AMSC 664 Final Report: Upgrade to the GSP Gyrokinetic Code

George Wilkie (*gwilkie@umd.edu*)
Supervisor: William Dorland (*bdorland@umd.edu*)

July 1, 2012

Abstract

Simulations of turbulent plasma in a strong magnetic field can, in some cases, take advantage of the gyrokinetic approximation, the result of which is a closed set of equations that can be solved numerically. An existing code, GSP, uses a novel Particle-in-Cell method. In this project, we change the velocity-space representation in GSP. This transformation will eventually make the algorithm more suitable for simulations of turbulence in Tokamak plasmas, while retaining the efficiency and accuracy of the original code.

1 Background

Plasma physics concerns the collective dynamics of collections of many charged particles interacting self-consistently with their respective electromagnetic fields. In a uniform magnetic field, the motion of a charged particle (we are usually referring to ions unless otherwise specified) in the plane perpendicular to the magnetic field is a circle whose radius (the gyroradius ρ) depends on its velocity in that plane:

$$\rho = \frac{v_\perp}{\Omega}$$

(Ω is the *gyrofrequency*). If this radius is much smaller than the characteristic size of the plasma, we can use that ratio as an expansion parameter. Gyrokinetics specifically is the regime in which turbulent fluctuations can be about the size of this gyroradius. This is in contrast to drift-kinetic theory, where the gyroradius needs to be much smaller than the size of the fluctuations. In order to get a closed theory in gyrokinetics, we demand that the drift velocity be small compared to the thermal velocity $v_t \equiv \sqrt{\frac{T}{m}}$. Such drifts come about when there is curvature or a gradient in the magnetic field or if there exists another force field (such as the electric field).

We can summarize the gyrokinetic assumptions thusly:

$$\frac{\rho}{L} \sim \frac{v_E}{v_t} \sim \frac{\omega}{\Omega} \sim \frac{\nu}{\Omega} \sim \epsilon \ll 1$$

Here, $v_E = \frac{E}{B_0}$ is the $E \times B$ drift speed, ω is the frequency of turbulent fluctuations, and ν is the characteristic collision frequency. These assumptions are well justified in plasmas found in magnetic fusion confinement devices and in astrophysical plasmas.

The gyrokinetic equation results from expanding the distribution function in the small parameter $\epsilon \equiv \frac{\rho}{L}$:

$$f(\mathbf{r}, \mathbf{v}, t) = F_0 + \epsilon \delta f + \ldots$$

Where δf represents the fluctuations in the distribution function, which is in turn responsible for fluctuations in density, flux, and temperature, and all perturbed fields that depend thereon.

The function we will be solving for is the gyroaverage of the perturbed distribution $g \equiv \langle \delta f \rangle_{\mathbf{R}}$ (we use these two symbols interchangeably depending on the circumstance). When changing to coordinates $E \equiv \frac{1}{2}v^2$ and $\mu \equiv \frac{v_\perp^2}{2B_0}$, we get our form of the *gyrokinetic equation*

$$\frac{\partial g}{\partial t} + v_\parallel \cdot \frac{\partial g}{\partial z} + (\mathbf{v_E} + \mathbf{v_c} + \mathbf{v_{\nabla B}}) \cdot \nabla_{\mathbf{R}} g = C\left[g\right] - \mathbf{v_E} \cdot \nabla F_0 + (\mathbf{v_c} + \mathbf{v_{\nabla B}}) \cdot \langle \mathbf{E} \rangle_{\mathbf{R}} F_0 + \langle E_z \rangle_{\mathbf{R}} v_\parallel F_0$$

Where:

- $\mathbf{v_E} = \mathbf{E} \times \hat{\mathbf{z}}$ is the "E cross B" drift velocity due to the perturbed electric field.
- $\mathbf{v_c} = \frac{v_\parallel^2}{R_c} \hat{\mathbf{r}} \times \hat{\mathbf{z}}$ is the curvature drift of a particle in a curved magnetic field of radius R_c.
- $\mathbf{v_{\nabla B}} = -\frac{1}{2} v_\perp^2 \frac{\nabla \mathbf{B}}{B_0} \times \hat{\mathbf{z}}$ is the "grad-B" drift velocity in a nonuniform magnetic field.
- $\mathbf{E} \equiv -\nabla \phi$ is the perturbed electric field.
- $F_0 = \frac{n_0}{(2\pi T)^{3/2}} e^{-\frac{1}{2}\frac{v^2}{T}}$ is the equilibrium distribution function
- $C[g]$ is the collision operator, descirbed further below.

We have equated the $\hat{\mathbf{z}}$-direction to be along the background magnetic field so that $\mathbf{B} = B_0 \hat{\mathbf{z}}$.

2 Approach

We solve the gyrokinetic equation above by the method of characteristics, following randomly-placed Lagrangian markers in phase space and integrating when appropriate. The form of the equation shown is intended to emphasize that:

- Gyrocenter markers move in well-defined trajectories according to the standard drifts of charged particle motion.

- With the exception of specifying the geometry (which would include knowledge of ∇F_0, ∇B, and R_c) and the collision operator, the gyroaveraged electric field (and thus the electrostatic potential ϕ) determines several terms in the gryokinetic equation and calculating it is an important part of the algorithm.

To find the fields, we first find the potential:

$$\phi = \frac{\sum\limits_s \int d^3\mathbf{v} J_0\left(\frac{k_\perp v_\perp}{\Omega}\right)\langle \delta f\rangle_{\mathbf{R},\mathbf{s}}}{\sum\limits_s \frac{q_s^2 n_{0,s}}{T_s}\left[1 - e^{-k_\perp^2\rho_s^2} I_0\left(k_\perp^2\rho_s^2\right)\right]}$$

Once we have this, we can calculate the fields with $\mathbf{E} = -i\mathbf{k}\phi$ and gyroaverage by multiplying by J_0.

Change of Coordinates

Define, as constants of motion, the appropriately normalized *energy* $E \equiv \frac{1}{2}v^2$ and *magnetic moment* $\mu = \frac{v_\perp^2}{2B_0}$. The Jacobian becomes:

$$\mathcal{J} d^3\mathbf{v} = dv_x dv_y dv_z = d\theta v_\perp dv_\perp dv_\parallel = d\theta \frac{B}{\sqrt{2}} \frac{dE d\mu}{\sqrt{E - \mu B}}$$

It is important to note that, when doing analytic integrations in these coordinates, one must sum over positive and negative $v_\parallel$ since the square root leaves this ambiguous. When we do this integral numerically, there are already both signs of $v_\parallel$ present.

Important quantities expressed in these coordinates include:

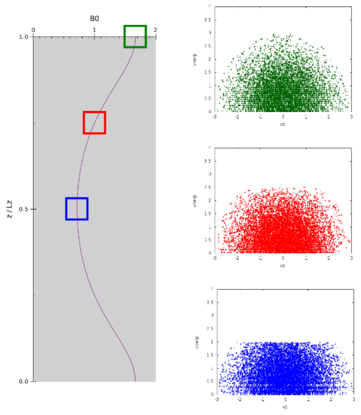
- $v_\parallel = \pm\sqrt{2}\sqrt{E - \mu B}$
- $v_\perp = \sqrt{2\mu B}$
- $F_0 = \frac{n_0}{(2\pi m T)^{3/2}} e^{-E}$

Monte Carlo Integration

Given our normalized $w \equiv \langle \delta f\rangle / F_0$, we solve the above integral discretely in μ and Monte Carlo in E:

$$\phi \propto \int\limits_0^{\frac{E_{max}}{B}} d\mu J_0\left(a\sqrt{\mu B}\right) \int\limits_\mu^{E_{max}} \frac{dE e^{-E}}{\sqrt{E - \mu B}} w(E, \mu)$$

$$\approx \sum_j^{N_\mu} (\Delta\mu)_j J_0(a\sqrt{\mu_j B}) I_j$$



Where

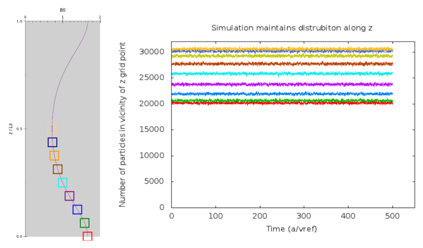
$$I_j = \int \frac{dE\, e^{-E}}{\sqrt{E-\mu B}} w(E,\mu_j) = \langle w \rangle_{p(E)} \approx \frac{1}{N_p}\sum_i^{N_p} w(E_i,\mu_j) = \sum_j^{N_\mu} (\Delta\mu)_j J_0(a\sqrt{\mu_j B}) I_j$$

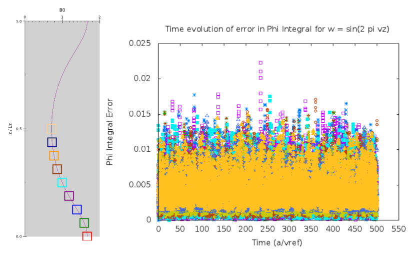
We have defined the probabilistic density of markers:

$$p(E|\mu_j) \equiv \frac{Ae^{-E}}{\sqrt{E-\mu_j B}}$$

Below are illustrations of how this distribtion of markers appears at different magnetic field strengths.

It is important that this distribution remain constant in time near each given z-grid location. Below it is shown that the number of particles at a given z location, once normalized properly, remain with that density. To show that the distribution (and not just the overall density) remains at least approximately constant, we also show that the error in the integral, while stochastic, does not diverge:





The Collision Operator

To calculate the effect of the collision operator, we switch over to using the distribution function $h = \langle\phi\rangle F_0 + g$ to comply with the literature.

We apply Godunov splitting to the gyrokinetic equation so that all the terms besides $C[h]$ are evaluated explicitly, after which the effect of the collision operator (a mixture of diffusion and integral terms) is calculated implicitly. Let $A[h]$ represent the explicit part of the gyrokinetic equation, such that:

$$\frac{dh}{dt} = A[h] + C[h]$$

Applying a finite difference scheme in time, we arrive at the pre-collision distribution:

$$h^* = h^n + \Delta t A[h^n]$$

And use this to obtain the distributon after a full timestep:

$$h^{n+1} = h^* + \Delta t C[h^{n+1}]$$

Since $C[h]$ contains integral terms, it ought to be considered as a non-sparse matrix. However, as in the code GS2, we apply the Sherman-Morrison method that allows us to use one or two tridiagonal matrices (the ones used for the diffusive part of the operator) to properly invert the integral terms. See Appendix B of [Barnes, 2009] for details.

The collision operator is written as the sum of several terms:

$$C[h] = L[h] + D[h] + C_L U_L[h] + C_D U_D[h] + C_E E[h]$$

The table defines and interprets each of these terms:

L	$\frac{\nu_D}{2}\frac{\partial^2 h}{\partial \xi^2} - \frac{1}{4}\alpha^2 \nu_D \left(1+\cos^2\xi\right) h$	Diffusion in pitch-angle (at constant energy)
U_L	$\nu_D F_0 J_0(\alpha) v_\parallel \int d^3v' \nu_D(v') v'_\parallel J_0(\alpha') h(\mathbf{v}')$ $+\nu_D F_0 J_1(\alpha) v_\perp \int d^3v' \nu_D(v') v'_\perp J_1(\alpha') h(\mathbf{v}')$	Conserves momentum due to L operator.
D	$\frac{1}{2v^2}\frac{\partial}{\partial v}\left(\nu_\parallel v^4 F_0 \frac{\partial}{\partial v}\frac{h}{F_0}\right) - \frac{1}{4}\alpha^2 \nu_\parallel \sin^2\xi h$	Diffusion in energy (at constant pitch angle)
U_D	$-\Delta\nu F_0 J_0(\alpha) v_\parallel \int d^3v' \Delta\nu(v') v'_\parallel J_0(\alpha') h(\mathbf{v}')$ $-\Delta\nu F_0 J_1(\alpha) v_\parallel \int d^3v' \Delta\nu(v') v'_\perp J_1(\alpha') h(\mathbf{v}')$	Conserves momentum due to D operator.
E	$\nu_E v^2 J_0(\alpha) F_0 \int d^3v' \nu_E(v') v'^2 J_0(\alpha') h(\mathbf{v}')$	Conserves energy

The different νs here are various functions of speed v, to be interpreted as collision frequencies, and $\alpha = \frac{k_\perp v_\perp}{\Omega}$. The constants C_L, C_D, and C_E can be computed ahead of time, and are just integrals over velocity space independent of h.

This collision operator is artificial: it is not derived from first principles but it does satisfy important physical conservation laws as well as the Boltzmann H-theorem. All of these are shown by integrating the collision operator over velocity space. The zeroth, first, and second moments of the collison operator vanish by design (note that the integrand and the outside factor of each of these terms have a similar form; integration by parts along with special definitions of ν_D, $\nu_\parallel$, ν_E and $\Delta\nu$ are required).

All terms, including inter-species collision terms between electrons and ions, are included in the upgraded GSP. This part of the code still requires validation, especially the conservation options.

3 Implementation

Our approach is the Particle-in-Cell technique: a hybrid Eulerlian/Lagrangian scheme. A pure Lagrangian approach would require the fields be superimposed from every particle onto every particle (requiring $\mathcal{O}(N_{part}^2)$ operations). Instead, particles are locally deposited onto a grid; the relevant quantities are calculated on that grid; then reinterpolated back to the particles who then time-advance. This is computationally advantageous if $N_{part} \gg N_{grid}$, which makes the grid operations cheap and requiring only $\mathcal{O}(N_{part}) + \mathcal{O}(N_{grid} \log N_{grid})$ operations (the $N_{grid} \log N_{grid}$ comes from the Fourier transform required to accurately compute the gyroaverage.

The particles are each assigned a random position, a random E drawn from a continuous distribution (see Appendix E), one of several discrete values of μ, and a "weight" w. This weight is simply the normalized version of the function we are solving for: $w \equiv \frac{\langle \delta f \rangle}{F_0}$. It is initialized randomly, but is advanced in time according to the ODE along characteristic curves (the right-hand-side of the gyrokinetic equation).

The time-stepping algorithm for the particle weights is a second-order Runge Kutta "predictor/corrector" method:

1. **Initialize particles in phase space**
2. **Predictor step**
   - Calculate fields at step n
   - Calculate marker weights along characteristics for step $n +^1 /_2$
   - Advance marker positions for half timestep
   - Update weights with collision operator for step $n +^1 /_2$
3. **Corrector step**

    - Calculate fields at step $n + ^1/_2$
    - Calculate marker weights along characteristics for step $n + 1$
    - Advance marker positions for the full timestep
    - Update weights with collision operator for step $n + 1$

4. **Output results as necessary**

- Repeat steps 2 to 4 as necessary.

Further details on the algorithm are in Appendices C and D.

4 Databases

- GSP Source code (includes relevant revision history under branches/gwilkie and branches/df):
  http://gyrokinetics.svn.sourceforge.net/viewvc/gyrokinetics/gsp/bran ches/df
  - Important sections of code (lines):
    - Main body, particle orbits: (429-609)
    - Calculating Phi: (931-1119)
    - Calculating Fields: (1237-1364)
    - Updating the weights: (1366-1424)
    - Collision operator: (1457-2883)
- Original GSP source code
  http://gyrokinetics.svn.sourceforge.net/viewvc/gyrokinetics/gsp/trunk/
- Article with analytic result:
  - Ricci, et al, "Gyrokinetic linear theory of the entropy mode in a Z pinch." *Physics of Plasmas*, **13**: 062102
- Data files included in attached tarball.

5 Validation

We need to ensure that the changes we have made to GSP do not affect its accuracy, especially in regimes in which it was originally accurate. We performed this in two stages: a local validation of the ϕ integral, and a global validation of the code overall, including particle orbits and time-dependence.

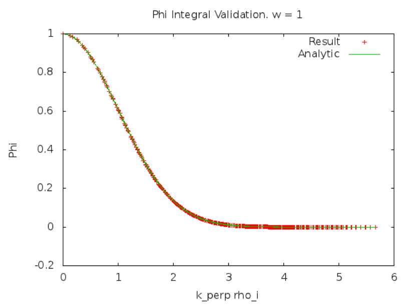
Validation of Phi

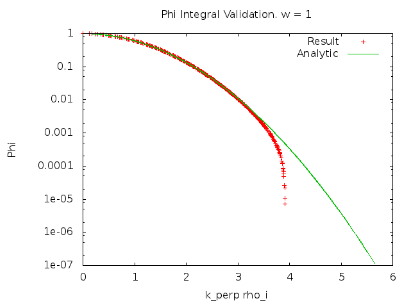
Since it is such a key part of the algorithm, we validate the ϕ integral separately. We are validating both the Monte Carlo integration over E and the discrete midpoint rule of μ. To so, we give the function $w(E,\mu)$ an artificial value such that we know the value of the integral analytically. We have used the following three functions:

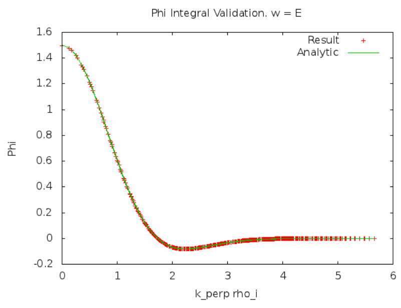
1. $w = 1 \rightarrow \phi \propto e^{-k_\perp^2/2}$
2. $w = E \rightarrow \phi \propto \frac{1}{2}\left(3 - k_\perp^2\right) e^{-k_\perp^2/2}$
3. $w = sin(2\pi\sqrt{E - \mu B}) \rightarrow \phi \propto D_+(\pi) e^{-k_\perp^2/2}$

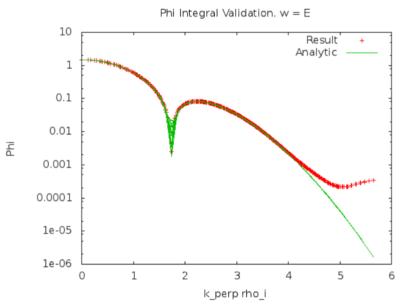
Where D_+ is the Dawson integral. The latter function is most indicative of the kinds of perturbations that are seen, and have the added advantage of having some structure in velocity space and depends on both E and μ.

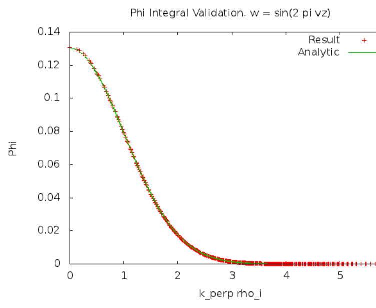
Each of these integrals is performed for many different combinations of k_x and k_y (where the result depends only on $k_\perp = \sqrt{k_x^2 + k_y^2}$), and the numerical result is compared with the known analytic function, given in green. We also plot the logarithmic scale of these graphs to the right to more clearly show the differences as a function of $k_\perp$.



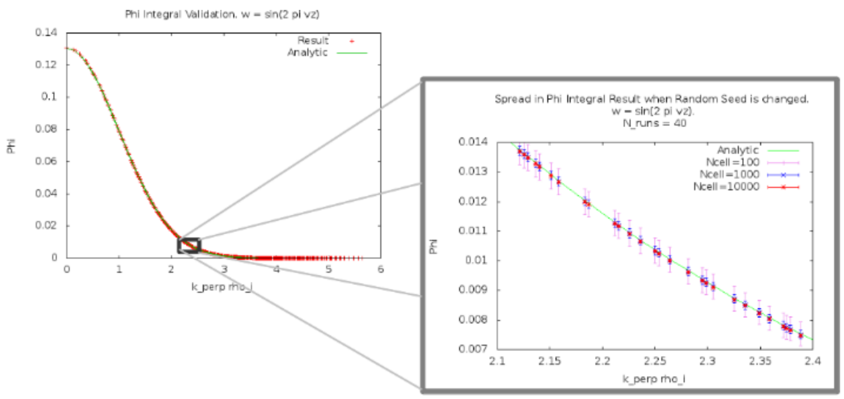




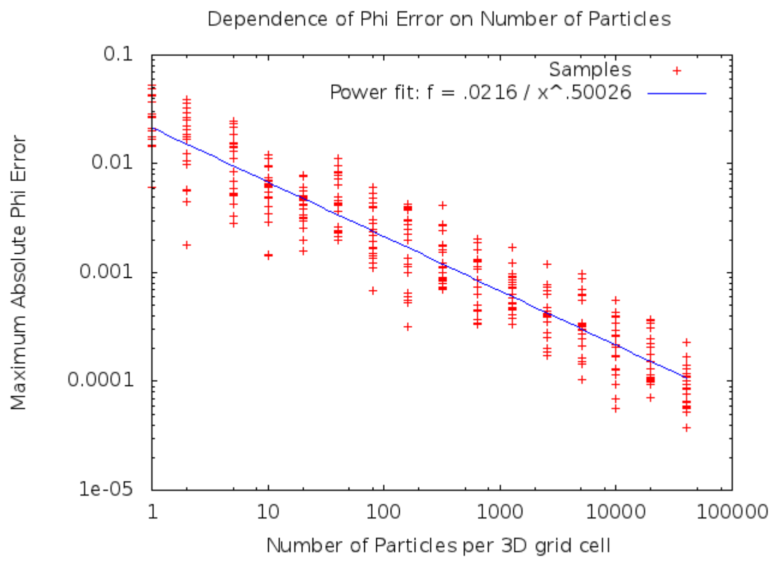




Upon more careful examination, it is realized that the error is stochastic and simply changing the random number seed will result in a different error. Below is an example of several different resolutions, each run 40 times each with different seeds.

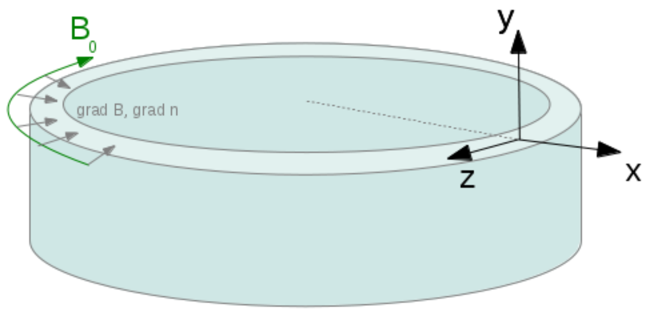


And finally, to make it clear that the error that we see is from the Monte Carlo integration (and is thus remedied by adding more particles), we calculate the integral 40 times for many different numbers of particles per cell (there are 1024 cells to generate the appropriate resolution of $k_\perp$ shown agove).



Physical Validation: Z-pinch Entropy Mode

Here, we combine the calculation of the fields along with particle trajectories subject to curvature and gradient drifts. The geometry of the problem is show below:



The gradient of the magnetic field and the plasma density is toward the center ($-\hat{\mathbf{x}}$). Such a configuration gives rise to an instability along the $\hat{\mathbf{y}}$-direction whose dispersion relation is known (see [Ricci, 2006]). We compare our code to another, more established code GS2. This code is purely Eulerian and solves the gyrokinetic equations fully implicitly.

To study the growth rates, we look at the growth of the following function:

$$\Phi(k_y) = \sum_{kx,kz} |\phi(k_x, k_y, k_z)|^2$$

This is a measure of how much electrostatic energy is in a particular k_y mode. Its growth rate is found by solving the finite difference formula for exponential growth:

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} = \gamma \left(\frac{\Phi^{n+1} + \Phi^n}{2} \right)$$

Where the superscript denotes the time step, and γ is the growth rate. This quantity is updated as an exponentially-weighted moving average so that the output is easily read. Alternatively, one could simply estimate the slope of a logarithmic chart of such growth.

According to the dispersion relation, different k_y modes are going to have different growth rates. Here, we compare to this growth rate spectrum to the GS2 prediction for a variety of test conditions. Let L_n be the density gradient length scale (such that $\frac{\nabla n}{n} = \frac{1}{L_n}$); similarly for the temprature scale L_T; R is the radius of curvature; and T_i and T_e are the ion and electron temperatures, respectively.

1. **Standard Case**: $\frac{L_n}{R} = 0.5$, $R = 1$, $T_e = T_i = 1$, $\frac{1}{L_T} = 0$

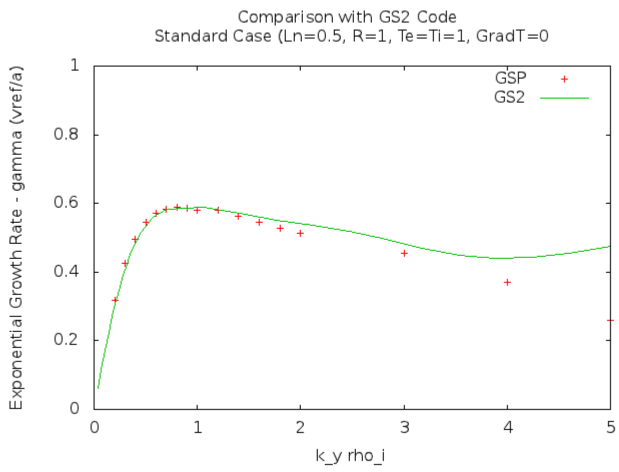
2. **High n Gradient Case**: Same as standard with $\frac{L_n}{R} = 2.0$

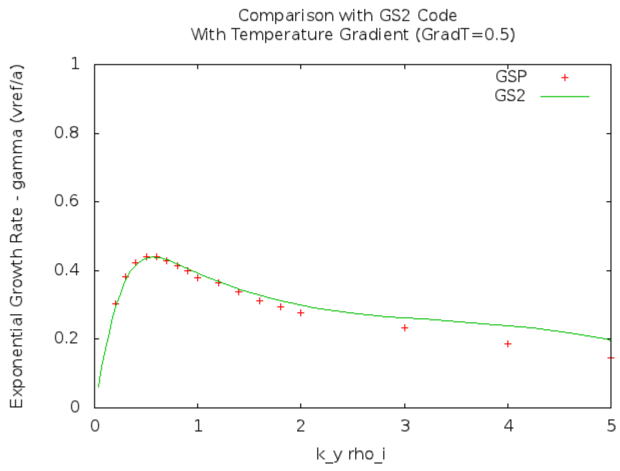
3. **Temperature Gradient Case** : Same as standard with $\frac{L_T}{R} = 0.5$ in addition to a density gradient

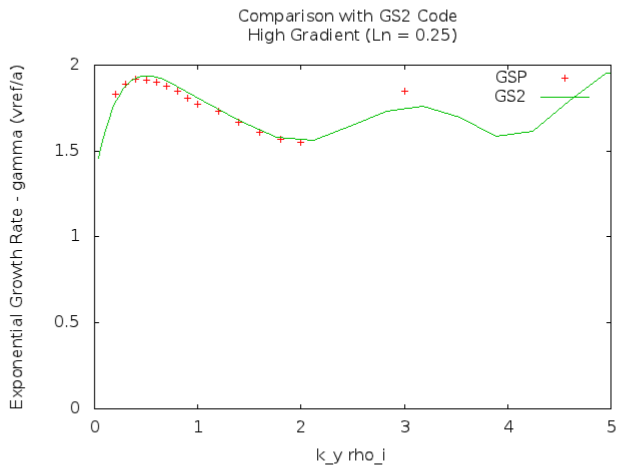
4. **Cold Electron Case**: Same as standard with $T_e = 0.5$

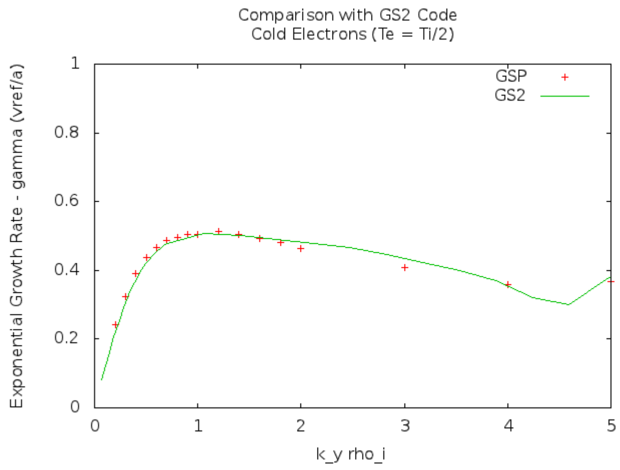
5. **Large Radius Case**:Same as standard with $\frac{L_n}{R} = 0.5$, but $R = 4$

The growth rate spectra are plotted below:









The general conclusion is that the predictions begin to fail at higher k_y. This corresponds to "finite Larmor radius" effects: where the scale of perturbations are smaller than a typical gyroradius. As smaller scales are desired, more particles would be required to achieve accuracy. Otherwise, the agreement is remarkable.

6 Testing

Here we ensure that the parallelization of GSP is maintained. The parallization scheme is relatively simple:

- Each processor is responsible keeping track of a fraction of all the particles.
- Each processor manipulates its own copy of the grid (including Fast Fourier Transforms)

  - The exception is the collision operator. Since the grid there is in 5D phase, space, it makes sense to split these among the processors.

- The only communication among processors is when the grid is calculated. A sum_allreduce operation is performed which adds the contribution from all the particles on each of the processors.

We perform the following scaling tests:

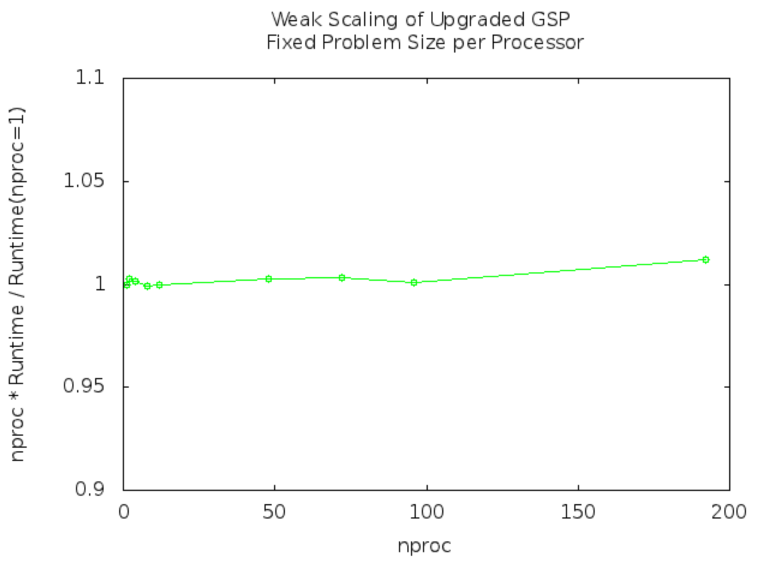
- **Weak scaling**: Increase the number of processors in proportion to the problem size. (1 to 192 cores)

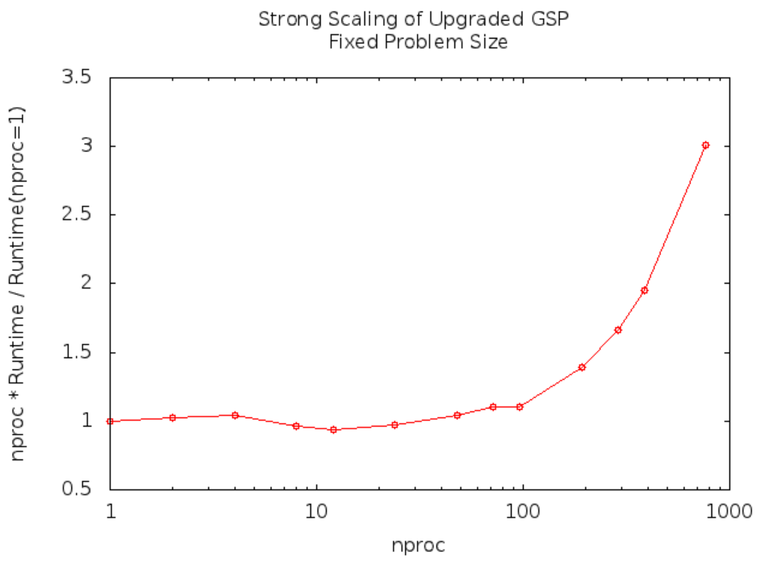
- **Strong scaling**: Increase the number processors for *fixed* problem size. (1 to 960 cores)

The former is just as important: it means that we can always add more processors to reduce noise. We do not expect this parallelization scheme to work when the number of particles *per processor* approaches the number of grid points. We have added an expensive per-particle operation (the sqrt necessary to find $v_{\parallel}$) that may cause scaling to become poor well before this point.

These tests were run on Hopper, a NSERC supercomputer, utilizing up to 960 cores. For these cases, $N_{part.percell} = 1600$ and $N_{grid} = 256$, giving 400k particles total. Therefore, we would not expect the problem to scale well as $N_{proc} \to 1600$.

The quantities plotted should be constant for perfect scaling. We find that the algorithm scales well weakly, but is sensitive to strong scaling. The inclusion of the sqrt operation is likely to blame for this decrease in performance at high numbers of processors.


Weak Scaling of Upgraded GSP
Fixed Problem Size per Processor
nproc * Runtime / Runtime(nproc=1)
1.1
1.05
1
0.95
0.9
0
50
100
150
200
nproc


Strong Scaling of Upgraded GSP
Fixed Problem Size
nproc * Runtime / Runtime(nproc=1)
3.5
3
2.5
2
1.5
1
0.5
1
10
100
1000
nproc

7 Schedule and Milestones

Phase I: Analytics
September - December 2011

- Understand rigorous derivation of the gyrokinetic equation
- Become familiar with analytical aspects of the gyrokinetic equation
- Make velocity coordinate change
- Understand algorithm used by GSP to solve GK equation
- Understand Abel collision operator and how it is to be implemented in GSP
- **Milestone:** Derive GK equation in velocity coordinates E and μ.

Phase II: Numerics
December 2011 - March 2012

- Become familiar with GSP code
- Make proposed changes to the code
- Ensure code still runs
- Code collision operator
- **Milestone:** GSP Code updated with new velocity coordinates.

Phase III: Testing and Validation
April 2012

- Debug updated GSP code
- Validate and test against previous version and known results
- Run test cases, organize results
- **Milestone:** GSP code in new velocity coordinates validated and tested

Phase IV: Communication
May 2012

- Prepare final presentation
- **Milestone:** Final presentation given

8 Deliverables

- Updated GSP source code
- Sample input file and instructions
- Test case comparison data
- Final presentation
- Final Report

9 Bibliography

- Abel, et al. "Linearized model Fokker-Planck collision operators for gyrokinetic simulations. 1. Theory." *Physics of Plasmas*, **15**:122509 (2008)
- Antonsen and Lane, "Kinetic equations for low frequency instabilities in inhomogeneous plasmas." *Physics of Fluids*, **23**:1205 (1980)
- Aydemir, "A unified Monte Carlo interpretation of particle simulations and applications to non-neutral plasmas." *Physics of Plasmas*, **1**:822 (1994)
- Barnes, et al. "Linearized model Fokker-Planck collision operators for gyrokinetic simulations. 2. Numerical Implementation and Tests." *Physics of Plasmas*, **16**:072107 (2009)
- Barnes, "Trinity: A Unified Treatment of Turbulence, Transport, and Heating in Magnetized Plasmas." PhD Thesis, University of Maryland Department of Physics (2009)
- Broemstrup, "Advanced Lagrangian Simulations Algorithms for Magnetized Plasma Turbulence." PhD Thesis, University of Maryland Department of Physics (2008)
- Catto, "Linearized gyro-kinetics." *Plasma Physics*, **20**:719 (1978)
- Dimits, et al, "Comparisons and physics basis of tokamak transport models and turbulence simulations." *Physics of Plasmas*, **7**: 969 (2000)
- Frieman and Chen, "Nonlinear gyrokinetic equations for low-frequency electromagnetic waves in general plasma equilibria." *Physics of Fluids*, **25**:502 (1982)
- Howes, et al. "Astrophysical gyrokinetics: basic equations and linear theory." *The Astrophysical Journal*, **651**: 590 (2006)
- Ricci, et al, "Gyrokinetic linear theory of the entropy mode in a Z pinch." *Physics of Plasmas*, **13**: 062102 (2006)

Appendix A: Gyroaveraging

Often, quantities will need to be "gyroaveraged" over a circle determined by the particles' velocity and the local magnetic field. This operation is defined by:

$$\langle A(\mathbf{r})\rangle_{\mathbf{R}} \equiv \int d\theta\, A(\mathbf{R}+\boldsymbol{\rho}(\theta))$$

Where $\langle A(\mathbf{r})\rangle_{\mathbf{R}}$ means the "gyroaverage of A at fixed gyrocenter $\mathbf{R}$". Likewise, we have:

$$\langle A(\mathbf{R})\rangle_{\mathbf{r}} \equiv \int d\theta\, A(\mathbf{r}-\boldsymbol{\rho}(\theta))$$

In Fourier space, we find that the gyroaverage operation becomes a Bessel function factor:

$$\langle \tilde{a}(\mathbf{k_r})\rangle_{\mathbf{R}} = J_0\left(\tfrac{k_\perp v_\perp}{\Omega}\right)\tilde{a}(\mathbf{k_R})$$

Appendix B: Poisson's Equation

The fields above are found by applying Poisson's equation, which to this order is the condition of *quasineutrality*:

$$\sum_s \delta n_s q_s = 0$$

Where the sum is over all species (typically ions and electrons). The perturbed density δn is the moment of the distribution function:

$$\delta n = \int d^3\mathbf{v} \left\langle \delta f \right\rangle_{\mathbf{r}}$$

Where we acknowledge that the charges form rings and we need to average at constant *position in space* and not at constant gyrocenter since particles of different gyrocenters will contribute to the charge at a particular point in space. Since the function h (which we eliminate in our formulism) is already a gyroaveraged quantity: $\left\langle h \right\rangle_{\mathbf{R}} = h$, we can write:

$$\delta f + \phi F_0 = \left\langle \delta f \right\rangle_{\mathbf{R}} + \left\langle \phi \right\rangle_{\mathbf{R}} F_0$$

Therefore:

$$\left\langle \delta f \right\rangle_{\mathbf{r}} = \left\langle \left\langle \delta f \right\rangle_{\mathbf{R}} \right\rangle_{\mathbf{r}} + F_0 \left\langle \left(\left\langle \phi \right\rangle_{\mathbf{R}} - \phi \right) \right\rangle_{\mathbf{r}}$$

In Fourier space, the sum over species in Poisson's equation looks like:

$$\sum_s q_s \int J_0 \left\langle \delta f \right\rangle_{\mathbf{R}} d^3\mathbf{v} + \phi \frac{n_{0,s}}{(2\pi)^{3/2}} \int e^{-\frac{1}{2}v^2} \left(J_0 - 1 \right) d^3\mathbf{v}$$

Rearranging and applying an integral identity for I_0:

$$\phi = \frac{\sum_s \int d^3\mathbf{v} J_0 \left\langle \delta f \right\rangle_{\mathbf{R,s}}}{\sum_s \frac{q_s^2 n_{0,s}}{T_s} \left[1 - e^{-k_\perp^2 \rho_s^2} I_0 \left(k_\perp^2 \rho_s^2 \right) \right]}$$

Appendix C: Algorithm for Calculating the Fields

We summarize here how the fields are found given $\left\langle \delta f \right\rangle$:

1. For each species:
   - For each μ_i:
     (a) Use bilinear interpolation to find $\left\langle \delta f \right\rangle$ on a grid in **R**-space.
     (b) In the processes of depositing the particles onto the grid, perform the Monte Carlo integration over the velocity space variable E
     (c) Fourier transform in X and Y
     (d) Multiply by J_0 (stored in memory) and normalize

- Integrate discretely over μ (mid-point rule)

2. Sum over species to find ϕ
3. Fourier transform over Z
4. Calculate $\langle \mathbf{E} \rangle_{\mathbf{R}} = -i\mathbf{k}J_0\phi$
5. Inverse Fourier transform in 3D
6. Reinterpolate so that the field is known at each particles' position.

Appendix D: Algorithm for the Collision Operator

The algorithm to calculate the collision operator is reminiscent of the fields, except we are working on a 5D grid.

1. Calculate ϕ for the particles' current positions and weights.
2. Deposit particles onto a 5D phase-space grid
    - Use bilinear interpolation in X, Y, and Z
    - Use nearest-neighbor interpolation for v and ξ
3. Convert from g to h using ϕ found on spatial grid
4. Fourier transform on X and Y
5. Invert saved tridiagonal matrix to apply L operator
6. Use the same matrix and output from Step 5 to apply the U_L terms with the Sherman-Morrison formulism
7. Invert saved tridiagonal matrix to apply D operator
8. Use the same matrix and output from Step 7 to apply the U_D and U_E terms with the Sherman-Morrison formulism
9. Inverse Fourier transform
10. Convert from h back to g
11. Reinterpolate g to each particle's position using weighting scheme that nullifies effect as $\nu \to 0$

Appendix E: Normalization and Generation of the Monte Carlo Probability Distribution

We have defined $p(E) = A\frac{e^{-E}}{\sqrt{E-\mu B}}$. Since $\int\limits_{\mu B}^{E_{max}} p(E)dE = 1$ necessarily, we find the normalization constant:

$$\frac{1}{A} = e^{-\mu B}\mathrm{Erf}\left(\sqrt{E_{max} - \mu B}\right)$$

Furthermore, in order to obtain this distribution from a standard [0,1] uniform random distribution, we need to invert the cumulative distribution:

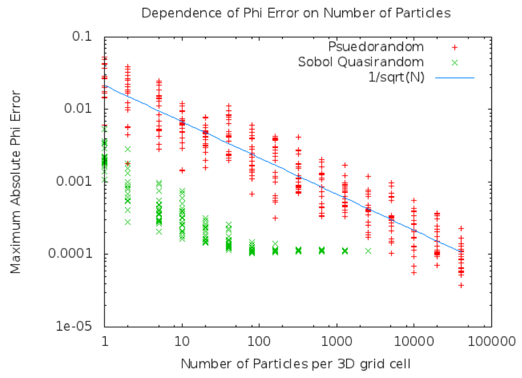
$$y = P(x) = A\int\limits_{\mu B}^{x} \frac{dEe^{-E}}{\sqrt{E-\mu B}}$$

To obtain:

$$x = \mu B + \left(\mathrm{Erf}^{-1}\left[y\mathrm{Erf}\left(\sqrt{E_{max} - \mu B}\right)\right]\right)^2$$

Appendix F: Quasirandom Sequence Monte Carlo Improvement

It was discovered that we can get better than the typical $\frac{1}{\sqrt{N}}$ error dependence from Monte Carlo by uniformly filling the integration domain with particles distributed according to a *quasirandom sequence* rather than pseudorandom or truly random placement. This avoids errors that result from a spurious distribution that would occur by chance.



While the implementation appears to be imperfect, the decrease in error is worth note.