Monte Carlo Solution to Coulomb Collisions

Alexander Liemen

August 29, 2024

1. Problem Introduction
2. Algorithms
   - TAKIZUKA and ABE's Algorithm
   - NANBU's Algorithm
3. Testcases and *My* Results
   - TRUBNIKOV Test
   - DIRAC Initial Distribution
   - Disorder Induced Heating of a Cold Sphere
4. Problems and Comparison Differences
   - Deviation from [6] for TRUBNIKOV's Test
   - Cold Sphere $\gg$ DEBYE-Length
   - Field Solver Scaling Issue

Outline

Simulate Relaxation Behaviour of a Charged Particle Cloud

Particle characteristics:

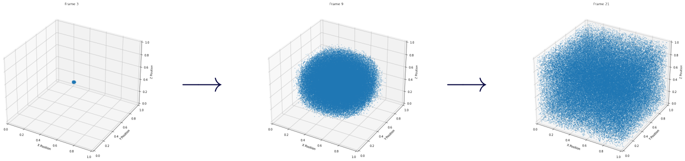
- Equal mass and charge
- No "physical radius"
- Focus on COULOMB interactions

Collision implementation:

- Direct-Simulation-Monte-Carlo (DSMC) approach

Flexibility of the algorithm:

- Mass and charge saved individually per particle
- Suitable for mixed plasmas (e.g. ion-electron interactions)



Landau-Fokker-Planck

Both algorithms converge to the Landau-Fokker-Planck equation.

- Boltzmann equation [4, p. 1] (total force field F^μ):

$$\frac{\partial f_a}{\partial t} + v^\mu \frac{\partial f_a}{\partial x^\mu} + \frac{F^\mu}{m}\frac{\partial f_a}{\partial v^\mu} = \left(\frac{\delta f_a}{\delta t}\right)_{\mathrm{c}}$$

- Lorentz force and binary collisions give Landau form [6]:

$$\left(\frac{\delta f_a}{\delta t}\right)_{\mathrm{c}} = -\sum_b \frac{\partial}{\partial v_j} \frac{e_a^2 e_b^2 \ln\Lambda}{8\pi\epsilon_0^2 m_a} \int \mathrm{d}v' \left[\frac{\delta_{jk}}{u} - \frac{u_j u_k}{u^3}\right]\left[\frac{f_a}{m_b}\frac{\partial f_b(v')}{\partial v'_k} - \frac{f_b}{m_a}\frac{\partial f_a(v')}{\partial v_k}\right]$$

- Coulomb-logarithm $\ln\Lambda$ and Debye-length λ_{D}:

$$\ln\Lambda = \ln\left(\frac{\lambda_{\mathrm{D}}}{b_0}\right), \qquad \lambda_{\mathrm{D}} = \sqrt{\frac{\varepsilon_0 k_{\mathrm{B}} T}{n q^2}}, \qquad b_0 = \frac{|q_1 q_2|}{2\pi\varepsilon_0 k_{\mathrm{B}}^3 m_{12} T}$$

Outline

General Algorithm Simulation Step

Assume a `ParticleContainer` instance f (particle distribution).

```
Select Δt

E ← runFieldSolver(f)
pairs ← selectCollisionPairs(f)
for Every Particle Pair (i, j) do
    Δv ← getCollisionUpdate(i, j)
    v_{i,j} ← v_{i,j} ± Δv/2
end for

v ← v + Δt/2 · e/m · E/ε_0 // Kick 1
x ← x + v · Δt // Drift
v ← v + Δt/2 · e/m · E/ε_0 // Kick 2

particleNodeDistributionUpdate()
```

Select Δt

$E \leftarrow$ runFieldSolver(f)

pairs $\leftarrow$ selectCollisionPairs(f)

for Every Particle Pair (i, j) **do**

  $\Delta v \leftarrow$ getCollisionUpdate(i, j)

  $v_{i,j} \leftarrow v_{i,j} \pm \frac{\Delta v}{2}$

end for

$v \leftarrow v + \frac{\Delta t}{2} \cdot \frac{e}{m} \cdot \frac{E}{\epsilon_0}$ `// Kick 1`

$x \leftarrow x + v \cdot \Delta t$ `// Drift`

$v \leftarrow v + \frac{\Delta t}{2} \cdot \frac{e}{m} \cdot \frac{E}{\epsilon_0}$ `// Kick 2`

particleNodeDistributionUpdate()

Outline

Takizuka and Abe (1977) [6, p. 4310]

Particles i and j, number density n, $\mathbf{u} = \mathbf{v}_i - \mathbf{v}_j$, reduced mass m_{ij}, $u_\perp = \sqrt{u_x^2 + u_y^2}$. Get velocity update $\Delta\mathbf{v}$:

- Variance for $\delta = \tan\left(\frac{\Theta}{2}\right)$ (Θ scattering angle):

$$\langle \delta^2 \rangle = \frac{e_i^2 e_j^2 n \ln\Lambda}{8\pi\epsilon_0^2 m_{ij}^2 u^3} \cdot \Delta t,$$

  sample θ normally distributed around mean 0, calculate Θ.
- Sample the azimuthal scattering angle Φ uniformly in $[0, 2\pi]$.
- Calculate velocity update (conserves *kinetic* energy):

$$\Delta\mathbf{v} = \begin{pmatrix} \frac{u_x u_z}{u_\perp} \sin\Theta\cos\Phi - \frac{u_y u}{u_\perp}\sin\Theta\sin\Phi \\ \frac{u_y u_z}{u_\perp} \sin\Theta\cos\Phi - \frac{u_x u}{u_\perp}\sin\Theta\sin\Phi \\ -u_\perp \sin\Theta\cos\Phi \end{pmatrix} - \mathbf{u}(1 - \cos\Theta).$$

Outline

Nanbu (1997) [3, p. 4644]

- Calculate $s \sim \frac{\Delta t}{t_{\mathrm{relaxation}}}$ and solve (look-up table) for A:

$$s = \frac{\ln \Lambda}{4\pi} \left(\frac{2e^2}{\epsilon_0 m} \right)^2 \frac{n \Delta t}{u^3}, \qquad \coth A - A^{-1} = e^{-s}.$$

- $U_{1,2} \in [0,1]$ uniformly sampled, scattering angle Θ, azimuthal Φ:

$$\cos \Theta = \frac{\ln \left(e^{-A} + 2U_1 \sinh A \right)}{A}, \quad \sin \Theta = \sqrt{1 - \cos^2 \Theta}, \quad \Phi = 2\pi U_2.$$

- Calculate the velocity update (careful, here is a different convention in use: $u_\perp^2 = u_y^2 + u_z^2$):

$$\Delta \mathbf{v} = \mathbf{u}(1 - \cos \Theta) - \frac{\sin \Theta}{u_\perp^2} \begin{pmatrix} -u_\perp^2 \cos \Phi \\ u_y u_x \cos \Phi + u u_z \sin \Phi \\ u_z u_x \cos \Phi - u u_y \sin \Phi \end{pmatrix}.$$

Outline

Outline

Trubnikov Test

Initial conditions:

- Sample positions uniformly, sample velocities according to:

$$f_0(\mathbf{v}) = \left(\frac{m}{2\pi}\right)^{\frac{3}{2}} \frac{1}{\sqrt{T_\parallel} T_\perp} \exp\left(-\frac{m}{2}\left(\frac{v_\parallel^2}{T_\parallel^2} + \frac{v_\perp^2}{T_\perp^2}\right)\right), \quad T = \frac{T_\parallel}{3} + \frac{2T_\perp}{3}.$$

- Energy conservation $\frac{\mathrm{d}T}{\mathrm{d}t} = 0$, analytical solution ($\Delta T = T_\perp - T_\parallel$):

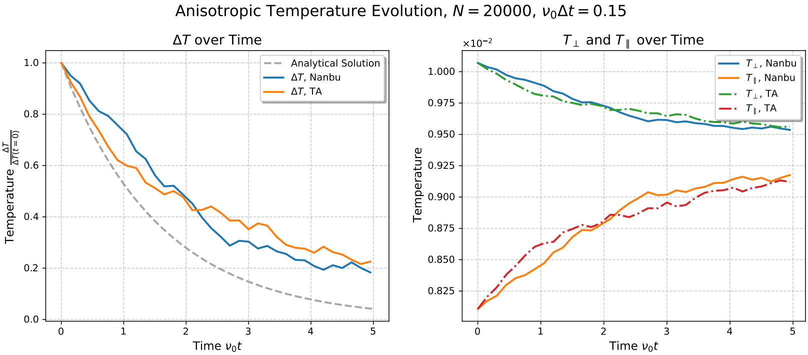
$$\Delta T(t) = \Delta T \cdot e^{-\frac{t}{\tau}}, \quad \tau = \frac{5}{8}\sqrt{2\pi}\,\tau_0, \quad \tau_0 = \frac{\sqrt{m}}{\pi\sqrt{2}e^4}\frac{T^{\frac{3}{2}}}{n \, \ln\Lambda}$$

- Numerical temperature (ignore constants):

$$T_i = \frac{1}{N}\sum_{i=1}^{N} v_i^2, \quad T_\parallel = T_x, \quad T_\perp = \frac{T_y + T_z}{2}.$$

- No self-consistent field (no influence for uniform distribution).

Simple Example with One Realization

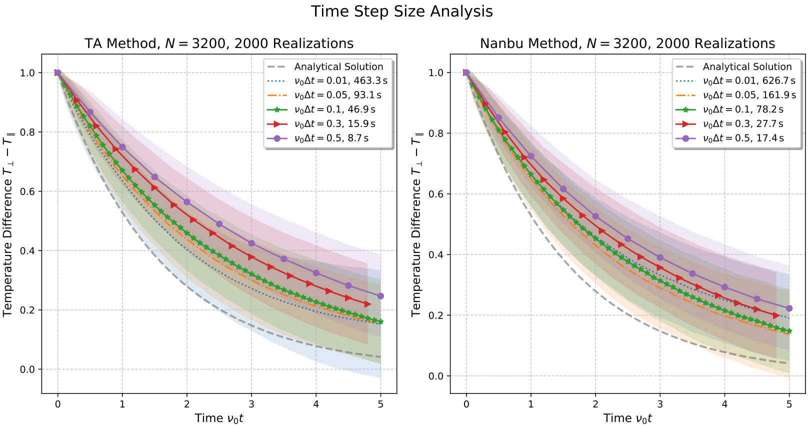


Gives the expected behaviour.

Problem

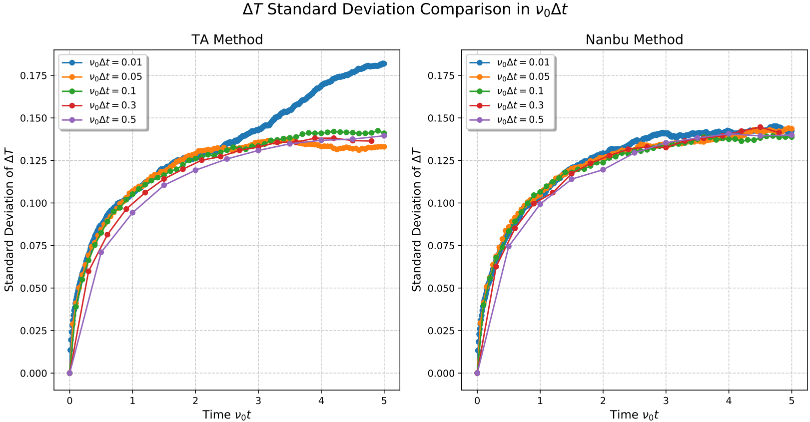
Line is slightly above the analytical solution!

Accuracy Behaviour: Time vs. Step Sizes



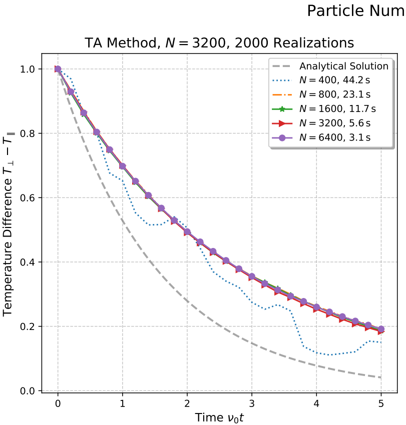
Note: Converges with smaller time step sizes. NANBU is *very slightly* better for a given $\nu_0 \Delta t$, but needs significantly longer.

Precision Behaviour: Time vs. Step Sizes



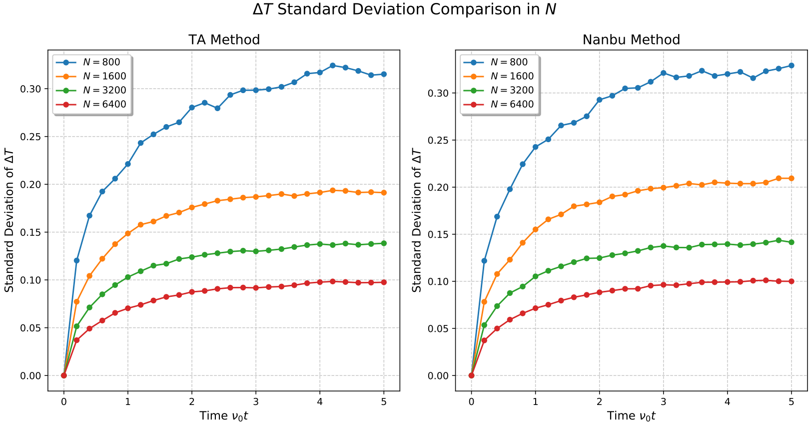
Note: Standard deviation does not grow once the plasma is relaxed.
Time step size has no influence.

Accuracy Behaviour: Time vs. Numbers of Particles



Note: Uses $\nu_0 \Delta t = 0.2$. Does not change accuracy, only affects precision (next two slide).

Precision Behaviour: Time vs. Numbers of Particles



Note: Uses $\nu_0 \Delta t = 0.2$. More particles improve precision. $\textsc{Nanbu}$ is *very slightly* above TA.

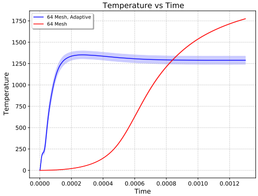
Outline

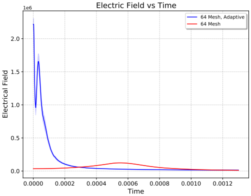
DIRAC Initial Distribution

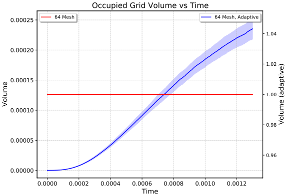
Initial conditions:

- Initialize all velocities to 0.
- Sample positions uniformly in a cube of length d_x centered in the middle of the domain.
- Activate self-consistent electric field.

Example $d_x = 0.005L$, 1000 particles, 500 timesteps and 10 realizations:



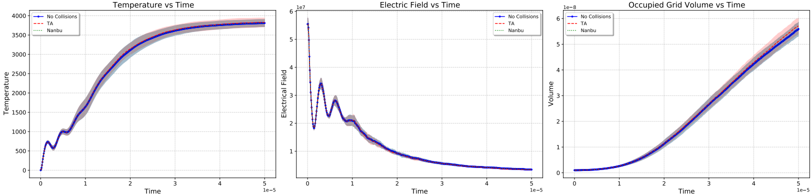




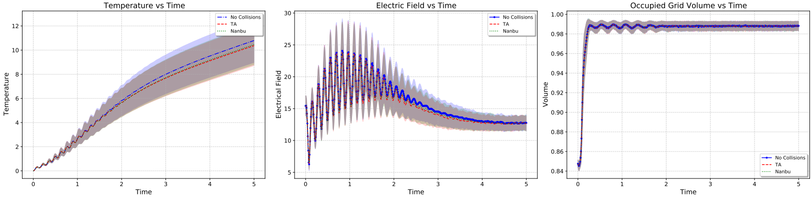
Note: No adaptive grid means all particles are within one cell, delayed and damped initial "surge" of the electrical field.

Influence of Collisions

$d_x = 0.001$, $N = 1000$, 64 adaptive grid, 500 timesteps, 10 realizations:



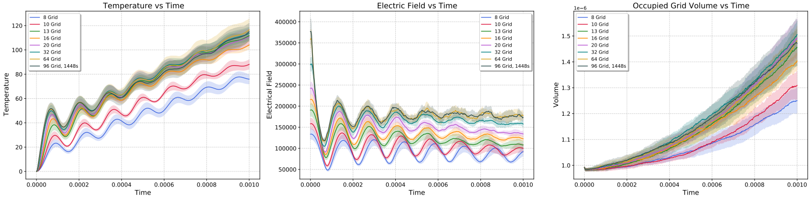
Using $d_x = 0.95$ instead:



Note: Collisions important close to equlibrium/at big timesteps. Drastic change in $|\Delta \mathbf{v}|$ over time: small window where collisions matter.

Influence of Mesh Grid Size

Using $d_x = 0.01$, $N = 500$, 400 timesteps over 10 realizations with the adaptive grid:

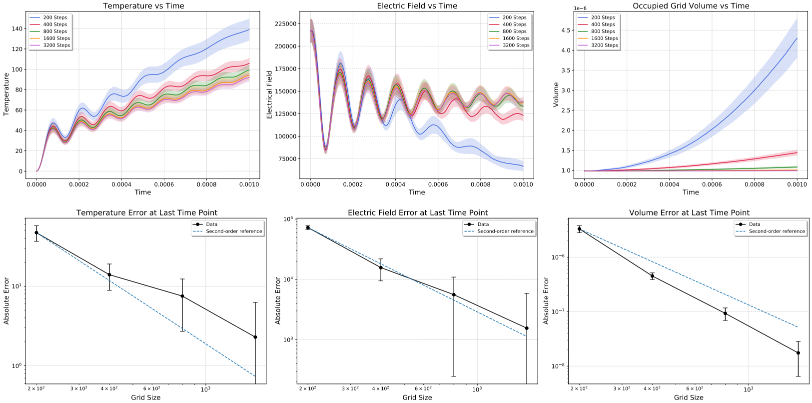


Note:

- Diminishing return within standard deviation after grid size 16 (since the potential is almost uniform).
- Field increases slightly with mesh size (faster space expansion and temperature increase).
- Slight upwards trend in temperature: potential energy is converted to kinetic energy.

Influence of (Time) Step Size

$d_x = 0.01$, $N = 500$, 16 grid size over 100 realizations and relative error:



Note: Roughly second order convergence, which is expected from a Leap-frog step.

Outline

Disorder Induced Heating of a Cold Sphere

Initial conditions according to [2, p. 595]:

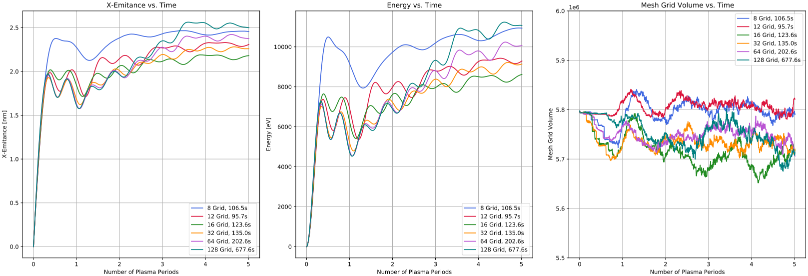
- $L = 100\,\mu\text{m}$, $N = 156055$, $n_0 = 6.67 \cdot 10^{18}\,\text{m}^{-3}$.
- Positions uniformly in sphere of radius $R = 17.74\,\mu\text{m}$.
- Velocities initially at 0.
- Linear focusing force: apply equal and opposite radial electrical force (not accumulative, so fast particles will escape after some steps).
- Additionally: reflect particles on the sphere “shell”.
- Analyze the x-emmitance:

$$\varepsilon_{x,\text{rms}} = \sqrt{\langle x^2 \rangle \langle v^2 \rangle - (xv)^2}.$$

- Expect oscillation period of $\tau = 4.3 \cdot 10^{-11}\,\text{s}$ and a final x-emittance value of $\varepsilon_{x,n}^{\text{eq}} = 0.491\,\text{nm}$.

Mesh-Grid Size Comparison

x emmitance (left) for different (adaptive) grid sizes and 1 realization:



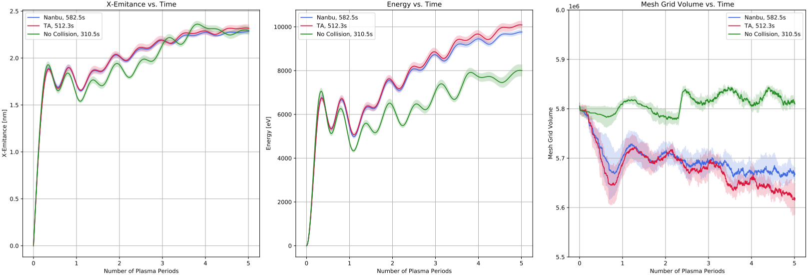
- The confinement works well (right plot).
- Dominant frequencies from FFT on the 64 grid emittance:

$$2.16 \cdot 10^{-11}\,\text{s}, \qquad 6.49 \cdot 10^{-11}\,\text{s}, \qquad 3.24 \cdot 10^{-11}\,\text{s}.$$

- Diminishing returns at small grids due to homogeneity of particles.
- Missing “smoothness” correlates with spikes in mesh grid volume.

Collision Algorithm Comparison

Using different collision algorithms and 5 realizations each:

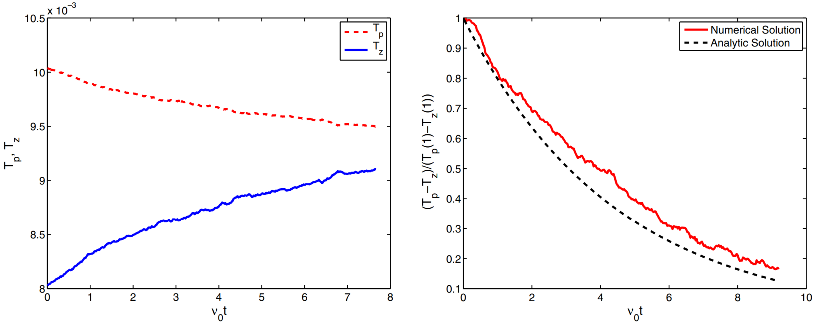


- As in [2, p. 595]: no collisions mean lower emittance.
- Collisions significant shortly in the beginning (as before).
- No significant difference between TA and NANBU.
- Overshoots expected emittance value (as before).

Outline

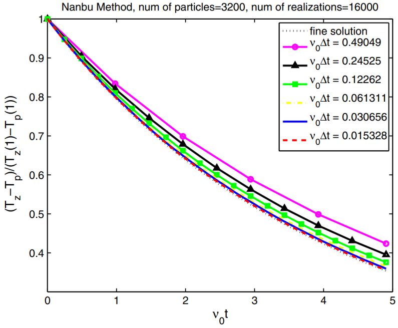
Outline

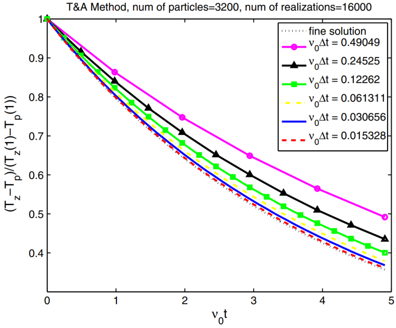
Trubnikov “Reference” Solution: Test Run



Note: similar behaviour, but simulation slightly above analytical solution.

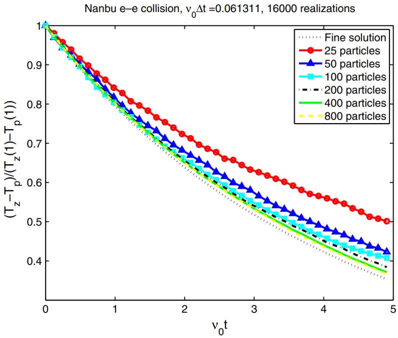
Trubnikov “Reference” Solution: Time Step Size

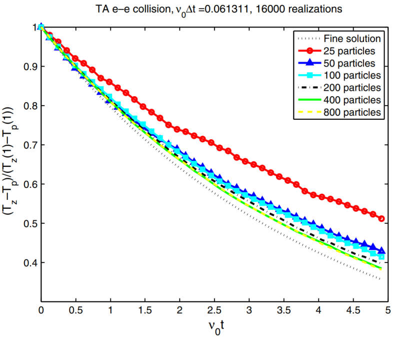




- [6] does not use the analytical solution $\rightarrow$ same convergence discrepancy.
- No explanation: uses a fine solution without providing any details.
- NANBU has slightly better accuracy (does not mention computation time).

Trubnikov “Reference” Solution: Number of Particles





- Paper notes convergence in N.
- Does not mention the analytical solution and uses a fine solution.
- Looking at the first graph, their analytical solution is also significantly lower.
- [6] never mentioned their domain size, n or $\ln\Lambda$, λ_D calculation.

PAUL SCHERRER INSTITUT

Outline

Insignificance of Collisions During Cold Sphere Heating

The program has to options:

- Calculate λ_D, compute collisions in a grid of size $\frac{L}{\lambda_\mathrm{D}}$.
- Compute collisions per usual grid cell.

Problem ($k_\mathrm{B}T_\mathrm{eq} \approx 1.96$meV):

$$\lambda_\mathrm{D} = \sqrt{\frac{\varepsilon_0 k_\mathrm{B} T}{n q^2}} \approx 0.13\,\mu\mathrm{m}.$$

- Gives on average 0.06 particles per cell.
- Confirms marginal significance of collisions in this context.

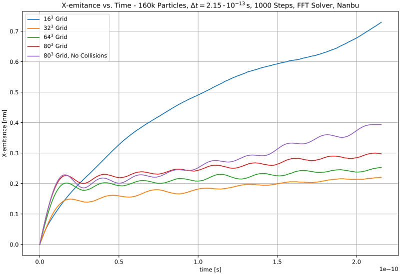
Therefore: here obtained result makes sense!

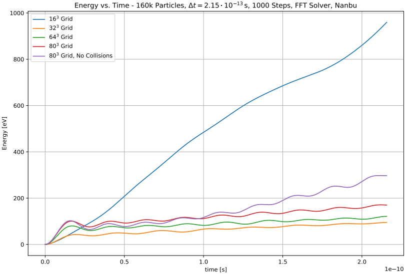
Outline

![PSI logo]

Adaptive Grid Sizing: Scaling Problem

Same simulation, no adaptive grid and different grid sizes:





- Oscillations are way more regular.
- Is way closer to the expected equilibrium ($0.491\,\mathrm{nm}$).
- Oscillation period seems unchanged.

Adaptive Grid Sizing: Implementation

- Inside `advance()`, calculate self-consistent field using:

```
if (this->computeSelfField_m) {
    if (this->adjust_field_dims) adjustFieldMeshDimensions();
    this->par2grid();
    this->fsolver_m->runSolver();
    this->grid2par();
    if (this->adjust_field_dims) resetBoundaries();
}
```

- Adjust grid dimensions, such that every particle is barely in it.
- After solving and interpolating onto the particles, the boundaries are resetted to the value before.

Adaptive Grid Sizing: Adjust Dimensions

```cpp
void adjustFieldMeshDimensions() {
    std::shared_ptr<ParticleContainer_t> pc = this->pcontainer_m;
    auto *mesh = &this->fcontainer_m->getMesh();
    auto *FL   = &this->fcontainer_m->getFL();

    // Calculate the maximum and minimum of all particle coordinates using Kokkos
    view_type* R               = &(pc->R.getView());
    MinMaxReducer<Dim> minMax;
    findMinMax(R, minMax);
    Vector_t<double, Dim> maxR = minMax.max_val, minR = minMax.min_val;

    // Now figure out componentwise global min/max values
    Vector_t<double, Dim> globalMaxR, globalMinR;
    for (size_t i = 0; i < Dim; i++) {
        ippl::Comm->reduce(&maxR[i], &globalMaxR[i], 1, std::greater<double>());
        ippl::Comm->reduce(&minR[i], &globalMinR[i], 1, std::less<double>());
    }

    // Calculate new mesh spacing
    Vector_t<double, Dim> hr = (globalMaxR-globalMinR) / mesh->getGridsize();

    // set the origin and mesh spacing of the mesh via
    mesh->setMeshSpacing(hr);
    mesh->setOrigin(globalMinR);

    this->rmin_m = globalMinR;
    this->origin_m = globalMinR;
    this->rmax_m = globalMaxR;
    this->hr_m = hr;

    extLayoutUpdate(FL, mesh);
    pc->update();
}
```

Adaptive Grid Sizing: Extended Layout Update

```
void extLayoutUpdate(ippl::FieldLayout<Dim>* fl, ippl::UniformCartesian<T, Dim>* mesh) {
    std::shared_ptr<ParticleContainer_t> pc = this->pcontainer_m;
    std::shared_ptr<FieldContainer_t> fc    = this->fcontainer_m;

    Field_t<Dim>* rho_m   = &(fc->getRho());
    VField_t<T, Dim>* E_m = &(fc->getE());

    rho_m->updateLayout(*fl);
    E_m->updateLayout(*fl);
    pc->getLayout().updateLayout(*fl, *mesh);

    std::get<FFTSolver_t<T, Dim>>(this->fsolver_m->getSolver()).setRhs(*rho_m);
}
```

- Does a few (technically redundant) updates.
- Additional `updateLayout` do not make a difference.

Adaptive Grid Sizing: Reset Boundaries

Calculating the electrical field:

```
void resetBoundaries() {
    this->origin_m = 0.0;
    this->rmin_m   = this->origin_m;
    if (this->initial_distr == "sphere") {
        this->rmax_m = 506.84;
    } else {
        this->rmax_m = 1.0;
    }
    this->hr_m     = this->rmax_m / this->nr_m;

    std::shared_ptr<ParticleContainer_t> pc = this->pcontainer_m;
    auto *FL = &this->fcontainer_m->getFL();
    auto *mesh = &this->fcontainer_m->getMesh();

    // set the origin and mesh spacing of the mesh via
    mesh->setMeshSpacing(this->hr_m);
    mesh->setOrigin(this->rmin_m);
}
```

- Manually reset `rmin`, `rmax` and recalculate `hr`.
- Update the mesh container.

References

[1] Giacomo Dimarco, Russell Caflisch, and Lorenzo Pareschi. Direct simulation Monte Carlo schemes for Coulomb interactions in plasmas. 2010. arXiv: 1010.0108. URL: https://arxiv.org/abs/1010.0108.

[2] C.E. Mitchell and J. Qiang. "A Parallel Particle-Particle, Particle-Mesh Solver for Studying Coulomb Collisions in the Code IMPACT-T". In: Proc. 6th International Particle Accelerator Conference (IPAC'15) (Richmond, VA, USA). International Particle Accelerator Conference 6. Geneva, Switzerland: JACoW, June 2015, pp. 593–595. ISBN: 978-3-95450-168-7. DOI: https://doi.org/10.18429/JACoW-IPAC2015-MOPMA024. URL: http://jacow.org/ipac2015/papers/mopma024.pdf.

[3] K. Nanbu. "Theory of cumulative small-angle collisions in plasmas". In: Phys. Rev. E 55 (4 Apr. 1997), pp. 4642–4652. DOI: 10.1103/PhysRevE.55.4642. URL: https://link.aps.org/doi/10.1103/PhysRevE.55.4642.

[4] Marshall N. Rosenbluth, William M. MacDonald, and David L. Judd. "Fokker-Planck Equation for an Inverse-Square Force". In: Phys. Rev. 107 (1 July 1957), pp. 1–6. DOI: 10.1103/PhysRev.107.1. URL: https://link.aps.org/doi/10.1103/PhysRev.107.1.

[5] Tomonor Takizuka and Hirotada Abe. "A binary collision model for plasma simulation with a particle code". In: Journal of Computational Physics 25.3 (1977), pp. 205–219. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(77)90099-7. URL: https://www.sciencedirect.com/science/article/pii/0021999177900997.

[6] Chiaming Wang et al. "Particle simulation of Coulomb collisions: Comparing the methods of Takizuka and Abe and Nanbu". In: J. Comp. Phys. 227 (Apr. 2008), pp. 4308–. DOI: 10.1016/j.jcp.2007.12.027.