Implementation Notes for Gyrokinetic Particle Pusher

Norman M. Cao

March 31, 2025

1 Equations of motion and geometry

1.1 Gyrokinetic characteristic equations

We aim to push particles in electrostatic gyrokinetics. The equations of motion are given by

$$B_\parallel^* \dot{\mathbf{R}} = \frac{1}{q}\hat{\mathbf{b}} \times \nabla H + v_\parallel \mathbf{B}^* \tag{1a}$$

$$B_\parallel^* \dot{p}_\parallel = -\mathbf{B}^* \cdot \nabla H \tag{1b}$$

$$\dot{\mu} = 0 \tag{1c}$$

With some definitions

$$\hat{\mathbf{b}} := \mathbf{B}/B \tag{2a}$$

$$\mathbf{B}^* := \mathbf{B} + \nabla \times (p_\parallel \hat{\mathbf{b}}/q) \tag{2b}$$

$$B_\parallel^* := \hat{\mathbf{b}} \cdot \mathbf{B} \tag{2c}$$

$$H = p_\parallel^2/2m + \mu B + q\mathcal{J}[\Phi] \tag{3a}$$

$$v_\parallel := \partial_{p_\parallel} H = p_\parallel/m \tag{3b}$$

and m, q are the species mass and charge respectively.

1.2 Cylindrical coordinates

We primarily work in a right-handed (R, φ, Z) cylindrical coordinate system. Thus, φ points into the (R, Z) plane. Recall that for a path $\mathbf{R} = (R(t), \varphi(t), Z(t))$, we have that

$$\dot{\mathbf{R}} = \dot{R}\hat{\mathbf{R}} + \dot{Z}\hat{\mathbf{Z}} + R\dot{\varphi}\hat{\boldsymbol{\varphi}}$$
$$\ddot{\mathbf{R}} = (\ddot{R} - R\dot{\varphi}^2)\hat{\mathbf{R}} + \ddot{Z}\hat{\mathbf{Z}} + (R\ddot{\varphi} + 2\dot{R}\dot{\varphi})\hat{\boldsymbol{\varphi}}$$

Typically we will work with the orthonormal basis of unit vectors $(\hat{\mathbf{R}}, \hat{\boldsymbol{\varphi}}, \hat{\mathbf{Z}})$. From the above expressions, the velocity acts on coordinates as

$$\dot{R} = \dot{\mathbf{R}} \cdot \hat{\mathbf{R}} \qquad \dot{Z} = \dot{\mathbf{R}} \cdot \hat{\mathbf{Z}} \qquad \dot{\varphi} = (\dot{\mathbf{R}} \cdot \hat{\boldsymbol{\varphi}})/R$$

then, the acceleration acts on components of the velocity vector (in the orthonormal basis) as

$$(\dot{\mathbf{R}} \cdot \hat{\mathbf{R}})' = \ddot{\mathbf{R}} \cdot \hat{\mathbf{R}} + R\dot{\varphi}^2$$
$$(\dot{\mathbf{R}} \cdot \hat{\mathbf{Z}})' = \ddot{\mathbf{R}} \cdot \hat{\mathbf{Z}}$$
$$(\dot{\mathbf{R}} \cdot \hat{\boldsymbol{\varphi}})' = \dot{R}\dot{\varphi} + R\ddot{\varphi} = \ddot{\mathbf{R}} \cdot \hat{\boldsymbol{\varphi}} - \dot{R}\dot{\varphi}$$

1.3 Magnetic field

We use the following representation of the magnetic field and current

$$\mathbf{B} = F(\psi)\nabla\varphi + \nabla\varphi \times \nabla\psi = \frac{F(\psi)\hat{\boldsymbol{\varphi}} + \hat{\boldsymbol{\varphi}} \times \nabla\psi}{R}$$
$$\nabla \times \mathbf{B} = F'\nabla\psi \times \nabla\varphi + \nabla \times (\nabla\varphi \times \nabla\psi)$$

where ψ is the poloidal flux. Note the second term in the current can be evaluated using an in-plane curl

It's useful to have the following representations for certain terms in the gyrokinetic equation:

$$\nabla B = \frac{\nabla(RB) - B\nabla R}{R}$$
$$2RB\nabla(RB) = \nabla(R^2B^2) = \nabla(F^2 + |\nabla\psi|^2) = 2F'\nabla\psi + 2\,\mathrm{Hess}[\psi]\nabla\psi$$
$$\nabla \times \hat{\mathbf{b}} = \frac{B(\nabla \times \mathbf{B}) - (\nabla B) \times \mathbf{B}}{B^2}$$

note that these can be written purely in terms of analytic derivatives of $\psi(R, Z)$ and $F(\psi)$.

2 C^1 interpolation on unstructured meshes

2.1 Field line tracing

Let $\vec{R}_B(R, Z; \varphi) = (R_B(...), Z_B(...))$ be the motion of the field-line trace starting at (R, Z) on a poloidal plane, parameterized by the toroidal angle φ moved along the field line.

$\vec{R}_B$ satisfies the ODE

$$\frac{\mathrm{d}\vec{R}_B}{\mathrm{d}\varphi} = \vec{b}_p \circ \vec{R}_B := \left.\frac{R\mathbf{B}_p}{B_t}\right|_{\vec{R}_B} = \left.\frac{R\hat{\boldsymbol{\varphi}} \times \nabla\psi}{F(\psi)}\right|_{\vec{R}_B}; \qquad \vec{R}_B(R, Z; 0) = (R, Z)$$

This ODE is essentially a reparameterization of the magnetic field line ODEs with φ as time. For $\frac{\mathrm{d}}{\mathrm{d}\varphi}$ we think of (R, Z) as being parameters. These ODEs have an associated variational equation

$$\frac{\mathrm{d}[D\vec{R}_B]}{\mathrm{d}\varphi} = ([D\vec{b}_p] \circ \vec{R}_B)[D\vec{R}_B]; \qquad D\vec{R}_B(R, Z; 0) = I_{2\times 2}$$

here we think of D as the differential in (R, Z) with φ as a parameter, that is:

$$D\vec{b}_p = \begin{bmatrix} \partial_R(\vec{b}_p \cdot \hat{\mathbf{R}}) & \partial_Z(\vec{b}_p \cdot \hat{\mathbf{R}}) \\ \partial_R(\vec{b}_p \cdot \hat{\mathbf{Z}}) & \partial_Z(\vec{b}_p \cdot \hat{\mathbf{Z}}) \end{bmatrix}$$

$$D\vec{R}_B = \begin{bmatrix} \partial_R R_B & \partial_Z R_B \\ \partial_R Z_B & \partial_Z Z_B \end{bmatrix}$$

2.2 Field-aligned interpolation

Suppose we are trying to interpolate $\phi(R, \varphi, Z)$ knowing its values on some equally spaced poloidal planes $\phi_i(R, Z)$. This can be accomplished by

$$\phi(R, \varphi, Z) = \sum_i p_i(\varphi)\phi_i(\vec{R}_B(R, Z; \varphi_i - \varphi))$$

here p_i are some piecewise polynomial basis functions. We can compute its gradient by

$$\nabla\phi = \sum_i \left[p'(\phi_i \circ \vec{R}_B)\nabla\varphi + p_i\nabla(\phi_i \circ \vec{R}_B)\right]$$

Using the chain rule,

$$\begin{aligned}\nabla(\phi_i \circ \vec{R}_B) &= \begin{bmatrix}\nabla R & \nabla Z\end{bmatrix} [D(\phi_i \circ \vec{R}_B)]^T - \nabla\varphi \left(\frac{\mathrm{d}(\phi_i \circ \vec{R}_B)}{\mathrm{d}\varphi}\right) \\ &= \begin{bmatrix}\hat{\mathbf{R}} & \hat{\mathbf{Z}}\end{bmatrix} [D\vec{R}_B]^T [[\nabla\phi_i] \circ \vec{R}_B] - \frac{\hat{\boldsymbol{\varphi}}}{R} \left([[\nabla\phi_i] \circ \vec{R}_B] \cdot \frac{\mathrm{d}\vec{R}_B}{\mathrm{d}\varphi}\right)\end{aligned}$$

Note that it's possible to show that $\mathbf{B} \cdot \nabla(\phi_i \circ \vec{R}_B) = 0$.

2.3 Choice of basis functions

Traditional cubic spline interpolation, which minimizes 'bending' and C^2 continuity, has polynomial coefficients which are computed by solving a linear system involving all of the data points as well as boundary conditions. Instead we rely on polynomial splines which involve only the 4 points in the neighborhood of any φ, and generally enforce C^1 continuity at the nodes.

Two options are considered. The first is cubic Hermite interpolation, with an array of polynomial coefficients

$$p = \begin{bmatrix} 0 & -1/2 & 1 & -1/2 \\ 1 & 0 & -5/2 & 3/2 \\ 0 & 1/2 & 2 & -3/2 \\ 0 & 0 & -1/2 & 1/2 \end{bmatrix}$$

where $p_i(t) = \sum_{j=0}^{3} p_{ij} t^j$ (here the array entries are being 0-indexed). This scheme exactly interpolates the nodes and also enforces C^1 continuity at the nodes with a value of the derivative given by the centered difference of the adjacent two nodes.

The second is a quadratic smoothing spline,

$$p = \begin{bmatrix} 1/4 & -1/2 & 1/4 \\ 1/2 & 0 & -1/4 \\ 1/4 & 1/2 & -1/4 \\ 0 & 0 & 1/4 \end{bmatrix}$$

This scheme can be thought of as the anti-derivative of linear interpolation on the derivative, computed via centered difference, while enforcing C^1 continuity at the nodes. The quadratic dependence sacrifices exact interpolation at the nodes in exchange for a derivative with less oscillations.

Finally we remark that parallel noise seems to be non-negligible; in theoretical cases, a Lanczos filter is applied along the field line to smooth out these high-frequency parallel fluctuations.

2.4 Interpolation on poloidal planes

Interpolation on poloidal planes uses rHCT elements, which are C^1 elements that minimize a 'bending energy'. The code is essentially a fork of the matplotlib `CubicTriInterpolator`[1] with a few optimizations.

3 Extraction of Ballooning Coefficients

3.1 Straight Field-line Coordinates

On the closed flux surfaces, let θ_g be the geometric poloidal angle relative to the magnetic axis, with $\theta_g = 0$ representing the outboard midplane. Using $F(\psi) = B_t/R$, we can compute the relationship between the

[1] `https://matplotlib.org/stable/api/tri_api.html#matplotlib.tri.CubicTriInterpolator`

straight field-line angle θ in terms of θ_g by

$$\theta = \frac{1}{q(\psi)} \int_{\theta_0(\psi)}^{\theta_g} \frac{F(\psi)}{B_p(\psi, \theta'_g)} \ell'(\theta'_g) \, \mathrm{d}\theta'_g$$
$$q(\psi) = \frac{1}{2\pi} \int_0^{2\pi} \frac{F(\psi)}{B_p(\psi, \theta'_g)} \ell'(\theta'_g) \, \mathrm{d}\theta'_g$$

where $\ell'(\theta_g)$ is the derivative of the arclength of the field line along the flux surface with respect to θ_g, and $\theta_0(\psi)$ is the arbitrary offset on each flux surface where $\theta = 0$ lies. An easy choice is $\theta_0(\psi) = 0$, which results in $\theta = 0$ being the outboard midplane. This relationship is then numerically inverted to get θ.

3.2 Ballooning Transform

Moving to flux coordinates (ψ, ζ, θ) with $\zeta = \varphi$, we can always write

$$\phi(R, \varphi, Z) = \sum_{n=-\infty}^{\infty} e^{in\zeta} \phi_n(\psi, \theta)$$

We can move to the covering space $\theta \mapsto \eta$, let $\rho = nq$, and introduce the eikonal factor

$$\phi_n(\psi, \theta) = \sum_{\ell=-\infty}^{\infty} \hat{\phi}_n(\psi, \eta + 2\pi\ell)$$
$$\hat{\phi}_n(\psi, \eta) = e^{-i\rho\eta} f_n(\rho, \eta)$$
$$f_n(\rho, \eta) = \int_{-\infty}^{\infty} \mathrm{d}\theta_k \, e^{i\rho\theta_k} \tilde{f}_n(\theta_k, \theta)$$

the complete representation is then

$$\phi(R, \varphi, Z) = \sum_{n=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{d}\theta_k \, e^{in(\zeta - q(\eta - \theta_k + 2\pi\ell))} \tilde{f}_n(\theta_k, \theta + 2\pi\ell)$$

Now, if we had an exact ballooning symmetry, we would have for a fixed parameter θ_k (related to the radial wavenumber),

$$f_n(\rho, \eta) = e^{i\rho\theta_k} \tilde{F}_n(\eta)$$

This is exactly analogous to $f(x) = \tilde{F} e^{ikx}$ with k is a fixed parameter. Our goal is to find an approximation to $\tilde{f}_n$.

Taking $S = n(\zeta - q\eta)$, we can compute

$$\nabla(e^{iS} f_n(\rho, \eta)) = e^{iS} \left[i\nabla S f_n + \partial_q f_n \nabla q + \partial_\eta f_n \nabla\eta \right]$$
$$\nabla S = n(\nabla\zeta - q\nabla\eta - \eta\nabla q)$$

Note that in the ballooning representation, we also have

$$\tilde{S} = n(\zeta - q(\eta - \theta_k))$$
$$\nabla(e^{i\tilde{S}} \tilde{f}_n(\theta_k, \eta)) = e^{i\tilde{S}} (i\nabla\tilde{S} \tilde{f}_n + \partial_\eta \tilde{f}_n \nabla\eta)$$
$$\nabla\tilde{S} = n(\nabla\zeta - q\nabla\eta - (\eta - \theta_k)\nabla q)$$

Note furthermore we have the gyroaverage operator taking the eikonal approximation

$$\left\langle e^{i\tilde{S}} \tilde{f}_n(\theta_k, \eta) \right\rangle \approx e^{i\tilde{S}} J_0(k_\perp \rho) \tilde{f}_n(\theta_k, \eta)$$
$$\mathbf{k}_\perp = \nabla\tilde{S}$$

3.3 Up-down Symmetry

One final thing we do is to modify θ_0 to account for up-down symmetry. We compute $v_r = \mathbf{v}_D \cdot \hat{\mathbf{r}}$ and $v_y = \mathbf{v}_D \cdot \hat{\mathbf{y}}$ where $\hat{\mathbf{y}} = \hat{\mathbf{r}} \times \hat{\mathbf{b}}$ is the binormal vector. We take $\theta_0 = \operatorname{argmax}_\theta v_y$.

3.4 Approximation by Gauss-Hermite Functions

Finally, we approximate $f_n(\rho, \eta)$ (equivalently $\tilde{f}_n(\theta_k, \eta)$) with complex-amplitude Gauss-Hermite functions.

Note that naively, if we try to extract ϕ_n from the data ϕ, aliasing issues cause pollution in the toroidal mode number spectrum. Instead, we first upsample the toroidal resolution using field-aligned interpolation (rather than numerically following field lines, we use constant $\alpha = \zeta - q\theta$), perform the FFT, then truncate in frequency space. Additionally, we apply a Lanczos filter in the parallel direction to reduce high-frequency parallel noise – not sure what its source is.

To compute gradients of ϕ, note the coordinate transform

Define v_D^q, v_D^α by

$$\mathbf{v}_D \cdot \nabla = v_D^q \partial_q + v_D^\alpha \partial_\alpha = v_D^R \partial_R + v_D^Z \partial_Z + v_D^\varphi \partial_\varphi$$

Note that $v_D^\varphi = \mathbf{v}_D \cdot \hat{\boldsymbol{\varphi}}/R$.

We have the following coordinate transforms

$$q = q(\psi) \qquad \alpha = \zeta - q\theta \qquad \eta = \theta$$

$$\begin{bmatrix} \mathrm{d}q \\ \mathrm{d}\alpha \\ \mathrm{d}\eta \end{bmatrix} = \begin{bmatrix} q' & 0 & 0 \\ -q'\theta & 1 & -q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{d}\psi \\ \mathrm{d}\zeta \\ \mathrm{d}\theta \end{bmatrix} = \begin{bmatrix} q' & 0 & 0 \\ -q'\theta & 1 & -q \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \psi_R & 0 & \psi_Z \\ 0 & 1 & 0 \\ \theta_R & 0 & \theta_Z \end{bmatrix} \begin{bmatrix} \mathrm{d}R \\ \mathrm{d}\varphi \\ \mathrm{d}Z \end{bmatrix}$$