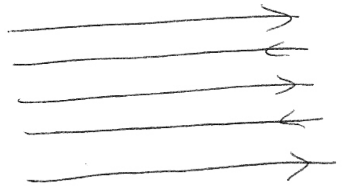
Implementation of Large Scale *E* x *B* Shear Flow in the GS2 Gyrokinetic Turbulence Code

G.W. Hammett (PPPL)
W. Dorland (U. Maryland)
N.F. Loureiro (Maryland/PPPL)
T. Tatsuno (Maryland)

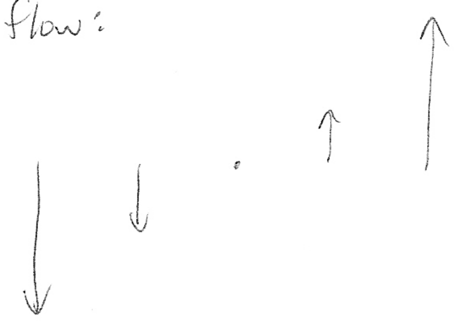
APS-DPP Meeting
Philadelphia, Oct. 30 - Nov.3, 2006
(includes updates 11/22/2006)

Implementation of Equilibrium-Scale ExB shear in GS2

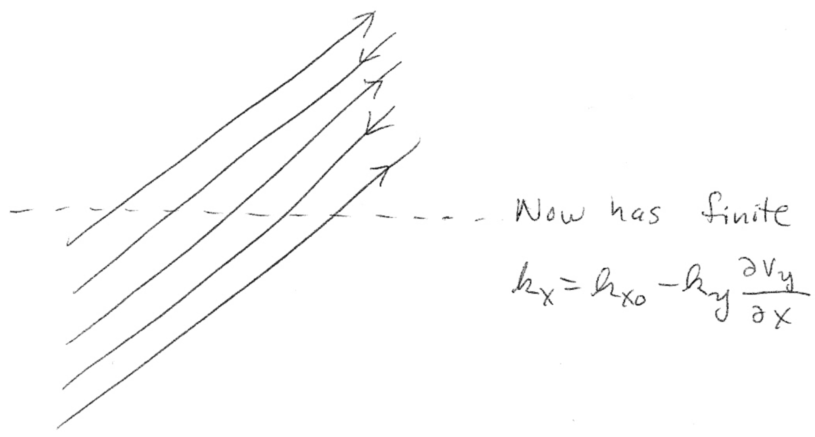
Eddies initially with $k_x = 0$:



+ sheared flow:



At a later time:



GS2 solves Eq. of form:

$$\frac{\partial}{\partial t} f\left(k_x, k_y, \theta, E, \mu, t\right) = L\left(k_x, k_y, \theta, E, \mu\right) f\left(k_x, k_y, \ldots, t\right) + NonlinTerms$$

Include equilbrium scale ExB shear by substitution:

$$k_x \rightarrow k_x\left(t\right) = k_{x0} - k_y \frac{\partial v_y}{\partial x} t$$

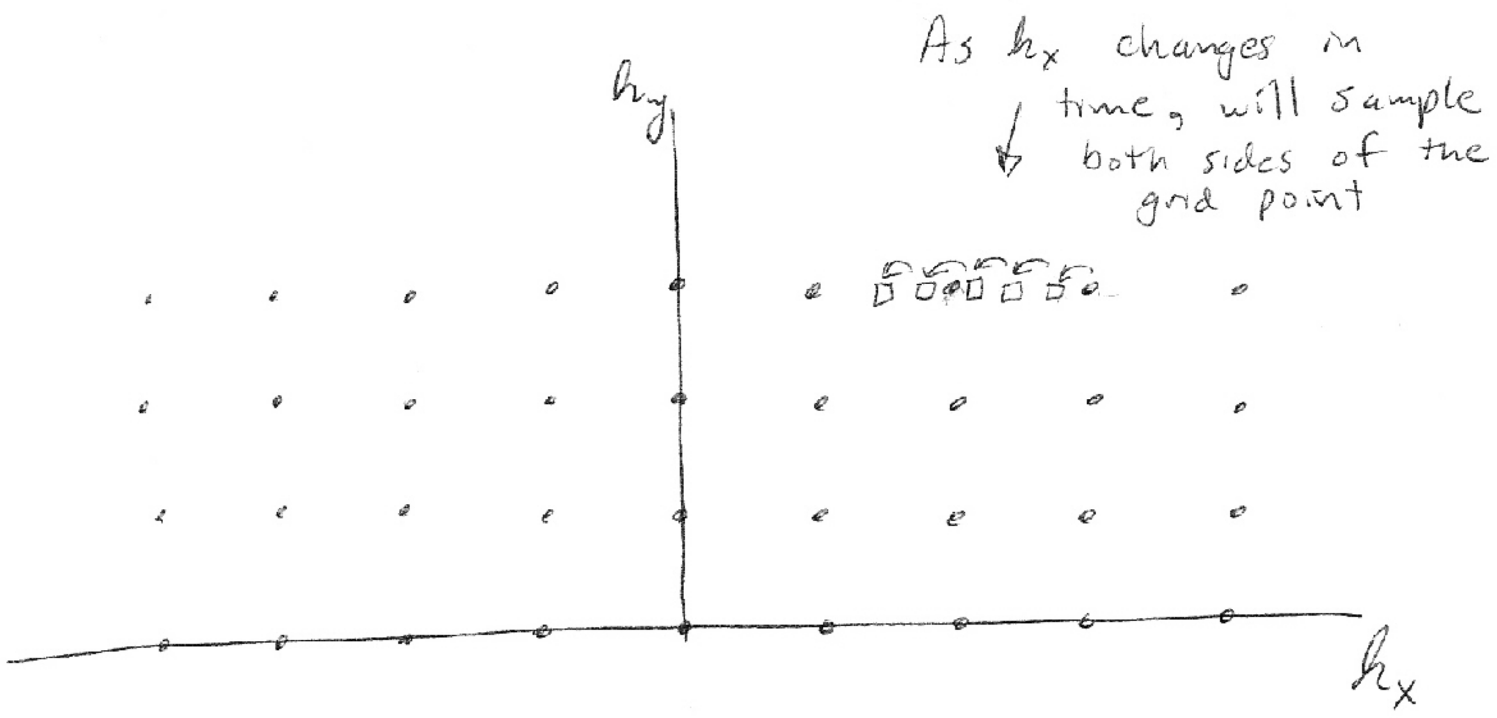
$$\frac{\partial}{\partial t} f\left(k_x(t), k_y, \theta, \ldots, t\right) = \underbrace{L\left(k_x(t), k_y, \theta, \ldots\right)} f\left(k_x(t), k_y, \ldots, t\right) + NonlinTerms$$

GS2 treats all linear terms implicitly. Recalculating all of the implicit response matrices every time step when k_x changes a little bit would be very expensive.

Instead of recalculating response matrices, just use Nearest-Grid-Point interpolations. Most easily implemented in reverse:

Whenever $k_x(t)$ differs from k_{x0} by more than $\Delta k_x/2$, shift f :

$$f\left(k_{xi}, k_y\right) \Leftarrow f\left(k_{xi} + \Delta k_x, k_y\right)$$


As k_x changes in time, will sample both sides of the grid point
k_y
k_x

∴ Errors will tend to cancel over time, making the method quasi-2^{cd} order accurate.

(Like Nearest Grid Point NGP that is widely used in Gyrokinetic PIC codes, or somewhat like Godunov splitting).

(Could generalize to true 2cd order accuracy by doing 2 iterations per time step and interpolating. Have to be careful: can interpolate the operator *L*, which is a smooth function of k_x, but the distribution function *f* is not a smooth function of k_x.)

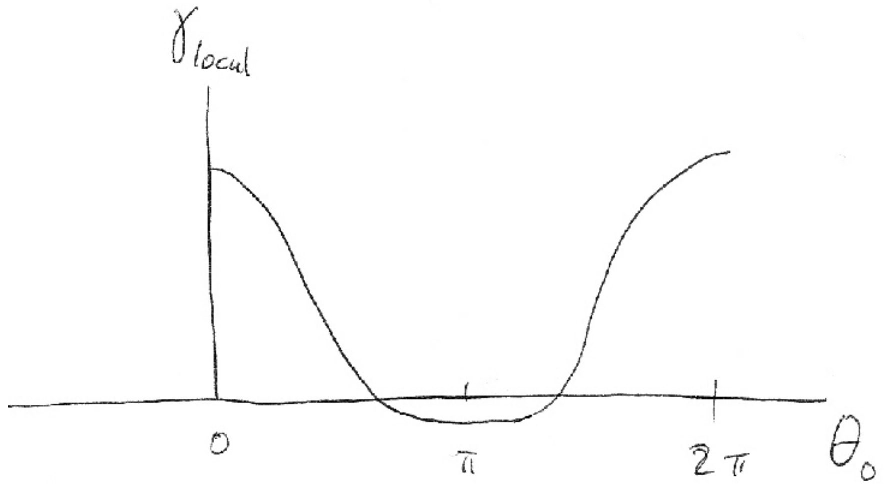
With sheared magnetic field:

$$k_x(t) = \underset{\overset{|||}{-k_y \hat{s}\theta_0}}{k_{xi}} + k_y \hat{s}\theta - k_y \frac{\partial v_y}{\partial x} t$$

$$k_x(t) = k_y \hat{s}(\theta - \theta_0) - \underbrace{k_y \frac{\partial v_y}{\partial x} t}_{}$$

EXB shear twists the eddy.

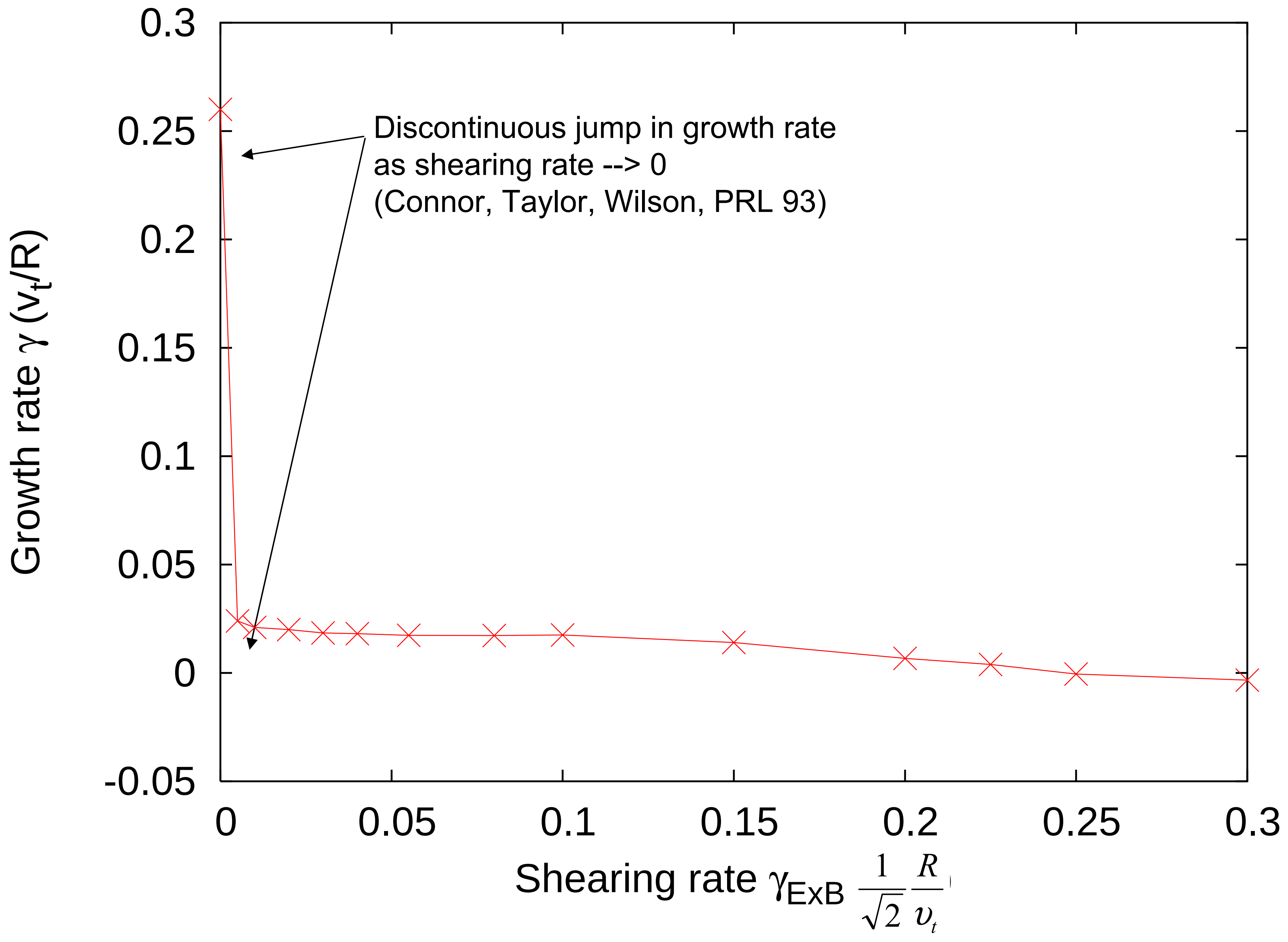
Can undo this twist by moving in θ along the sheared ~~mag~~ magnetic field.

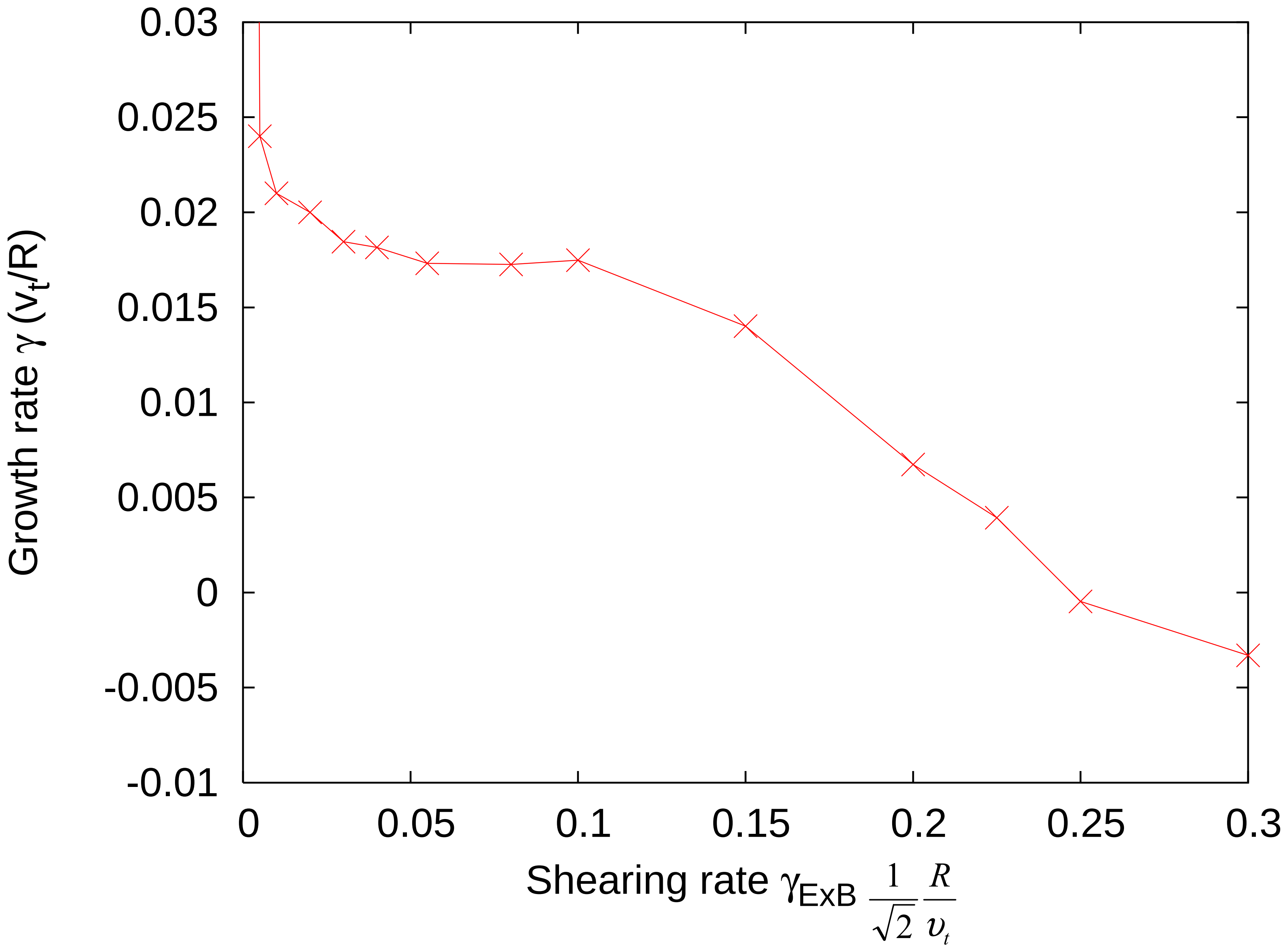


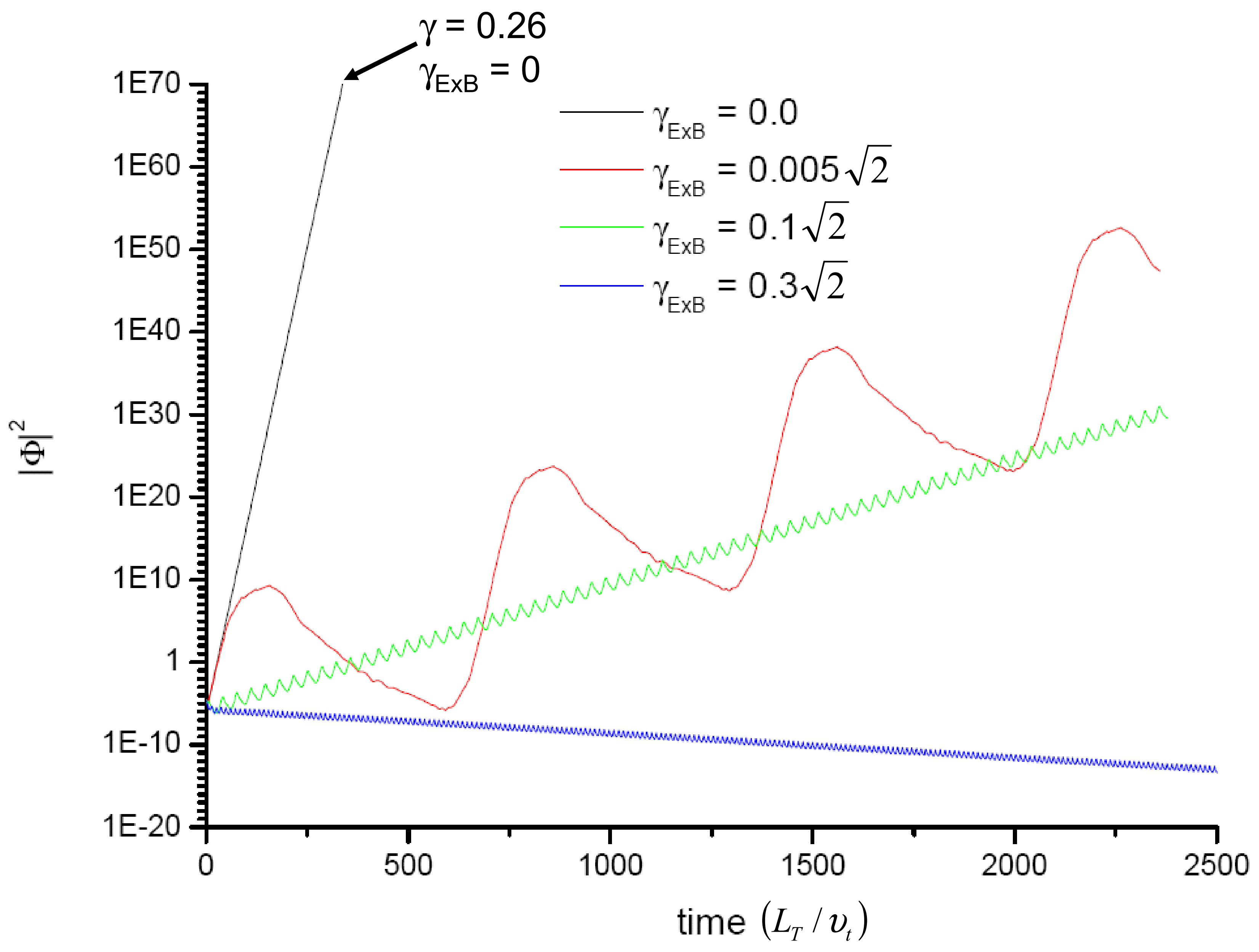
As mode moves along θ_0, the time-averaged growth rate changes ~~dis~~ discontinuously between

$$\gamma_{ExB} = \frac{\partial v_y}{\partial x} \neq 0 \quad \text{and} \quad \gamma_{ExB} = 0$$

But not really relevant if mode grows to nonlinear amplitude before moving out of the bad curvature region.


Discontinuous jump in growth rate
as shearing rate --> 0
(Connor, Taylor, Wilson, PRL 93)
Growth rate γ (v_t/R)
Shearing rate $\gamma_{ExB} \frac{1}{\sqrt{2}} \frac{R}{v_t}$
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3


0.03
0.025
0.02
0.015
0.01
0.005
0
-0.005
-0.01
0
0.05
0.1
0.15
0.2
0.25
0.3
Growth rate γ (v$_t$/R)
Shearing rate $\gamma_{ExB}\,\frac{1}{\sqrt{2}}\frac{R}{v_t}$



Floquet mode representation shows large growth of wave packet while in bad curvature region, even though time-averaged growth rate is small. (At weak shear, the fluctuations will go nonlinear long before shear has any effect.)

Hammett, Dorland, Loureiro

References:

Waltz, Dewar, Garbet, Phys. Plasmas 5, 1784 (1998)

Connor, Taylor, Hastie, Plasma Phys. Control. Fusion 46, B1 (2004)

and references therein.

The results presented in these slides have been slightly updated from those shown at the APS meeting, to include a small amount of collisions. GS2's normalized collisionality is set in these runs to vnewk=0.01, much less than the growth rate in the absence of ExB shear of γ=0.26.

Implementation of Large Scale $E \times B$ Shear Flow in the GS2 Gyrokinetic Turbulence Code

Gregory W. Hammett[1], W. Dorland[2], N.F Loureiro[1,2], T. Tatsuno[2]

[1] *Princeton University Plasma Physics Laboratory*

[2] *Univ. of Maryland*

Abstract

GS2 is a 5-D gyrokinetic flux-tube code for studying turbulence in general geometry plasmas. It has always self-consistently included small-scale zonal flows (with radial wavelengths smaller than the simulation domain) that are driven by the turbulence, but for simplicity it neglected equilibrium-scale zonal flows (which are not radially periodic on the simulation scale) that can be driven by beam injection, neoclassical effects, etc. Shearing rate estimates of corrections due to equilibrium-scale flows could be made based on earlier gyrofluid and gyrokinetic nonlinear simulations by Waltz et al. and Dimits et al. Large-scale sheared flows can be implemented directly in a flux-tube code by transforming to moving coordinates where the physical radial wavenumber k_x is related to the radial wavenumber k'_x in shearing coordinates by $k_x(t) = k'_x - k'_y \gamma_{E\times B} t$, where $\gamma_{E\times B}$ is the shearing rate. Direct implementation of this is challenging in GS2's implicit algorithm, as that would require a large overhead of recalculating the implicit response matrices every time step. By discretizing the effective k_x, we can avoid this large overhead, and yet still be effectively 2cd order accurate (similar to how Godunov splitting is effectively 2cd order accurate) and it converges well for typical parameters. Examples and tests of the GS2 implementation will be presented.

1 Setting up the gyrokinetic equations with sheared-flow

Write the usual gyrokinetic equation in the generic form:

$$\frac{\partial f}{\partial t} = S[f] \tag{1}$$

where f is the distribution function and S is all of the usual linear and nonlinear terms in the gyrokinetic equation that operate on f. GS2 uses a pseudo-spectral representation for f in the two perpendicular directions:

$$f(x, y, t) = \sum_k f_k(t) e^{i(k_x x + k_y y)}$$

(the dependence of f on 3 other phase-space coordinates is not explicitly denoted here) and thus solves an equation of the form:

$$\frac{\partial f_k}{\partial t} = S_k[f] \tag{2}$$

A centered 2cd-order time advancement algorithm for this equation is:

$$\frac{f_k^{n+1} - f_k^n}{\Delta t_{n+1/2}} = \frac{1}{2}(S_k[f^{n+1}] + S_k[f^n]) \tag{3}$$

GS2 has a sophisticated set of algorithms for solving this equation, using implicit methods for the linear parts of the RHS and Pseudo-Spectral/explicit Adams-Bashforth methods for the nonlinear parts of the RHS.

To handle an equilibrium scale sheared $E \times B$ flow, we need to modify this equation to:

$$\frac{\partial f}{\partial t} + v_{y0}(x)\frac{\partial f}{\partial y} = S[f] \tag{4}$$

(here I'm using slab-like notation, where x is the radial direction and y is the mostly-poloidal perpendicular direction, but this is all completely general if interpreted as field-line following Clebsch coordinates). Assuming a uniform sheared flow over the scale of the flux tube simulation, this gives:

$$\frac{\partial f}{\partial t} = -x\gamma_E \frac{\partial f}{\partial y} + S[f] \tag{5}$$

(Although I'm using slab-like coordinates here for simplicity, this is the form of the sheared flow term in general geometry as well, leading to just a constant shearing-rate term like this in Clebsch coordinates.)

This sheared equilibrium-scale flow can not be treated in the standard flux-tube approach with standard periodic boundary conditions, because $v_{y0}(x) = x\gamma_E$ is not radially periodic. (Of course flux-tube simulations automatically can handle small-scale sheared flows that are periodic on the scale of the simulation that are self-consistently generated by the turbulence.)

One way to handle non-periodic large-scale sheared flows in a finite difference code would be to modify the radial boundary conditions from the standard

$$f(x, y, t) = f(x + L_x, y, t)$$

to

$$f(x, y + tv_{y0}(x), t) = f(x + L_x, y + tv_{y0}(x + L_x), t) \tag{6}$$

This is similar to the Kotschenreuther's twist-and-shift boundary conditions used to handle magnetic shear in non-spectral codes in configuration space. This is also how ZEUS and other astrophysical MHD codes do "shearing box" simulations of turbulent accretion driven by the Magneto-Rotational Instability.

Another approach is to go to coordinates that move with the flow:

$$\begin{aligned} x' &= x \\ y' &= y - x\gamma_E t \\ t' &= t \end{aligned} \tag{7}$$

so that the time derivative transforms as:

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial t'} - x'\gamma_e \frac{\partial f}{\partial y'}$$

and this second term cancels the shearing term in Eq.(5). One can now use simple radial periodicity in these moving coordinates: $f(x', y', t') = f(x' + L_x, y', t')$. The Poisson-bracket nonlinearities in the gyrokinetic equations are invariant to such coordinate transformations (as described in a paper by Cowley et al. on field-line following coordinates),

$$\{\Phi, f\} = \frac{\partial \Phi}{\partial x}\frac{\partial f}{\partial y} - \frac{\partial \Phi}{\partial y}\frac{\partial f}{\partial x} = \frac{\partial \Phi}{\partial x'}\frac{\partial f}{\partial y'} - \frac{\partial \Phi}{\partial y'}\frac{\partial f}{\partial x'}$$

but the $\omega_d = v_d \cdot \nabla$ operator, etc., do become a bit more complicated (essentially because the radial wavenumber k_x becomes a function of time). We will look at this issue for codes that use a spectral representation.

Because radial periodicity can be assumed in the moving shearing coordinates, we can use a spectral representation in those coordinates:

$$f(x', y', t') = \sum_k f_k(t') e^{i(k_x x' + k_y y')} \tag{8}$$

Transforming back to the unmoving coordinates, this becomes:

$$\begin{aligned} f(x, y, t) &= \sum_k f_k(t) e^{i(k_x x + k_y (y - x\gamma_E t))} && (9) \\ &= \sum_k f_k(t) e^{i((k_x - k_y \gamma_E t) x + k_y y)} && (10) \\ &= \sum_k f_k(t) e^{i(k_{x*}(t) x + k_y y)} && (11) \end{aligned}$$

One can thus think of this as a spectral basis where the radial wave number is a function of time, $k_{x*}(t) = k_x - k_y \gamma_E t$.

Taking the time derivative of f in this representation gives:

$$\frac{\partial f}{\partial t} = \sum_k \left[\frac{\partial f_k}{\partial t} - i \frac{\partial k_{x*}}{\partial t} x f_k \right] e^{i(k_{x*}(t) x + k_y y)} \tag{12}$$

Substituting this into Eq.(5), the second term above cancels the shearing term, so that Eq.(5) becomes:

$$\frac{\partial f_{\vec{k}}}{\partial t} = S_{\vec{k}_*(t)}[f] \tag{13}$$

This is exactly the form of the original gyrokinetic equation that GS2 is all set up to solve, except that the radial wave number has become a function of time, so that everywhere in the original equations that k_x appeared, we now replace it with $k_{x*}(t) = k_x - k_y \gamma_E t$. A centered time advance algorithm for this is

$$\frac{f_k^{n+1} - f_k^n}{\Delta t_{n+1/2}} = S_{\vec{k}_*(t_{n+1/2})}[f^{n+1/2}] = \frac{1}{2} \left(S_{\vec{k}_*(t_{n+1/2})}[f^{n+1}] + S_{\vec{k}_*(t_{n+1/2})}[f^n] \right) \tag{14}$$

While GS2 has the nice feature of having a robust implicit algorithm for solving the linear terms in this equation, this implicit algorithm has a large overhead in calculating the response matrices that have to inverted. These response matrices depend on k_x, and thus in the shearing case depend on $k_{x*}(t)$, and there would be a huge overhead involved if these response matrices had to be recalculated every time step as $k_{x*}(t)$ changed. Instead we will make

2 small but critical changes in this algorithm to eliminate this large overhead and to simplify the way it can be implemented in GS2 with a minimum amount of changes needed to other parts of the existing code. These changes are (1) instead of recalculating response matrices for the exact value of $k_{x*}(t)$, just use the response matrices for the one of the discretized values of k_x for which S_k has already been calculated, and (2) instead of keeping track of which discretized value of k_{x*} corresponds to a given f_k at a given time, we will periodically remap f_k, shifting it on the grid in k_x so that the nearest discretized value of $k_{x*}(t)$ is always just k_x.

Figure 1: This can be illustrated by some figures... showing the fixed (k_x, k_y) grid, and showing at some time the $(k_{x*}(t), k_y)$ grid slightly shifted from it.

1.1 The essential steps of the algorithm

More concretely, the equations used to implement this are the following. Express $k_{x*}(t) = k_x - k_y \gamma_E t$ as

$$k_{x*}(t) = k_x + s(k_y, t)$$

Calculate the wave number shift factor s at the new midpoint time as:

$$s(k_y, t_{n+1/2}) = s(k_y, t_{n-1/2}) - k_y \gamma_E \Delta t_n$$

If $s(k_y)$ at the new time step gets too big, larger than half the grid spacing in the radial wave number grid, $\Delta k_x = 2\pi / L_x$, then shift everything back to the nearest k_x grid point, by calculating

$$\texttt{jump}(k_y) = \texttt{nint}(s(k_y)/\Delta k_x)$$

$$s(k_y) = s(k_y) - \texttt{jump}(k_y)\Delta k_x$$

$$f(k_x, k_y) = f(k_x - \texttt{jump}(k_y)\Delta k_x, k_y)$$

(`nint` is a function that returns the nearest integer).

In addition to shifting the distribution function $f(k_x, k_y)$, all of the fields (Φ, $A_\parallel$, and $\Delta B_\parallel$ and/or their gyro-averaged versions if already derived from un-gyro-averaged fields) should be shifted in k_x as well. In principle the fields could be rederived from f after shifting f, but for a well resolved k_x grid this should not be very different from just shifting the fields, and at least as a start it is easiest just to have a separate subroutine do all of these shifts every time step and not bother recalculating the fields.

The above equations are fairly straightforward conceptually, and can be put into a single subroutine that will be called at the beginning of every time step. A more complicated part of actually implementing these equations will be handling the parallelization issues: since f is split among different processors, it will require interprocessor communication to shift the data $f(k_x, k_y) = f(k_x - \texttt{jump}(k_y), k_y)$. This is doable but requires some extra work to write the code to handle the interprocessor communication. For simplicity, Dorland's first implementation of this in GS2 focussed on the case where the data layouts where such that all of the k_x's were on the same processor.

1.2 Comments on the algorithm

When $f(k_{x*}(t), k_y)$ gets sheared beyond the highest resolved mode, it is just lost (presumed to be lost to some high k damping process like collisional diffusion and/or Landau damping). Likewise, when $k_x - \texttt{jump}(k_y)$ lies beyond the maximum resolution of the k_x grid, then setting the corresponding $f(k_x) = 0$ just represents how some unresolved k_x modes can be unsheared to resolved scales, and initially appear at resolved scales with zero amplitude.

At first it might seem that this shearing will kill all of the turbulence, but there are two processes that can allow the turbulence to remain, at least in some cases (of course the shear is supposed to suppress the turbulence in some cases, if the shear is strong enough, etc.).

One process is that there can be nonlinear scattering that will put energy into the newly resolved k_x modes.

The other process works even linearly and makes use of the magnetic shear: a linear mode can form that moves along the sheared magnetic fields in such a way as to untwist the shear that the perpendicular sheared flows are trying to put in. (I've called it a "linear mode" here, but I mean a suitably-generalized version of the concept of a linear mode, since it no longer has a simple $e^{-i\omega t}$ time dependence because the coefficients of the equation vary in time. However the coefficients are periodic in time with period T_p in a frame moving along the field line at a certain velocity, so there is a generalized eigenmode concept $f(\theta, t + T_p) = f(\theta - N2\pi, t)e^{-i\omega T_p}$.)

In fact, some methods for dealing perpendicular sheared flows go to a frame of reference that is moving along the field lines (see previous papers by Waltz, Dewar, Connor, others?). Using field-aligned coordinates doesn't eliminate this process: as particles flow out of the flux-tube at one end, they come back into the flux-tube at the other end. The flux tube is twisted in opposite directions at the two ends by the magnetic shear, so the way the parallel boundary conditions are implemented in the presence of magnetic shear can undo the twist that the sheared flow is trying to put in.

Using the nearest discrete value of k_x is roughly similar to the Nearest Grid Point (NGP) approximation in PIC codes (but in Fourier space instead of real space). Studies have found that NGP is often quite adequate. Using the nearest discrete value of k_x instead of the exact $k_{x*}(t)$ isn't formally 2cd order accurate, but since we are uniformly sweeping over values of k_{x*} that are slightly smaller than and slightly larger than a particular k_x grid point, the errors from the two sides should cancel on average and give 2cd-order-like behavior. This is kind of like Godunov splitting, which although is formally only 1st order accurate for any individual time step, is just as good as a 2cd order accurate splitting algorithm (like Strang splitting or ADI) when applied over many time steps, since the errors from successive time steps cancel (see Leveque's book, Finite Volume Methods for Hyperbolic Problems, 2002) as long as the time step Δt isn't changed often.

Figure 2: Illustrate this by adding a figure plotting ω_d vs. k_x, and showing a discretized version of this function, with some dots showing $k_{x*}(t)$ on successive time steps to indicate that it averages over errors.

Convergence can always be checked by making the k_x grid finer (by making the simulation domain larger in the x direction, since $\Delta k_x = 2\pi/L_x$). In general one would expect this to be well converged if the time it takes to sweep over one k_x grid point, $T = \Delta k_x/(k_y \gamma_E)$, is short compared to the growth rate, so that the $\gamma \Delta k_x/(\gamma_E k_y) \ll 1$. Or parameterizing $\gamma = \gamma_0 k_y/k_{y0}$, (a linear scaling often works well at longer wavelengths), this means

$$\gamma_0 \Delta k_x/(\gamma_E k_0) \ll 1.$$

For typical parameters of interest this is well satisfied and so this simple algorithm should work well.

There is probably a way to modify this algorithm to make it formally second order accurate for each time step. Discrete Fourier transformed quantities like $f(k_x, k_y)$ are not smooth functions of k_x and so can't be interpolated to intermediate values of k_x. (In principle, one could do Fourier transforms on an extended x grid and use a windowing technique to cause the integrands to vanish at large x, but this would probably be fairly inefficient.) However, the various operators in the gyrokinetic equation, such as the grad(B) drift frequency or the Bessel-function gyro-averages, are smooth functions of k_x and so could be interpolated. I.e., one might be able to approximate linear operators of the form $L_{k*} f_{k_*} = (w_1 L_{k_1} f_{k*} + w_2 L_{k_2} f_{k*})/(w_1 + w_2)$, where w_1 and w_2 are the appropriate interpolation weights, $w_2 = k_* - k_1$, $w_1 = k_2 - k_*$. Linearly this can be implemented by calling the time advancement algorithm twice, once with k_* set to the closest discrete value k_1 less than k_* and once with k_* set to the closest discrete value k_2 greater than k_*, and then do the appropriately weighted averages of the two solutions for f obtained at the future time step. But again, the simple discretized k_{x*} algorithm should work well in most cases, and should be the first algorithm tried. The convergence of it can always be checked simply by making the k_x mesh finer.

2 Electromagnetic generalization

While it is clear how going to moving coordinates could eliminate the shearing term from the gyrokinetic equation, Eq.(5), in the electrostatic limit, it isn't necessarily clear at first whether this works in the more general electromagnetic case, because then there is also a term involving $\partial A_\parallel/\partial t$, and going to moving coordinates will complicate this term. However there is a corresponding spatial derivative term involving $A_\parallel$ that will cancel also.

To see this, start with the gyrokinetic equation in the slab drift-kinetic limit for simplicity:

$$\frac{\partial f}{\partial t} + v_\parallel \nabla_\parallel f + v_E \cdot \nabla f + \frac{q}{m}\left(-\nabla_\parallel \Phi - \frac{1}{c}\frac{\partial A_\parallel}{\partial t}\right)\frac{\partial f}{\partial v_\parallel} = 0$$

The $E \times B$ term can be written as $v_E \cdot \nabla f = (c/B)\{\Phi, f\}$. The parallel gradient of Φ involves gradients along perturbed magnetic fields lines, and so can be written as:

$$\nabla_\parallel \Phi = \frac{\vec{B}}{|\vec{B}|} \cdot \nabla \Phi \tag{15}$$

$$= \frac{1}{B}\left[\vec{B}_0 + \nabla \times \vec{A}\right] \cdot \nabla \Phi \tag{16}$$

$$= \frac{\partial \Phi}{\partial z} - \frac{1}{B}\hat{b}_0 \times \nabla A_\parallel \cdot \nabla \Phi \tag{17}$$

$$\tag{18}$$

Picking out just the terms in the drift-kinetic/gyrokinetic equation involving an equilibrium scale electric field $\partial\Phi_0/\partial x$ or time derivative terms, we are left with:

$$\frac{\partial f}{\partial t} + \frac{c}{B}\frac{\partial \Phi_0}{\partial x}\frac{\partial f}{\partial y} - \frac{q}{mc}\left(\frac{\partial A}{\partial t} + \frac{c}{B}\frac{\partial \Phi_0}{\partial x}\frac{\partial A}{\partial y}\right)\frac{\partial f}{\partial v_\parallel} = \text{usual terms ...}$$

Thus there is a common operator that appears twice:

$$\frac{\partial}{\partial t} + \frac{c}{B}\frac{\Phi_0}{\partial x}\frac{\partial}{\partial y}$$

and in both cases the terms involving the equilibrium-scale sheared flows from Φ_0 will be transformed away by going to the moving coordinate system.