

NNLSQ

Grid in Cartesian coordinates

Two ways to generate noise/remove bias: coordinate-wise, or for each grid point (more expensive)

Construct velocity moment matrix and solve

Grid in spherical coordinates

Equal volume/equal radius?

TODO

- ☒ NTC collisions
- ☒ I/O of particle properties (basic)
- ☒ Particle generation (equal weights)
- ☒ Properties computation
 - ☒ basic ones
 - ☒ moments
- ☒ I/O of output
- ☒ Maxwellian test case (equal weights)
- ☒ Two species relaxation (equal weights)
- ☒ BKW test case (equal weights)
- ☒ Particle generation (variable weight)
- ☒ Grid merging
- ☒ Octree merging
 - ☒ Final tests for bin splitting
 - ☒ Basic compute (ndens, nparticles) for refinement
 - ☒ Tests
 - ☒ Bin property computation
 - ☒ Tests
 - ☒ New particle computes
 - ☒ Tests

- ☐ Refine based on average particle weight not on bin weight
- ☐ mixing rule VHS creator
- ☐ Load dummy data if not found in interactions.toml

TODO: features

- ☐ Add time to output (since we can change dt on the fly)
- ☐ Compute sigma_g_vhs directly (to avoid additional multiplication)
- ☐ Avoid duplicate computation of particle indices in grid-based merging? Avoid second loop? Use Welford's algorithm + initial estimate of mean as middle of box? Or just shift data?
- ☐ Logging struct? (Union of nothing/actual logging struct, write out stuff like "increased octree merging buffer", "increase particle array size")
- ☐ Subsplit into two groups based on energy and merge. Find x s.t. Vol(particles in sphere but not in cube + particles in cube but not in sphere) is minimal. Plus we shift it by the velocity of the neutrals? Plus we can weight by Maxwellian distribution?
- ☐ NNLS with ResizeableArrays
- ☒ NNLS: a lot of effort spent on first large merge, so maybe run that separately and time the rest?

TODO: tests

- ☐ energy / momentum conservation in scattering
- ☒ correct indexing
- ☒ BKW var weight reference solution
- ☐ 1D - no merging, particles don't switch cells during variable weight collisions!!!

Misc thoughts

Best to do octree merging in reverse order? So that less shifting around of particle indices? Do without shifting of bins at all - it's not needed since we're looping anyway? Overwrite current + fill remaining

Comparisons

- ☐ Merging grid-based: pass slice/without species? is that any different?
- ☐ Make more structs immutable - is that better?

Merging tests:

-
- ☐ BKW - preserve production term?