + S_3(\xi) = 1$
Momentum conservation	$0 \cdot S_1(\xi) + 0.5 \cdot S_2(\xi) + 1.0 \cdot S_3(\xi) = \xi$
Energy conservation	$0^2 \cdot S_1(\xi) + 0.5^2 \cdot S_2(\xi) + 1.0^2 \cdot S_3(\xi) = \xi^2$

- **More difficult to do the inverse grid → particles mapping.**
  We want to preserves moments $\leq$ order of elements.
  Solution: mapping with pseudo-inverse.

1D velocity space



2D velocity space



Mapping with pseudo-inverse

[Hirvijoki et al, arXiv:1802.05263 (2018)]

- Particle distribution of $N_{\text{particles}}$ particles $\quad f_{\text{particle}}(\mathbf{v}) = \Sigma_k \, w_k \, \delta(\mathbf{v} - \mathbf{v}_k)$
  Finite element space $\mathcal{F}$ and a function $u \in \mathcal{F}$
- $u = f_{\text{particle}}(\mathbf{v})$ generally not possible
  For basis functions $\phi_j \in \mathcal{F}$ we write $\quad u = \Sigma_j \, c_j \, \phi_j$
  Solution c_j should fulfill weak equivalence $\quad \int d\mathbf{v} \; \phi_i \; f_{\text{particle}} = \int d\mathbf{v} \; \phi_i \; u$
- Mass matrix $\mathrm{M}_{ij} = \int d\mathbf{v} \; \phi_i \phi_j \; [N_{\text{vertices}} \times N_{\text{vertices}}]$ $\quad$ "Positions" matrix $\mathrm{V}_{jk} = \phi_j(\mathbf{v}_k) \; [N_{\text{vertices}} \times N_{\text{particles}}]$
  Coefficients vector $\mathbf{c}$ $\quad$ Weights vector $\mathbf{w}$

  Weak equivalence $\quad \mathrm{M}\mathbf{c} = \mathrm{V}\mathbf{w} \quad \Rightarrow \quad \mathbf{c} = \mathrm{M}^{-1}\mathrm{V}\mathbf{w}$

- Particles → grid mapping $\quad f_{\text{grid}} = \mathrm{V}\mathbf{w} \quad (= \mathrm{M}\mathbf{c})$
  On grid do operations (e.g. collisions) $\quad \mathbf{c} \rightarrow \mathbf{c}_{\text{new}}$
  V rectangular so inverse not well defined $\Rightarrow$ grid → particles mapping with pseudo-inverse V^{+}
  E.g. if $N_{\text{particles}} > N_{\text{vertices}}$ use right inverse $\quad \mathbf{w}_{\text{new}} = \mathrm{V}^T(\mathrm{V}\mathrm{V}^T)^{-1}(\mathrm{M}\mathbf{c}_{\text{new}})$
- Use XGC resampling to ensure pseudo-inverse calculation (good particle coverage in phase-space).

New mapping implemented in XGC using PETSc

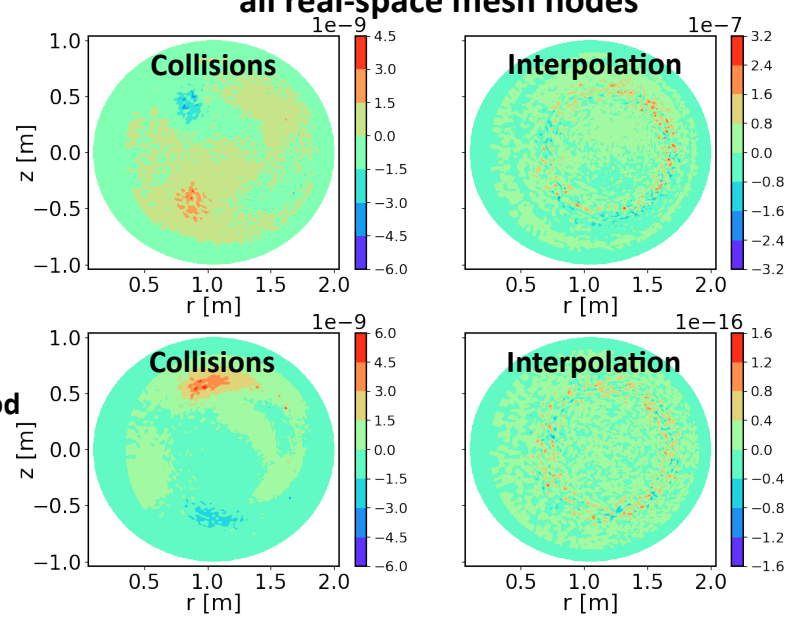
- Fokker-Planck-Landau collision operator in XGC conserves particle density, total parallel momentum and total kinetic energy.
- We want interpolation error << collision error < tolerance.
- Collision and pseudo-inverse interpolation errors can be made arbitrarily small by stricter tolerances in iterative solvers. Bilinear interpolation errors set by the algorithm, e.g. does not conserve energy.

Circular tokamak example

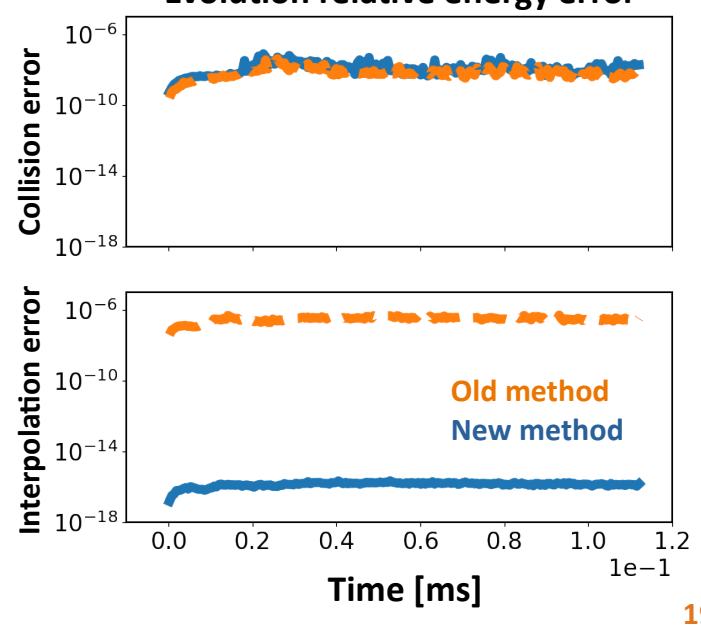
Map of relative energy change in all real-space mesh nodes

Old bilinear method

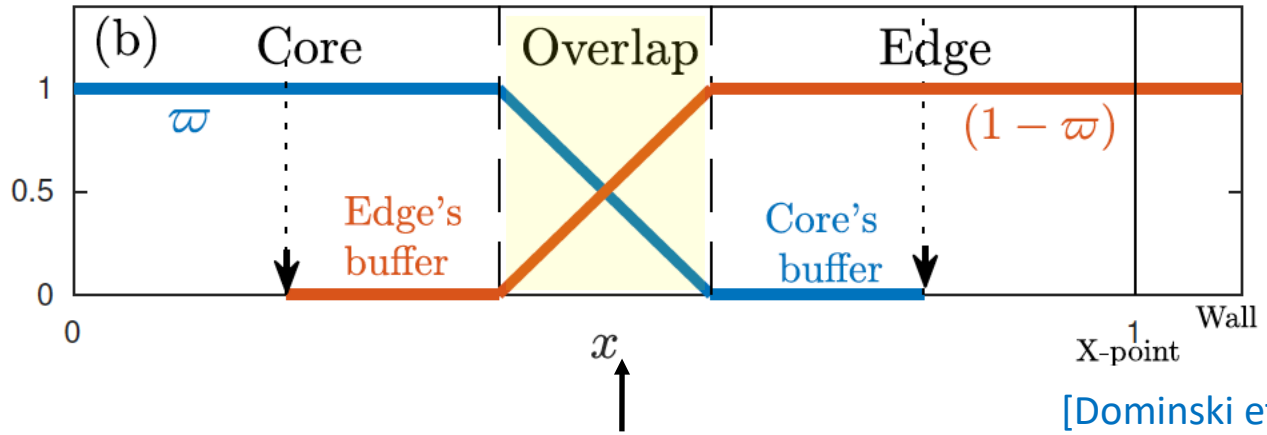
New pseudo-inverse method



Evolution relative energy error



Necessity for more accurate resampling and v-space interpolation in WDMApp f-coupling



[Dominski et al, PoP (2021)]

Need to construct a common $f(x, v)$ in the overlap region

- GENE - XGC: XGC's $f(x, v)$ needs to be smooth. Otherwise, Courant-Friedrichs-Lewy condition can be an issue in GENE. Also, GENE's $f(x, v)$ may need to be converted to XGC's particles.

- GEM - XGC: delta-f GEM uses much less particles/vertex than XGC. Particle number needs to be up/down sampled with accurate velocity-space interpolation for the good common $f(x, v)$.

- A common Fokker-Planck-Landau operation, without an artificial heating, is needed in the coupler for the overlap region. Different $C(f)$ operator in each code can yield different $f(x, v)$.

Outline

- Motivation
- Brief introduction to the XGC code
- Particle resampling in a PIC code
- Updated velocity-space interpolation using a pseudo-inverse
- Summary

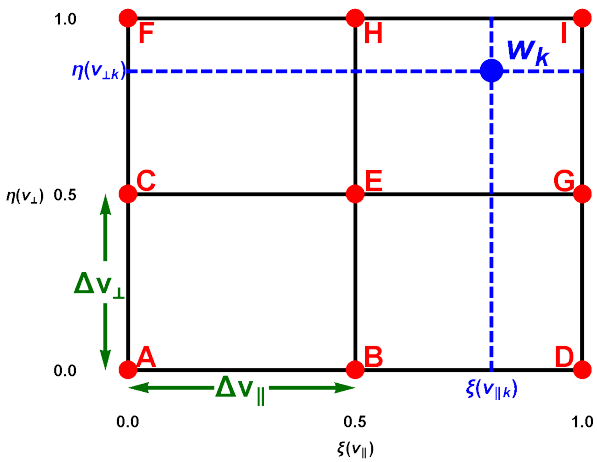
Summary

- Particle resampling and pseudo-inverse velocity mapping are two techniques that will help enable long-time simulations of High-confinement mode plasmas with steep edge pedestals.
- Particle resampling is a powerful tool in XGC with several applications.
- New pseudo-inverse velocity mapping implemented in XGC works as intended: conserves particle number, momentum and energy exactly.
- Both collision error and interpolation error can now be made arbitrarily small down to machine precision by choosing stricter tolerances in the iterative solvers.

Backup slides

Mapping marker particles ↔ grid

- 2D uniform rectangular grid in $v_\parallel$, $v_\perp$.
  $\xi(v_\parallel) = (v_\parallel - v_{\parallel,i-1})/(2\Delta v_\parallel)$; $\eta(v_\perp) = (v_\perp - v_{\perp,j-1})/(2\Delta v_\perp)$



- Map marker particles (with weights w_k) to the 9 vertices using
  $\mathcal{P}_{k\to A} = S_1(\xi(v_{\parallel k}))\, S_1(\eta(v_{\perp k}))$
  $\mathcal{P}_{k\to B} = S_2(\xi(v_{\parallel k}))\, S_1(\eta(v_{\perp k}))$
  ⋮
  $\mathcal{P}_{k\to I} = S_3(\xi(v_{\parallel k}))\, S_3(\eta(v_{\perp k}))$

- Distribution on the mesh $f_{\mathrm{mesh},\beta} = V_\beta^{-1}\, \Sigma_k\, w_k \mathcal{P}_{k\to\beta}$; V_β - velocity space volume of vertex β .

- After source operations, inverse mapping back to particle α
  $\mathcal{P}_{\beta\to\alpha}^{(-1)} = \mathcal{P}_{\alpha\to\beta}/(\Sigma_k\, \mathcal{P}_{k\to\beta})$; New weight $w_\alpha^{\mathrm{new}} = \Sigma_\beta\, f_{\mathrm{mesh},\beta}^{\mathrm{new}}\, V_\beta\, \mathcal{P}_{\beta\to\alpha}^{(-1)}$.

- Shape functions $S_1(\xi)$, $S_3(\xi)$ are negative in parts of the domain ⇒
  Mapping coefficients $\mathcal{P}_{k\to\beta}$ can be negative ⇒
  Problem! Risk of division by 0 in inverse grid → particles mapping.

Proof of conservation 1D: one particle

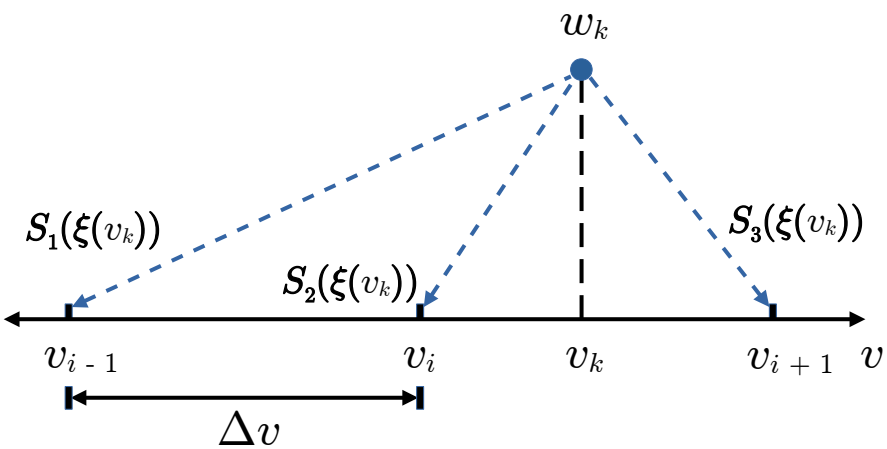
Quadratic Lagrange P_2 shape functions

$$S_1(\xi) = 1 - 3\,\xi + 2\,\xi^2$$

$$S_2(\xi) = 4\,\xi - 4\,\xi^2$$

$$S_3(\xi) = -\,\xi + 2\,\xi^2$$

$$\xi(v) = (v - v_{i-1})/(2\Delta v) \in [0,\,1]$$



Particle conservation

$$n_{\text{particle},\,k} = w_k$$

$$n_{\text{grid},\,i-1} + n_{\text{grid},\,i} + n_{\text{grid},\,i+1} = w_k\,S_1(\xi(v_k)) + w_k\,S_2(\xi(v_k)) + w_k\,S_3(\xi(v_k)) = w_k$$

Momentum conservation

$$P_{\text{particle},\,k} = w_k\,v_k$$

$$P_{\text{grid},\,i-1} + P_{\text{grid},\,i} + P_{\text{grid},\,i+1} = w_k\,S_1(\xi(v_k))\,v_{i-1} + w_k\,S_2(\xi(v_k))\,v_i + w_k\,S_3(\xi(v_k))\,v_{i+1} = w_k\,v_k$$

Energy conservation

$$E_{\text{particle},\,k} = w_k\,v_k^2$$

$$E_{\text{grid},\,i-1} + E_{\text{grid},\,i} + E_{\text{grid},\,i+1} = w_k\,S_1(\xi(v_k))\,v_{i-1}^2 + w_k\,S_2(\xi(v_k))\,v_i^2 + w_k\,S_3(\xi(v_k))\,v_{i+1}^2 = w_k\,v_k^2$$

Mapping with pseudo-inverse

If $N_{\text{particles}} < N_{\text{vertices}}$ (unusual case) use left pseudo-inverse:

- Mass matrix $\mathrm{M}_{ij} = \int d\mathbf{v}\ \phi_i \phi_j$ — invertible square matrix size $N_{\text{vertices}} \times N_{\text{vertices}}$

  Particle positions matrix $\mathrm{V}_{jk} = \phi_j(\mathbf{v}_k) = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}$ — tall skinny matrix size $N_{\text{vertices}} \times N_{\text{particles}}$

  Weak equivalence — $\mathrm{M}\mathbf{c} = \mathrm{V}\mathbf{w}$

- Particles → grid mapping — $f_{\text{grid}} = \mathrm{V}\mathbf{w} \quad (= \mathrm{M}\mathbf{c})$

  On grid do operations (e.g. collisions) — $\mathbf{c} \rightarrow \mathbf{c}_{\text{new}}$

  Left pseudo-inverse ($\mathrm{V}^{+}\mathrm{V} = \mathrm{I}$) — $\mathrm{V}^{+} = (\mathrm{V}^{T}\mathrm{V})^{-1}\mathrm{V}^{T}$

  New weights (preserves moments $\leq$ order) — $\mathbf{w}_{\text{new}} = (\mathrm{V}^{T}\mathrm{V})^{-1}\mathrm{V}^{T}(\mathrm{M}\mathbf{c}_{\text{new}})$

  If identity operation on grid: — $\mathbf{w}_{\text{new}} = (\mathrm{V}^{T}\mathrm{V})^{-1}\mathrm{V}^{T}(\mathrm{V}\mathbf{w}) = \mathbf{w}$

Mapping with pseudo-inverse

If $N_{\text{particles}} > N_{\text{vertices}}$ (typical case) use right (Moore-Penrose) pseudo-inverse:

- Mass matrix $\mathrm{M}_{ij} = \int d\mathbf{v}\ \phi_i \phi_j$ — invertible square matrix size $N_{\text{vertices}} \times N_{\text{vertices}}$

  Particle positions matrix $\mathrm{V}_{jk} = \phi_j(\mathbf{v}_k) = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$ — short fat matrix size $N_{\text{vertices}} \times N_{\text{particles}}$

  Weak equivalence — $\mathrm{M}\mathbf{c} = \mathrm{V}\mathbf{w}$

- Particles → grid mapping — $f_{\text{grid}} = \mathrm{V}\mathbf{w} \qquad (= \mathrm{M}\mathbf{c})$

  On grid do operations (e.g. collisions) — $\mathbf{c} \rightarrow \mathbf{c}_{\text{new}}$

  Right pseudo-inverse ($\mathrm{V}\mathrm{V}^{+} = \mathrm{I}$) — $\mathrm{V}^{+} = \mathrm{V}^{T}(\mathrm{V}\mathrm{V}^{T})^{-1}$

  New weights (preserves moments $\leq$ order) — $\mathbf{w}_{\text{new}} = \mathrm{V}^{T}(\mathrm{V}\mathrm{V}^{T})^{-1}(\mathrm{M}\mathbf{c}_{\text{new}})$

  If identity operation on grid: — $\mathbf{w}_{\text{new}} = \mathrm{V}^{T}(\mathrm{V}\mathrm{V}^{T})^{-1}(\mathrm{V}\mathbf{w}) \neq \mathbf{w}$

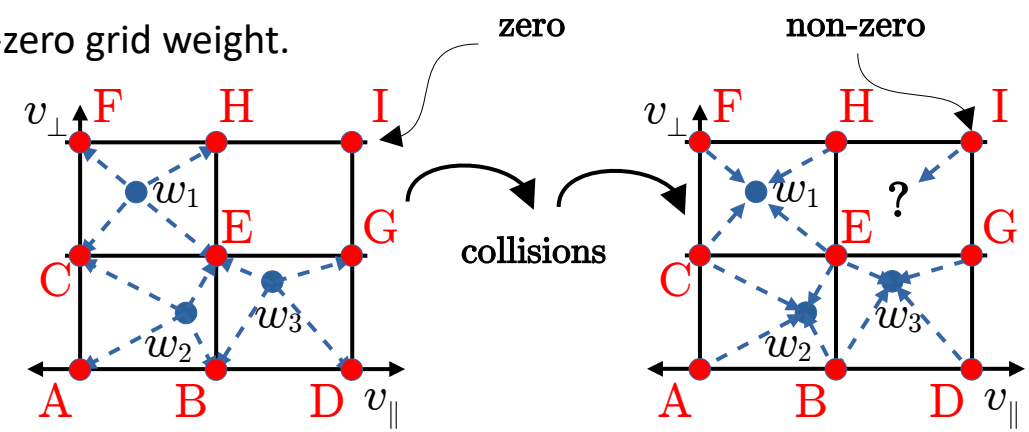
  Idempotence (map twice = map once): — $\mathrm{V}^{T}(\mathrm{V}\mathrm{V}^{T})^{-1}\mathrm{V}\mathrm{V}^{T}(\mathrm{V}\mathrm{V}^{T})^{-1}\mathrm{V}\mathbf{w} = \mathrm{V}^{T}(\mathrm{V}\mathrm{V}^{T})^{-1}\mathrm{V}\mathbf{w}$

We need the resampling in XGC for the velocity mapping

To ensure that pseudo-inverse calculation always work all basis functions (cells) must have particles.

Example

- For simplicity assume 1st order bilinear mapping, 3 particles in 4 cells.
- After particles → grid mapping vertex I has 0 grid weight.
- After collision operation vertex I has non-zero grid weight.
- Vertex I has no particle to write back grid weight to in grid → particles mapping.

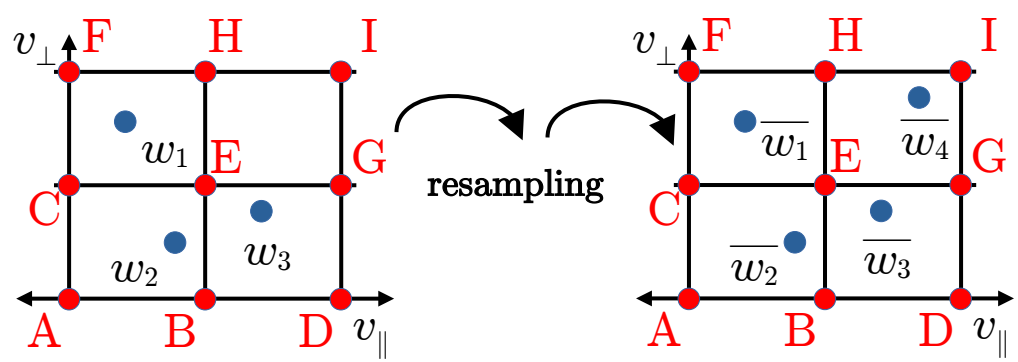


Solution

- Use resampling in XGC to fill all cells before particles → grid mapping.

[Faghihi et al, JCP (2020)]

[Dominski et al, PoP (2020)]



Implementation details

- Particle positions and weights arrays are constructed in the Fortran part of XGC.
- Interface to C++ where PETSc interpolation routines are used.
- Pseudo-inverse calculated with iterative Krylov solver in PETSc.
- At this point the new interpolation is slow. Bottlenecks are to construct the M_{ij} and V_{jk} matrices.
- There is room for substantial speedup:
  — Besides MPI implement OpenMP parallelization (no communication between ranks/threads).
  — The code can be rewritten to be more efficient.
  — Rewrite all XGC source routines into C++ with Kokkos?
  — Use a different collision operator with adaptive mesh refinement (AMR) [M. F. Adams].

AMR example