`stella` Update: Full-Flux Annulus

G. O. Acton[1,2,3], M. Barnes[1], S. Newton[2],
F. I. Parra[5], W. Dorland[6], H. Thienpondt[4]

[1] Rudolf Peierls Centre For Theoretical Physics, University of Oxford, Oxford, OX1 3PU, UK

[2] Culham Centre for Fusion Energy, United Kingdom Atomic Energy Authority,
Abingdon, OX14 3EB, UK

[3] Max-Planck-Institut für Plasmaphysik, Teilinstitut Greifswald, Germany

[4] Laboratorio Nacional de Fusion, CIEMAT, 28040 Madrid, Spain

[5] Princeton Plasma Physics Laboratory, Princeton, NJ 08540, USA

[6] Department of Physics, University of Maryland, College Park, MD 20742, United States of America

Monday 13th April 2026
TSVV-J meeting

Table of Contents

1. Introduction to `stella`

2. Motivation for Full-Flux Annulus

3. Flux Annulus Formalism

4. Results

Table of Contents

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations

`stella`: Introduction

- **`stella`** is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently

$$\frac{\partial g_\nu}{\partial t} + \underbrace{\underbrace{\mathcal{S}_\nu[g_\nu,\varphi_\nu]}_{\text{streaming}} + \underbrace{\mathcal{M}_\nu[g_\nu]}_{\text{mirror}}}_{\text{Implicit}} + \underbrace{\underbrace{\mathcal{D}_\nu[g_\nu,\varphi_\nu] + \mathcal{G}_\nu[\varphi_\nu]}_{\text{drifts}} + \underbrace{\mathcal{N}_\nu[g_\nu,\varphi_\nu]}_{\text{non-linear}} = \underbrace{\mathcal{C}_\nu[\{g_{\nu'}\},\{\varphi_{\nu'}\}]}_{\text{collisions}}}_{\text{Explicit}},$$

$$\partial_t g_\nu = \textstyle\sum_{i=1}^{3}(\partial_t g_\nu)_i \quad \left| \quad \begin{aligned} (\partial_t g_\nu)_1 + \mathcal{D}_\nu[g_\nu,\varphi_\nu] + \mathcal{G}_\nu[\varphi_\nu] + \mathcal{N}_\nu[g_\nu,\varphi_\nu] &= \mathcal{C}_\nu \\ (\partial_t g_\nu)_2 + \mathcal{M}_\nu[g_\nu] &= 0 \\ (\partial_t g_\nu)_3 + \mathcal{S}_\nu[g_\nu,\varphi_\nu] &= 0 \end{aligned} \right.$$

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly

$$\mathcal{S}_\nu \doteq \quad v_{\rm th} \quad v_\parallel \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z \left(\frac{\partial g_\nu}{\partial z} + \frac{Z_\nu}{T_\nu} \frac{\partial \varphi_\nu}{\partial z} F_{0,\nu} \right)$$

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly

$$\mathcal{S}_\nu \doteq \underbrace{v_{\mathrm{th}}}_{\sim\sqrt{m_i/m_e}} v_\parallel \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z \left(\frac{\partial g_\nu}{\partial z} + \frac{Z_\nu}{T_\nu} \frac{\partial \varphi_\nu}{\partial z} F_{0,\nu} \right)$$

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly

$$\frac{g_\nu^{n+1} - g_\nu^{n}}{\Delta t} = -v_{\mathrm{th}} v_{\|} \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z \left(\frac{\partial g_\nu^{n}}{\partial z} + \frac{Z_\nu e}{T_\nu} \frac{\partial \varphi_\nu^{n}}{\partial z} F_{0,\nu} \right) \tag{1}$$

`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly

$$\frac{g_\nu^{n+1} - g_\nu^{n}}{\Delta t} = -v_{\mathrm{th}} v_{\|} \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z \left(\frac{\partial g_\nu^{n+1}}{\partial z} + \frac{Z_\nu e}{T_\nu} \frac{\partial \varphi_\nu^{n+1}}{\partial z} F_{0,\nu} \right) \tag{1}$$

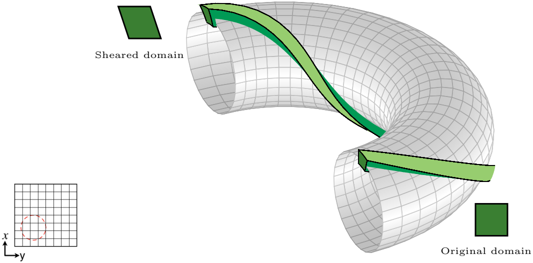
`stella`: Introduction

- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly
- Simulates turbulence on gyroradius scales. Solves for $g = \langle \delta f \rangle_{\mathbf{R}_\nu}$

`stella`: Introduction

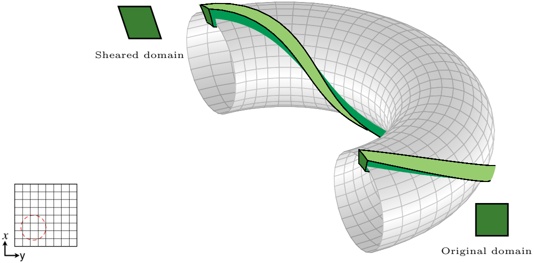
- `stella` is an operator-split, implicit-explicit δf gyrokinetic code for general magnetic field configurations
- Code name
- Each term in the gyrokinetic (GK) equation is solved independently
- Electron dynamics imposes stringent CFL condition on time step → treat parallel streaming and mirror terms implicitly
- Simulates turbulence on gyroradius scales. Solves for $g = \langle \delta f \rangle_{\mathbf{R}_\nu}$
- Uses $\{v_\parallel, \mu\}$ for the velocity space coordinates → good for stellarators

`stella`: Introduction



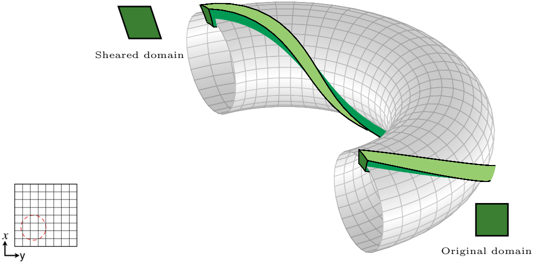
- Simulation coordinates: $(x, y, z) \rightarrow (\psi, \alpha, \zeta)$

`stella`: Introduction



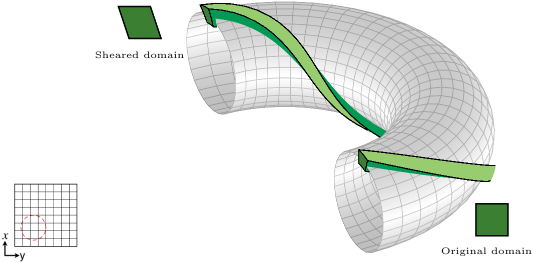
- Simulation coordinates: $(x, y, z) \rightarrow (\psi, \alpha, \zeta)$
- Initialise some δf and ϕ at $t = 0$ on simulation domain

`stella`: Introduction



- Simulation coordinates: $(x, y, z) \rightarrow (\psi, \alpha, \zeta)$
- Initialise some δf and ϕ at $t = 0$ on simulation domain
- Evolve gyrokinetic equation pseudo-spectrally

`stella`: Introduction



- Simulation coordinates: $(x, y, z) \to (\psi, \alpha, \zeta)$
- Initialise some δf and ϕ at $t = 0$ on simulation domain
- Evolve gyrokinetic equation pseudo-spectrally
  - Decay in $v_\parallel$; $g(t, \boldsymbol{x}, v_\parallel \to \pm\infty, \mu) \to 0$
  - Turbulence is taken as periodic in perpendicular directions, $k_x, k_y \gg 1/L$
  - Use twist-and-shift boundary conditions in z to capture extended modes

Table of Contents

Flux tube formalism

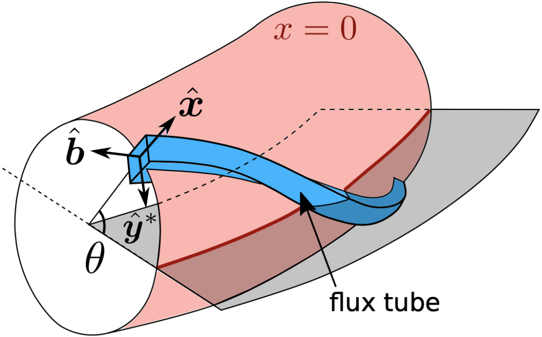


Image: Nicolas Christen, Bistable turbulent transport in fusion plasmas with rotational shear (2021)

- Flux-tube simulations are sufficient for tokamaks because we can stitch our flux tubes together

Flux tube formalism

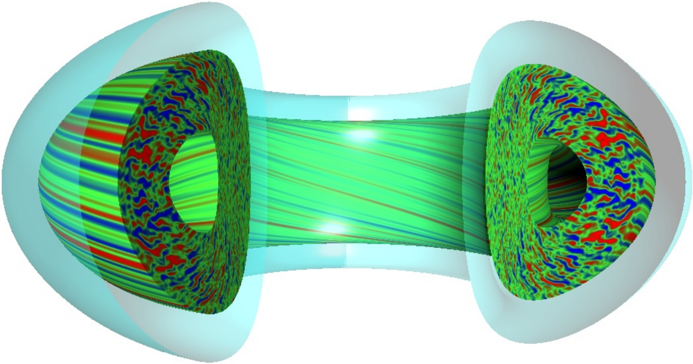


Image: J.Candy, Waltz, GYRO simulation of DIII-D

- Flux-tube simulations are sufficient for tokamaks because we can stitch our flux tubes together

Flux tube formalism: pros and cons

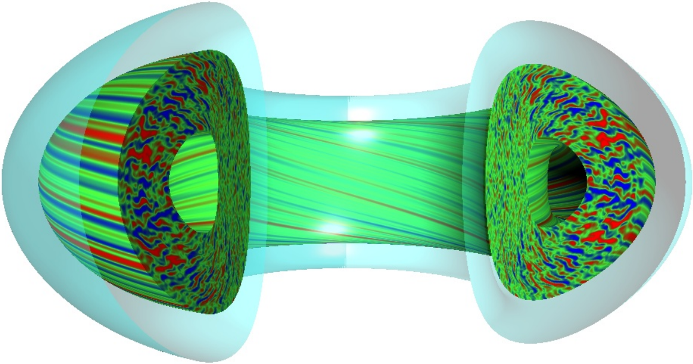


Image: J.Candy, Waltz, GYRO simulation of DIII-D

- Fast codes: perform many simulations in quick succession

Flux tube formalism: pros and cons

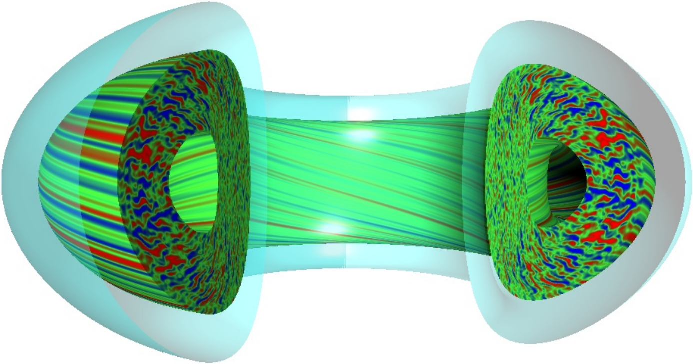


Image: J.Candy, Waltz, GYRO simulation of DIII-D

- Fast codes: perform many simulations in quick succession
- Easy to interpret: normal modes are well defined

Flux tube formalism: pros and cons

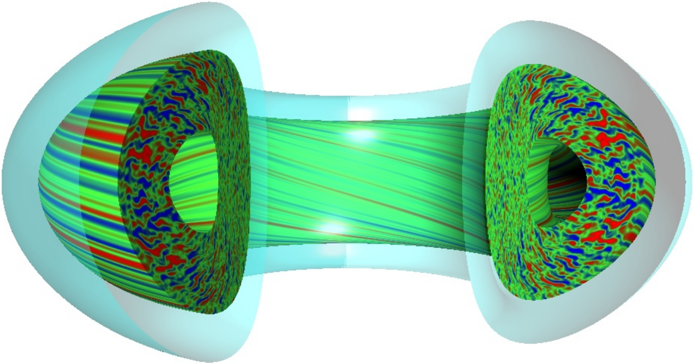
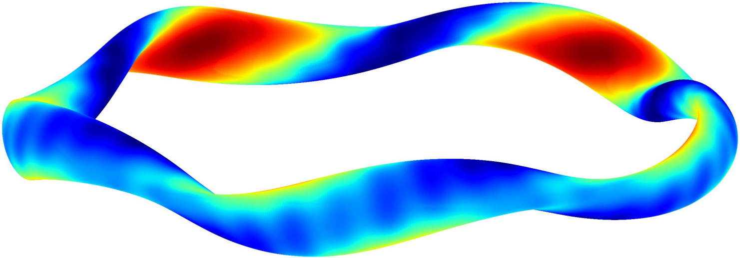


Image: J.Candy, Waltz, GYRO simulation of DIII-D

- Fast codes: perform many simulations in quick succession
- Easy to interpret: normal modes are well defined
- Retains spectral accuracy in spacial derivatives

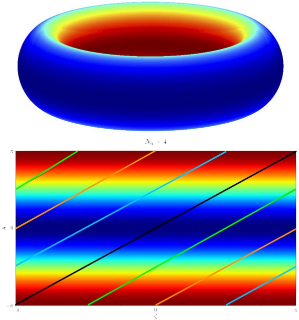
Flux tube formalism: pros and cons



- Fast codes: perform many simulations in quick succession
- Easy to interpret: normal modes are well defined
- Retains spectral accuracy in spacial derivatives
- Does not capture global effects like coupling between different field lines

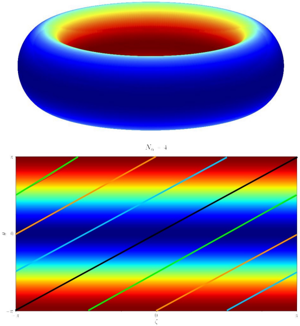
Motivation

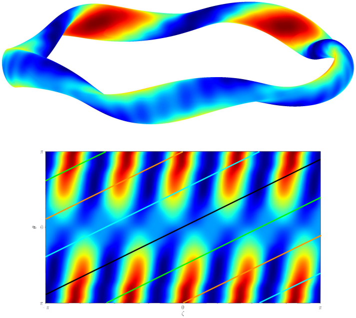
- Axisymmetry means all field lines are equivalent



Motivation

- Axisymmetry means all field lines are equivalent
- Stellarator magnetic geometry varies with field line: single field line is **not** sufficient!





Motivation

- Axisymmetry means all field lines are equivalent
- Stellarator magnetic geometry varies with field line: single field line is **not** sufficient!
- Modes on different field lines interact → complicates algorithms due to α-inhomogeneity

Motivation

- Axisymmetry means all field lines are equivalent
- Stellarator magnetic geometry varies with field line: single field line is **not** sufficient!
- Modes on different field lines interact → complicates algorithms due to α-inhomogeneity
- Particularly relevant for zonal flows

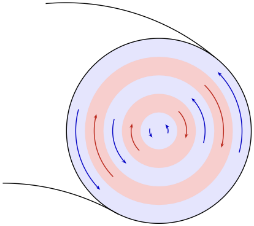


Table of Contents

Flux annulus formalism: motivation

- Stellarator geometry varies with field line
- Method of stitching together flux tubes no longer holds

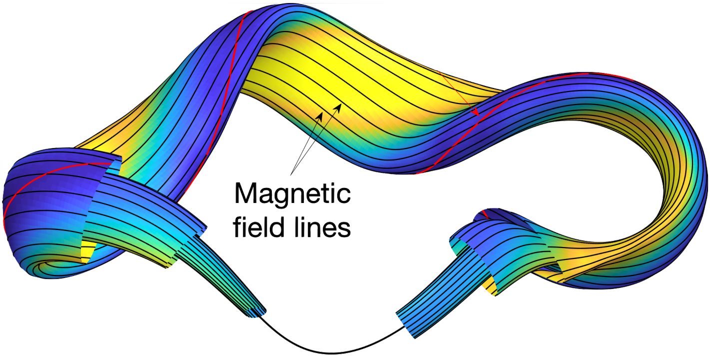
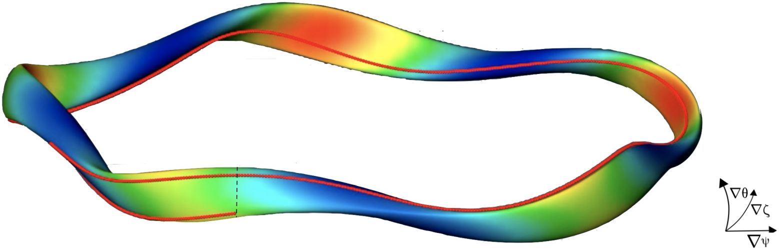


Image: UMD stellarator group

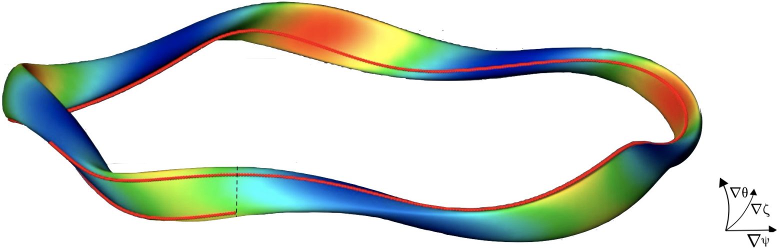
Flux annulus framework

- Domain is now 2π in ζ, not 2π in θ



Flux annulus framework

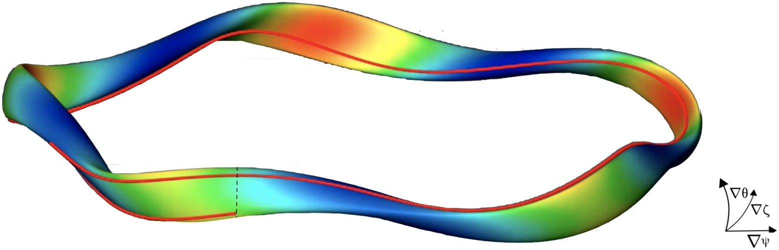
- Domain is now 2π in ζ, not 2π in θ



- Evolve pseudo-spectrally to retain spectral accuracy in derivatives

Flux annulus framework

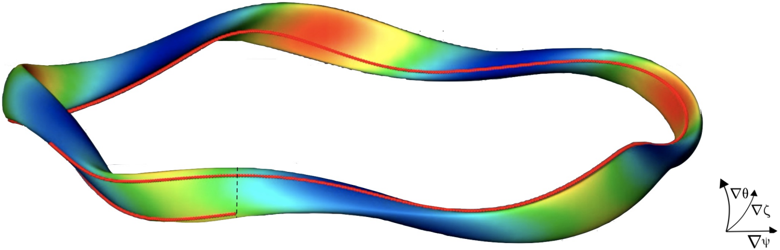
- Domain is now 2π in ζ, not 2π in θ



- Evolve pseudo-spectrally to retain spectral accuracy in derivatives
- N_y field lines that cover different geometry and are now coupled together

Flux annulus framework

- Domain is now 2π in ζ, not 2π in θ



- Evolve pseudo-spectrally to retain spectral accuracy in derivatives
- N_y field lines that cover different geometry and are now coupled together
- Apply twist-and-shift to entire poloidal domain

How 3D geometry modifies equations

- Geometry is no longer trivial
- But how does geometry enter our code?

How 3D geometry modifies equations

- Geometry is no longer trivial
- But how does geometry enter our code?

Simplified notation GK:

$$\frac{\partial g}{\partial t} = (\text{geometric factors}) \cdot (\boldsymbol{\nabla} g + \boldsymbol{\nabla} \left\langle \phi \right\rangle_{\boldsymbol{R}})$$

How 3D geometry modifies equations

- Geometry is no longer trivial
- But how does geometry enter our code?

Simplified notation GK:

$$\frac{\partial g}{\partial t} = \underbrace{(\text{geometric factors})}_{\text{e.g } \hat{\boldsymbol{b}}\cdot\boldsymbol{\nabla} z} \cdot \left(\boldsymbol{\nabla} g + \boldsymbol{\nabla} \underbrace{\langle \phi \rangle_{\boldsymbol{R}}}_{J_{0,\boldsymbol{k}}\hat{\phi}_{\boldsymbol{k}}}\right)$$

- Geometric factors are α-dependent
- Bessel functions $J_0(a_{\boldsymbol{k}})$ with $a_{\boldsymbol{k}} = \frac{k_\perp v_\perp}{\Omega_s}$

How 3D geometry modifies equations

- Geometry is no longer trivial
- But how does geometry enter our code?

Simplified notation GK:

$$\frac{\partial g}{\partial t} = \underbrace{(\text{geometric factors})}_{\text{e.g } \hat{\boldsymbol{b}}\cdot\boldsymbol{\nabla} z} \cdot \left(\boldsymbol{\nabla} g + \boldsymbol{\nabla} \underbrace{\langle\phi\rangle_{\boldsymbol{R}}}_{J_{0,\boldsymbol{k}}\hat{\phi}_{\boldsymbol{k}}}\right)$$

- Geometric factors are α-dependent
- Bessel functions $J_0(a_{\boldsymbol{k}})$ with $a_{\boldsymbol{k}} = \frac{k_\perp v_\perp}{\Omega_s}$
- α-inhomogeneity leads to convolutions

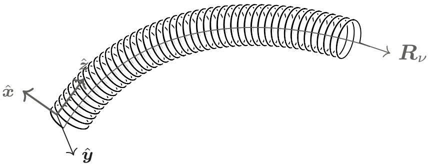
How 3D geometry modifies equations

- Geometry is no longer trivial
- But how does geometry enter our code?

Simplified notation GK:

$$\frac{\partial g}{\partial t} = \underbrace{\text{(geometric factors)}}_{\text{e.g } \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z} \cdot \left(\boldsymbol{\nabla} g + \boldsymbol{\nabla} \underbrace{\langle \phi \rangle_{\boldsymbol{R}}}_{J_{0,\boldsymbol{k}} \hat{\phi}_{\boldsymbol{k}}} \right)$$

- Geometric factors are α-dependent
- Bessel functions $J_0(a_{\boldsymbol{k}})$ with $a_{\boldsymbol{k}} = \frac{k_\perp v_\perp}{\Omega_s}$ ← $k_\perp$ and B in argument
- Gyro-averaging introduces coupling between different k_y-modes → no longer a local operation



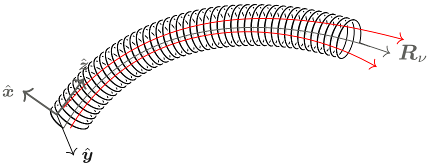
How 3D geometry modifies equations

- Geometry is no longer trivial
- But how does geometry enter our code?

Simplified notation GK:

$$\frac{\partial g}{\partial t} = \underbrace{(\text{geometric factors})}_{\text{e.g } \hat{\boldsymbol{b}}\cdot\boldsymbol{\nabla} z} \cdot \left(\boldsymbol{\nabla} g + \boldsymbol{\nabla} \underbrace{\langle\phi\rangle_{\boldsymbol{R}}}_{J_{0,\boldsymbol{k}}\hat{\phi}_{\boldsymbol{k}}}\right)$$

- Geometric factors are α-dependent
- Bessel functions $J_0(a_{\boldsymbol{k}})$ with $a_{\boldsymbol{k}} = \frac{k_\perp v_\perp}{\Omega_s}$ ← $k_\perp$ and B in argument
- Gyro-averaging introduces coupling between different k_y-modes → no longer a local operation



How 3D geometry modifies equations

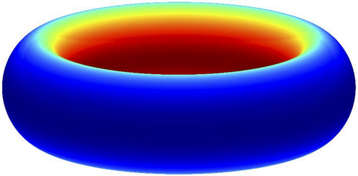
- Geometry is no longer trivial
- But how does geometry enter our code?

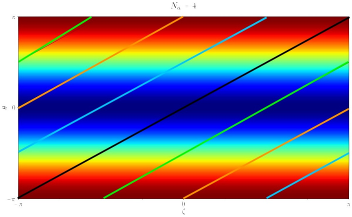
Simplified notation GK:

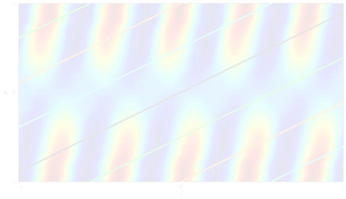
$$\frac{\partial g}{\partial t} = (\text{geometric factors}) \cdot \left(\boldsymbol{\nabla} g + \boldsymbol{\nabla} \left\langle \phi \right\rangle_{\boldsymbol{R}} \right)$$

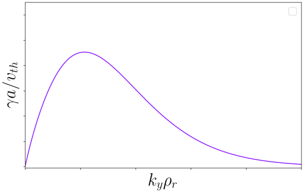
- Geometric factors are α-dependent
- Bessel functions $J_0(a_{\boldsymbol{k}})$ with $a_{\boldsymbol{k}} = \frac{k_\perp v_\perp}{\Omega_s} \leftarrow k_\perp$ and B in argument
- Gyro-averaging introduces coupling between different k_y-modes → no longer a local operation
- Develop iterative-implicit method → accurate treatment without stringent CFL restriction

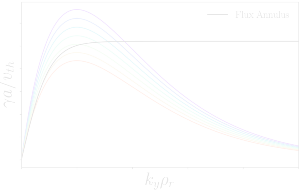
How 3D geometry modifies expectations



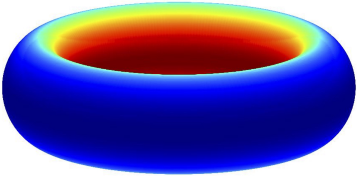


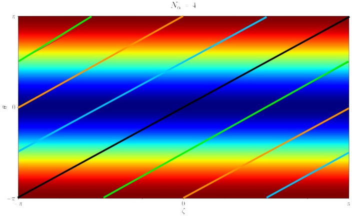


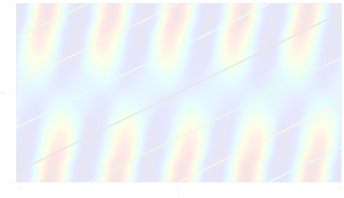


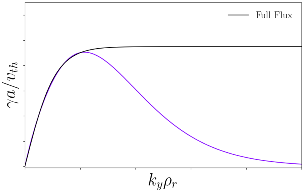


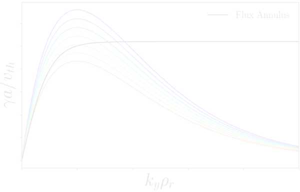
How 3D geometry modifies expectations



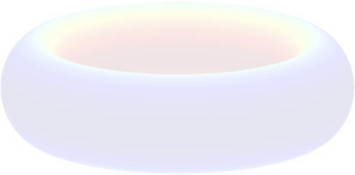


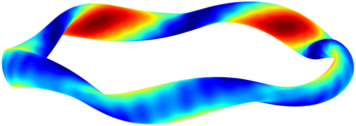


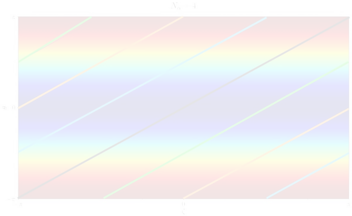


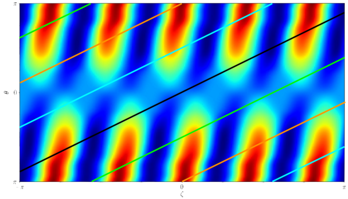


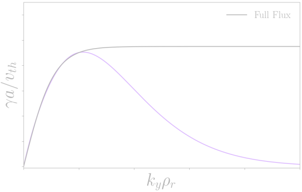
How 3D geometry modifies expectations











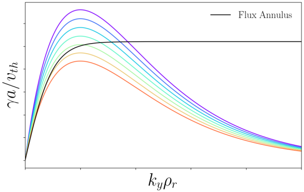


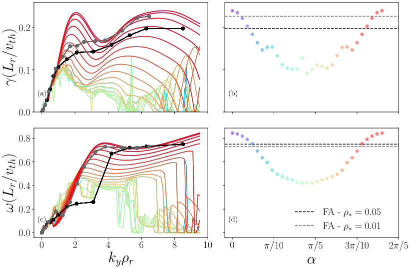
Table of Contents

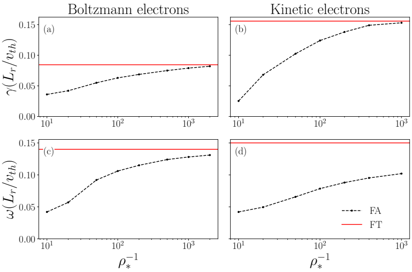
Benchmarked

- Simulates gyrokinetic turbulence on a full-flux annulus in general 3D geometry
- Capable of simulating adiabatic and kinetic electrons

Benchmarked

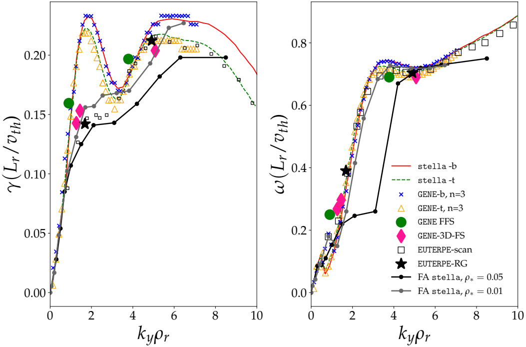
- Simulates gyrokinetic turbulence on a full-flux annulus in general 3D geometry
- Capable of simulating adiabatic and kinetic electrons
- Has been benchmarked against stella flux tube code in appropriate limits





Benchmarked

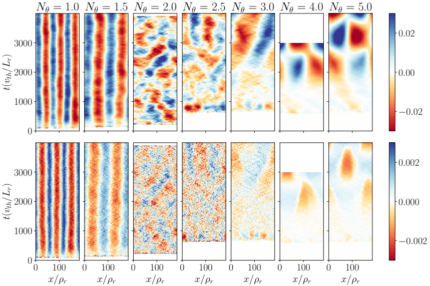
- Simulates gyrokinetic turbulence on a full-flux annulus in general 3D geometry
- Capable of simulating adiabatic and kinetic electrons
- Has been benchmarked against stella flux tube code in appropriate limits
- Shows good agreement with other full-surface codes

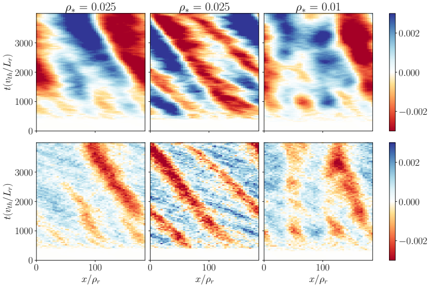


E. Sánchez, J. M. García-Regaña, A. B. Navarro, J. H. E. Proll, C. M. Moreno, A. González-Jerez, I. Calvo, R. Kleiber, J. Riemann, J. Smoniewski, et al.
Gyrokinetic simulations in stellarators using different computational domains.
Nuclear Fusion, 61(11):116074, 2021.

Benchmarked

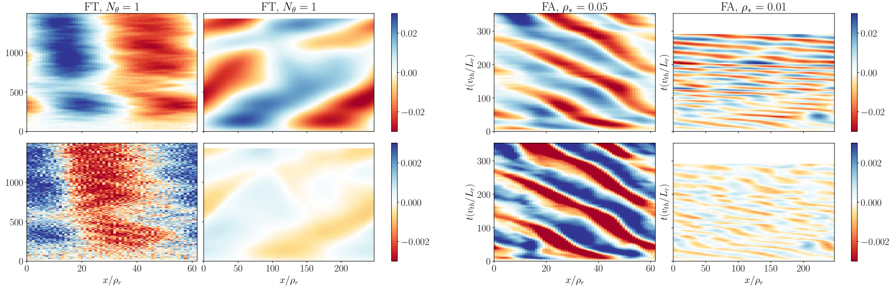
- Simulates gyrokinetic turbulence on a full-flux annulus in general 3D geometry
- Capable of simulating adiabatic and kinetic electrons
- Has been benchmarked against stella flux tube code in appropriate limits
- Shows good agreement with other full-surface codes
- Preliminary results shows differences between flux-tube and full-flux approaches with Boltzmann electrons





Benchmarked

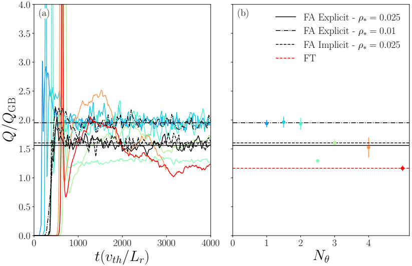
- Simulates gyrokinetic turbulence on a full-flux annulus in general 3D geometry
- Capable of simulating adiabatic and kinetic electrons
- Has been benchmarked against stella flux tube code in appropriate limits
- Shows good agreement with other full-surface codes
- Preliminary results shows differences between flux-tube and full-flux approaches with Boltzmann electrons and kinetic electrons

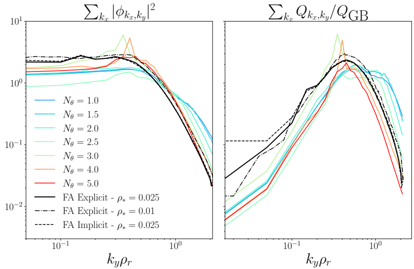


Simulation results: code efficiency

- Boltzmann electrons:

Simulation Case	Time Step (dt)	CPU Hours
FT, $N_\theta = 1$	0.049666	26,880
FT, $N_\theta = 5$	0.030470	80,640
FA (explicit), $\rho_* = 0.025$	0.1	43,550
FA (implicit), $\rho_* = 0.025$	0.1	51,690
FA (explicit)[1], $\rho_* = 0.01$	0.1	35,420





---

[1] Reduced resolution

Simulation results: code efficiency

- Kinetic electrons:

Parameter	FT_1	FT_2	FA_1	FA_2
Δt	1.6577×10^{-2}	2.1195×10^{-2}	7.7464×10^{-3}	5.6368×10^{-3}
t_{final}	1500	1470	355	270
CPU hours	46,080	76,800	107,520	128,640

- $\Delta t =$ `1E-006` for explicit kinetic electron simulations → numerically unstable
- Implicit treatment allows $\Delta t \sim$ `5E-003`

Simulation results: code efficiency

- Kinetic electrons:

Parameter	FT_1	FT_2	FA_1	FA_2
Δt	1.6577×10^{-2}	2.1195×10^{-2}	7.7464×10^{-3}	5.6368×10^{-3}
t_{final}	1500	1470	355	270
CPU hours	46,080	76,800	107,520	128,640

- $\Delta t =$ `1E-006` for explicit kinetic electron simulations → numerically unstable
- Implicit treatment allows $\Delta t \sim$ `5E-003`
- Currently $\sim \times 20$ slower than flux tube simulations with kinetic electrons
- Nonlinear simulations with kinetic electrons and geometric variation: 24 hours, 5 nodes, 640 cores, computes ~ 70 thermal times

Simulation results: code efficiency

- Kinetic electrons:

Parameter	FT_1	FT_2	FA_1	FA_2
Δt	1.6577×10^{-2}	2.1195×10^{-2}	7.7464×10^{-3}	5.6368×10^{-3}
t_{final}	1500	1470	355	270
CPU hours	46,080	76,800	107,520	128,640

- $\Delta t =$ `1E-006` for explicit kinetic electron simulations → numerically unstable
- Implicit treatment allows $\Delta t \sim$ `5E-003`
- Currently $\sim \times 20$ slower than flux tube simulations with kinetic electrons
- Nonlinear simulations with kinetic electrons and geometric variation: 24 hours, 5 nodes, 640 cores, computes ~ 70 thermal times

Improvements

- NB: this was before `stella` improvements → need to re-run with updated version
- Already identified inefficiencies
- Anticipate this can be reduced to $\sim \times 5$ speed of flux-tube
- Removing strong coupling (e.g. for CBC) improves code by ~ 10 times for KE

Summary and future work

Summary

- Developed an iterative-implicit algorithm to deal with full-flux annulus effects, with kinetic electrons, in arbitrary geometry
- FFA `stella` has been benchmarked in appropriate limits and against existing codes

Summary and future work

Summary

- Developed an iterative-implicit algorithm to deal with full-flux annulus effects, with kinetic electrons, in arbitrary geometry
- FFA `stella` has been benchmarked in appropriate limits and against existing codes

Future Work

- Investigate how zonal flows are supported in stellarators
- Combine full-flux with electromagnetic `stella`
- Optimise for computational speed-up

Backup slides: Derivation of twist-and-shift boundary conditions

*

$$A(t,x,y,z) = \sum_k \hat{A}_{k_x,k_y}(t,z)e^{ik_y(y-y_0)+ik_x(x-x_0)} \tag{1}$$

Set $y_0 = 0$, $x_0 = 0$ $A(t,x,y(x,\theta,z),z) = A(t,x,y'(\theta,z+2p\pi),z+2p\pi)$
But $y = y(\theta,z)$

$$\sum_k \hat{A}_{k_x,k_y}(t,z)e^{ik_y y+ik_x x} = \sum_k \hat{A}_{k_x,k_y}(t,z')e^{ik_y(y'(\theta,z'))+ik_x x} \tag{2}$$

So $y'(\theta,z') = y + \frac{\partial y}{\partial z}2\pi p = y + 2\pi p\frac{\partial y}{\partial \alpha}\frac{\partial \alpha}{\partial z} = y - 2\pi p\iota(\psi)\frac{\partial y}{\partial \alpha}$ Remembering $\iota(\psi) = \iota(\psi_0) + \iota'\frac{\partial \psi}{\partial x}$ so $y' = y - 2\pi p\iota(\psi_0)\frac{\partial y}{\partial \alpha} - 2(x-x_0)\pi p\iota'\frac{\partial y}{\partial \alpha}\frac{\partial \psi}{\partial x}$

$$\sum_k \hat{A}_{k_x,k_y}(t,z)e^{ik_y y+ik_x x} = \sum_k \hat{A}_{k_x,k_y}(t,z{+}2\pi p)e^{ik_y y+i(k_x-2\pi p\iota'\frac{\partial y}{\partial \alpha}\frac{\partial \psi}{\partial x}k_y)x'-i2\pi pk_y\iota p\frac{\partial y}{\partial \alpha}} \tag{3}$$

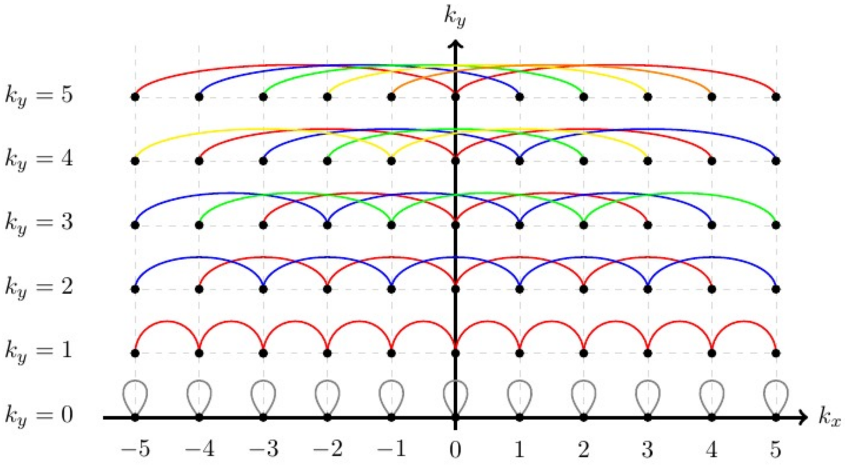
Let $\delta k_x = 2\pi p\iota'\frac{\partial y}{\partial \alpha}\frac{\partial \psi}{\partial x}k_y$ and $\Delta = -2\pi pk_y\iota\frac{\partial y}{\partial \alpha}$

$$\sum_k \hat{A}_{k_x,k_y}(t,z)e^{ik_y y+ik_x x} = \sum_k \hat{A}_{k_x,k_y}(t,z+2\pi p)e^{ik_y y+i(k_x-\delta k_x)x'}e^{i\Delta} \tag{4}$$

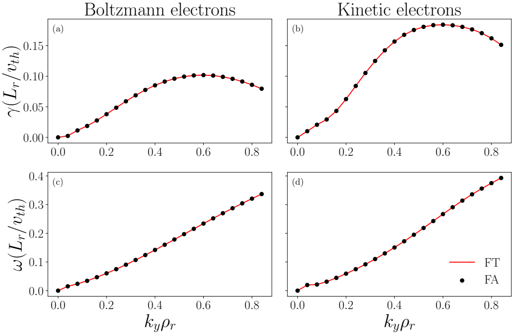
So relate $k_x = k_x' - 2\pi p\iota'\frac{\partial y}{\partial \alpha}\frac{\partial \psi}{\partial x}k_y$

Backup slides: Eigenmode chains

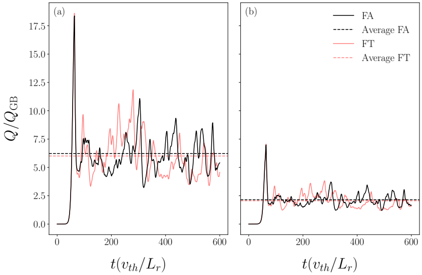
*

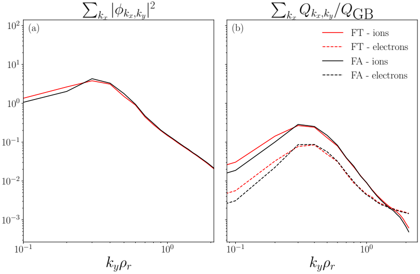


Backup slides: Miller geometry linear comparison



Backup slides: Miller geometry nonlinear comparison





Backup slides: Adiabatic, W7X comparison between GX and stella