Adjoint method for gyrokinetic optimisation

G. Acton[1,2], M. Barnes[1], S. Newton[1]

[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

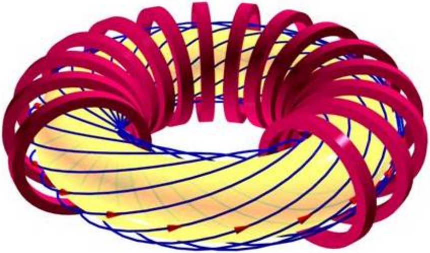
TSVV, 10/10/2022

Table of Contents

Table of Contents

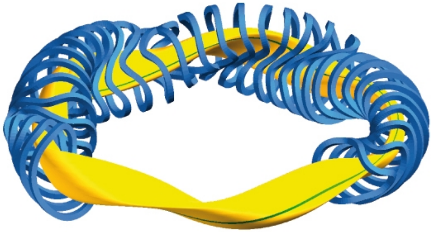
Motivation

- Linear microinstabilities → turbulence → produces stiff transport
- Desirable to have high temperature in the core, requiring a large temperature gradient → maximise temperature gradient whilst maintaining microstability
- Magnetic confinement fusion (MCF) devices are complicated, and the linear growth rate depends on a large number of parameters
- High-dimensionality of parameter space makes scans computationally expensive



Motivation

- Linear microinstabilities → turbulence → produces stiff transport
- Desirable to have high temperature in the core, requiring a large temperature gradient → maximise temperature gradient whilst maintaining microstability
- Magnetic confinement fusion (MCF) devices are complicated, and the linear growth rate depends on a large number of parameters
- High-dimensionality of parameter space makes scans computationally expensive



Framework of project

- Develop general adjoint model for gyrokinetics
- Implement into `stella`, perturbing magnetic geometry
- Can we increase temperature gradient?

Table of Contents

System set-up: introduction example

- System objective function:

$$\hat{L}[\boldsymbol{p}; f(\boldsymbol{p}, \boldsymbol{s})] = 0 \tag{1}$$

  where $\boldsymbol{p}$ = set of parameters, and f = function that depends on $\boldsymbol{p}$ (e.g. distribution function)

System set-up: introduction example

- System objective function:
$$\hat{L}[\boldsymbol{p}; f(\boldsymbol{p}, \boldsymbol{s})] = 0 \quad (1)$$
where $\boldsymbol{p}$ = set of parameters, and f = function that depends on $\boldsymbol{p}$ (e.g. distribution function)

- Want to optimise function $\hat{H} = \hat{H}[\boldsymbol{p}; f]$ with respect to $\{p_i\}$
$$\hat{H}[\boldsymbol{p}; f] = \underbrace{\langle \hat{h}[\boldsymbol{p}; f(\boldsymbol{p})], f \rangle}_{\text{inner product of } \hat{h} \text{ with } f} \quad (2)$$

System set-up: introduction example

- System objective function:

$$\hat{L}[\boldsymbol{p}; f(\boldsymbol{p}, \boldsymbol{s})] = 0 \tag{1}$$

where $\boldsymbol{p}$ = set of parameters, and f = function that depends on $\boldsymbol{p}$ (e.g. distribution function)

- Want to optimise function $\hat{H} = \hat{H}[\boldsymbol{p}; f]$ with respect to $\{p_i\}$

$$\hat{H}[\boldsymbol{p}; f] = \underbrace{\langle \hat{h}[\boldsymbol{p}; f(\boldsymbol{p})], f \rangle}_{\text{inner product of } \hat{h} \text{ with } f} \tag{2}$$

- Could use a finite difference method

$$\frac{\partial \hat{H}}{\partial p_i} = \frac{\hat{H}[p_i + \delta p_i; f(p_i + \delta p_i)] - \hat{H}[p_i; f(p_i)]}{\delta p_i} \tag{3}$$

but this is expensive when parameter space is large

System set-up: introduction example

- System objective function:
$$\hat{L}[\boldsymbol{p}; f(\boldsymbol{p}, \boldsymbol{s})] = 0 \tag{1}$$
where $\boldsymbol{p}$ = set of parameters, and f = function that depends on $\boldsymbol{p}$ (e.g. distribution function)

- Want to optimise function $\hat{H} = \hat{H}[\boldsymbol{p}; f]$ with respect to $\{p_i\}$
$$\hat{H}[\boldsymbol{p}; f] = \underbrace{\langle \hat{h}[\boldsymbol{p}; f(\boldsymbol{p})], f \rangle}_{\text{inner product of } \hat{h} \text{ with } f} \tag{2}$$

- Could use a finite difference method
$$\frac{\partial \hat{H}}{\partial p_i} = \frac{\hat{H}[p_i + \delta p_i; f(p_i + \delta p_i)] - \hat{H}[p_i; f(p_i)]}{\delta p_i} \tag{3}$$
but this is expensive when parameter space is large

- Alternatively use an adjoint method approach - Computation is independent of dimension of the parameter space.

Adjoint equations: introduction example

- Define an optimisation Lagrangian

$$\mathcal{L}[\boldsymbol{p}; f, \lambda] = \hat{H}[\boldsymbol{p}; f(\boldsymbol{p})] + \left\langle \hat{L}[\boldsymbol{p}; f(\boldsymbol{p})], \lambda \right\rangle \tag{4}$$

Recall $\hat{L}[\boldsymbol{p}; f(\boldsymbol{p})] = 0$

Adjoint equations: introduction example

- Define an optimisation Lagrangian

$$\mathcal{L}[\boldsymbol{p}; f, \lambda] = \hat{H}[\boldsymbol{p}; f(\boldsymbol{p})] + \left\langle \hat{L}[\boldsymbol{p}; f(\boldsymbol{p})], \lambda \right\rangle \tag{4}$$

  Recall $\hat{L}[\boldsymbol{p}; f(\boldsymbol{p})] = 0$

- For brevity we will consider the 1-D case for the derivation of the adjoint equations, then generalise to a multi-dimensional parameter space

$$\boldsymbol{p} \to p \qquad \boldsymbol{\nabla}_{\boldsymbol{p}} \to \mathrm{d}_p \tag{5}$$

Adjoint equations: introduction example

- Define an optimisation Lagrangian

$$\mathcal{L}[\boldsymbol{p}; f, \lambda] = \hat{H}[\boldsymbol{p}; f(\boldsymbol{p})] + \left\langle \hat{L}[\boldsymbol{p}; f(\boldsymbol{p})], \lambda \right\rangle \tag{4}$$

Recall $\hat{L}[\boldsymbol{p}; f(\boldsymbol{p})] = 0$

- For brevity we will consider the 1-D case for the derivation of the adjoint equations, then generalise to a multi-dimensional parameter space

$$\boldsymbol{p} \to p \qquad \boldsymbol{\nabla}_{\boldsymbol{p}} \to \mathrm{d}_p \tag{5}$$

- Take derivative of (4) with respect to p

$$\mathrm{d}_p\mathcal{L}[p; f, \lambda] = \mathrm{d}_p\hat{H} + \left\langle \mathrm{d}_p\hat{L}, \lambda \right\rangle + \left\langle \hat{L}, \mathrm{d}_p\lambda \right\rangle + \underbrace{\partial_{\mathcal{J}} \left\langle (\mathrm{d}_p\mathcal{J})\, \hat{L}, \lambda \right\rangle}_{\text{Takes into account } p\text{-dependence in Jacobian}} \tag{6}$$

with

$$\mathrm{d}_p\hat{H} = \left\langle \partial_p\hat{h}[p; f], f \right\rangle + \left\langle \hat{h}[p; d_p f], f \right\rangle + \left\langle \hat{h}[p; f], d_p f \right\rangle + \partial_{\mathcal{J}} \left\langle (\mathrm{d}_p\mathcal{J})\hat{h}, \lambda \right\rangle \tag{7}$$

Adjoint equations: introduction example

- Derivative $d_p f$ are computationally expensive

- Invert the operators $\hat{h}$ and $\hat{L}$ wherever they act on $\mathrm{d}_p f$, and collect coefficients of $\mathrm{d}_p f$ terms

Adjoint equations: introduction example

- Derivative $d_p f$ are computationally expensive

- Invert the operators $\hat{h}$ and $\hat{L}$ wherever they act on $\mathrm{d}_p f$, and collect coefficients of $\mathrm{d}_p f$ terms

- Resulting equation is:

$$\begin{aligned} \mathrm{d}_p \mathcal{L}[p; f, \lambda]\big|_{f,\lambda} &= \left\langle \partial_p \hat{h}[p; f], f \right\rangle + \left\langle \partial_p \hat{L}[p; f], \lambda \right\rangle \Big|_{f,\lambda} \\ &+ \underbrace{\left\langle \hat{h}^\dagger[p; f] + \hat{h}[p; f] + \hat{L}^\dagger[p; \lambda], \mathrm{d}_p f \right\rangle \Big|_{f,\lambda}}_{\substack{\text{computationally expensive to calculate} \\ \text{so set to zero}}} \end{aligned} \tag{8}$$

Adjoint equations: introduction example

- Derivative $d_p f$ are computationally expensive
- Invert the operators $\hat{h}$ and $\hat{L}$ wherever they act on $\mathrm{d}_p f$, and collect coefficients of $\mathrm{d}_p f$ terms
- Resulting equation is:

$$\mathrm{d}_p \mathcal{L}[p; f, \lambda]\big|_{f,\lambda} = \left\langle \partial_p \hat{h}[p; f], f \right\rangle + \left\langle \partial_p \hat{L}[p; f], \lambda \right\rangle\Big|_{f,\lambda} + \left\langle \hat{h}^\dagger[p; f] + \hat{h}[p; f] + \hat{L}^\dagger[p; \lambda], \mathrm{d}_p f \right\rangle\Big|_{f,\lambda} \tag{8}$$

Adjoint equations: introduction example

- Derivative $d_p f$ are computationally expensive
- Invert the operators $\hat{h}$ and $\hat{L}$ wherever they act on $\mathrm{d}_p f$, and collect coefficients of $\mathrm{d}_p f$ terms
- Resulting equation is:

$$\mathrm{d}_p \mathcal{L}[p; f, \lambda]\big|_{f,\lambda} = \left\langle \partial_p \hat{h}[p; f], f \right\rangle + \left\langle \partial_p \hat{L}[p; f], \lambda \right\rangle \tag{8}$$

$$\hat{h}^\dagger[p; f] + \hat{h}[p; f] + \hat{L}^\dagger[p; \lambda] = 0 \tag{9}$$

Adjoint equations: introduction example

- Derivative $d_p f$ are computationally expensive

- Invert the operators $\hat{h}$ and $\hat{L}$ wherever they act on $\mathrm{d}_p f$, and collect coefficients of $\mathrm{d}_p f$ terms

- Resulting equation is:

$$\nabla_{\boldsymbol{p}} \mathcal{L}[\boldsymbol{p}; f, \lambda]\big|_{f,\lambda} = \left\langle \partial_{\boldsymbol{p}} \hat{h}[\boldsymbol{p}; f], f \right\rangle + \left\langle \partial_{\boldsymbol{p}} \hat{L}[\boldsymbol{p}; f], \lambda \right\rangle \tag{8}$$

$$\hat{h}^{\dagger}[\boldsymbol{p}; f] + \hat{h}[\boldsymbol{p}; f] + \hat{L}^{\dagger}[\boldsymbol{p}; \lambda] = 0 \tag{9}$$

Adjoint equations: introduction example

- Derivative $d_p f$ are computationally expensive

- Invert the operators $\hat{h}$ and $\hat{L}$ wherever they act on $\mathrm{d}_p f$, and collect coefficients of $\mathrm{d}_p f$ terms

- Resulting equation is:

$$\boldsymbol{\nabla}_{\boldsymbol{p}} \mathcal{L}[\boldsymbol{p}; f, \lambda]\big|_{f,\lambda} = \left\langle \partial_{\boldsymbol{p}} \hat{h}[\boldsymbol{p}; f], f \right\rangle + \left\langle \partial_{\boldsymbol{p}} \hat{L}[\boldsymbol{p}; f], \lambda \right\rangle \tag{8}$$

$$\hat{h}^{\dagger}[\boldsymbol{p}; f] + \hat{h}[\boldsymbol{p}; f] + \hat{L}^{\dagger}[\boldsymbol{p}; \lambda] = 0 \tag{9}$$

- Computational cost = cost of solving original system + solving adjoint equation
- Including more $\boldsymbol{p}$'s does not increase the computation

Table of Contents

Adjoint system for gyrokinetics

- Now let's adapt this method for gyrokinetics

Adjoint system for gyrokinetics

- Now let's adapt this method for gyrokinetics
- Recall that this system has dependence on g_ν (through the gyrokinetic equation), ϕ (quasineutrality), $A_\parallel$, and $B_\parallel$ (Ampere's law)

Adjoint system for gyrokinetics

- Now let's adapt this method for gyrokinetics
- Recall that this system has dependence on g_ν (through the gyrokinetic equation), ϕ (quasineutrality), $A_\parallel$, and $B_\parallel$ (Ampere's law)
- Given that we want to take deriavatives of our optimisation lagrangian with respect to $\boldsymbol{p}$, we will encounter terms of the form

$$\boldsymbol{\nabla}_{\boldsymbol{p}} g_\nu, \quad \boldsymbol{\nabla}_{\boldsymbol{p}} \phi, \quad \boldsymbol{\nabla}_{\boldsymbol{p}} A_\parallel, \quad \boldsymbol{\nabla}_{\boldsymbol{p}} \delta B_\parallel \tag{10}$$

  These are computationally expensive to calculate so we set their coefficients to zero

Adjoint system for gyrokinetics

- Now let's adapt this method for gyrokinetics
- Recall that this system has dependence on g_ν (through the gyrokinetic equation), ϕ (quasineutrality), $A_\parallel$, and $B_\parallel$ (Ampere's law)
- Given that we want to take deriavatives of our optimisation lagrangian with respect to $\boldsymbol{p}$, we will encounter terms of the form

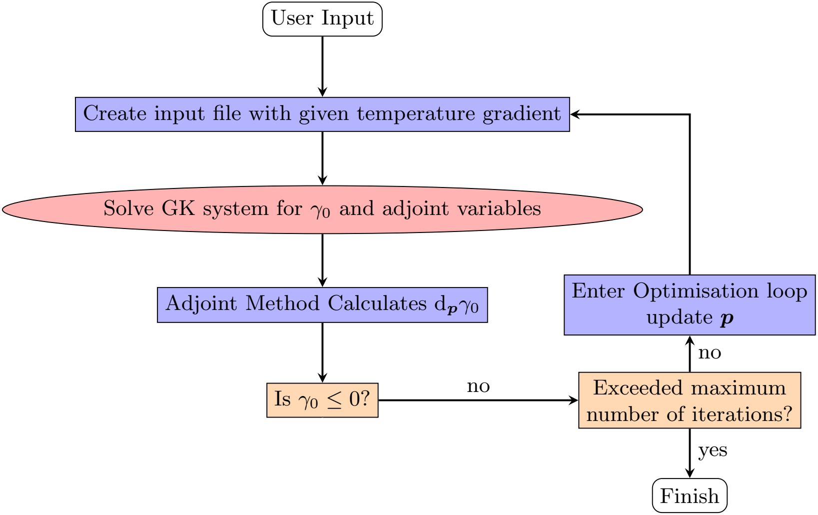
$$\boldsymbol{\nabla}_{\boldsymbol{p}} g_\nu, \quad \boldsymbol{\nabla}_{\boldsymbol{p}} \phi, \quad \boldsymbol{\nabla}_{\boldsymbol{p}} A_\parallel, \quad \boldsymbol{\nabla}_{\boldsymbol{p}} \delta B_\parallel \tag{10}$$

  These are computationally expensive to calculate so we set their coefficients to zero
- With some algebra, one can show that

$$\mathrm{d}_{\boldsymbol{p}} \gamma_0 = \text{stuff} \tag{11}$$

  where γ_0 is the linear growth rate, and "stuff" is a result of taking the adjoints of our functional operators

Adjoint optimisation app



Adjoint optimisation app

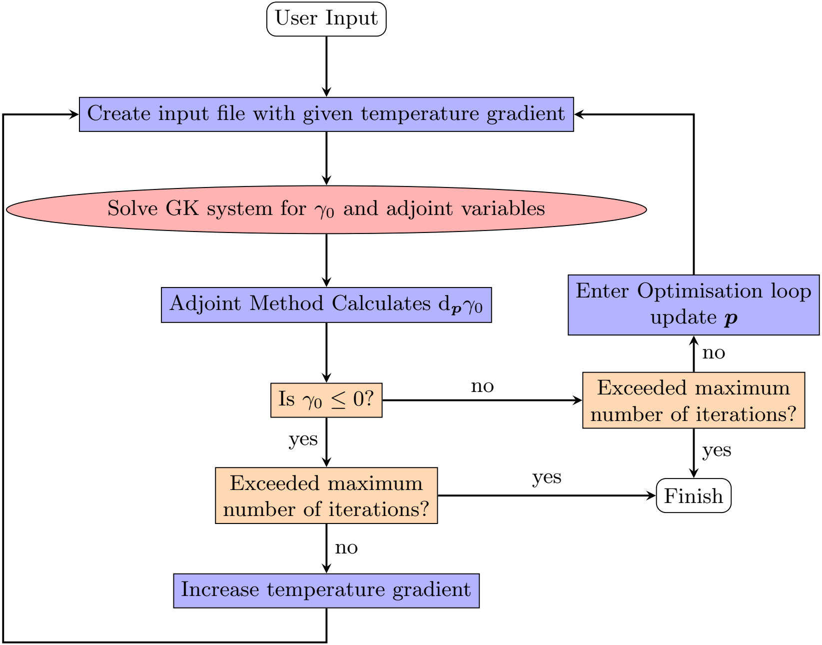


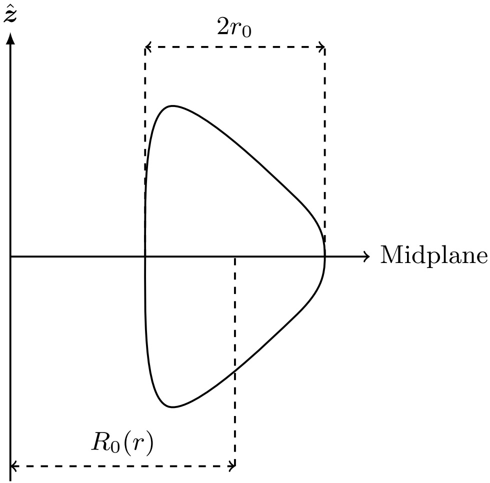
Table of Contents

System of interest

- Choose to consider perturbations to Miller geometry formalism
- Transform to polar coordinates:

$$R(r,\theta) = R_0(r) + r\cos\left[\theta + \sin(\theta)\delta(r)\right] \tag{12}$$

$$Z(r,\theta) = r\ \kappa(r)\sin(\theta) \tag{13}$$

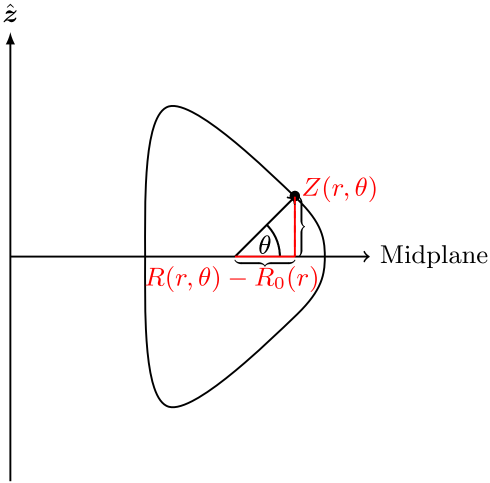


System of interest

- Choose to consider perturbations to Miller geometry formalism
- Transform to polar coordinates:

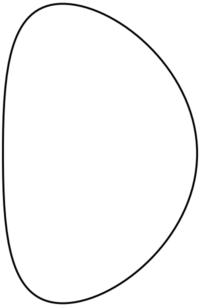
$$R(r,\theta) = R_0(r) + r\cos\left[\theta + \sin(\theta)\delta(r)\right] \tag{12}$$

$$Z(r,\theta) = r\,\kappa(r)\sin(\theta) \tag{13}$$

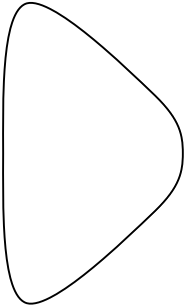


Triangularity

- Can vary triangularity, δ, of flux surface



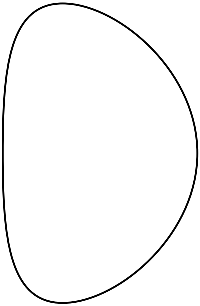
Small δ



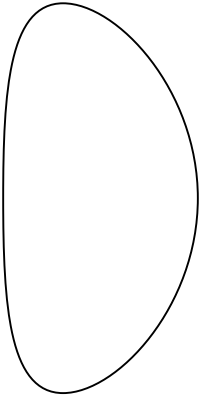
Large δ

Elongation

- Can very elongation, κ, of flux surface



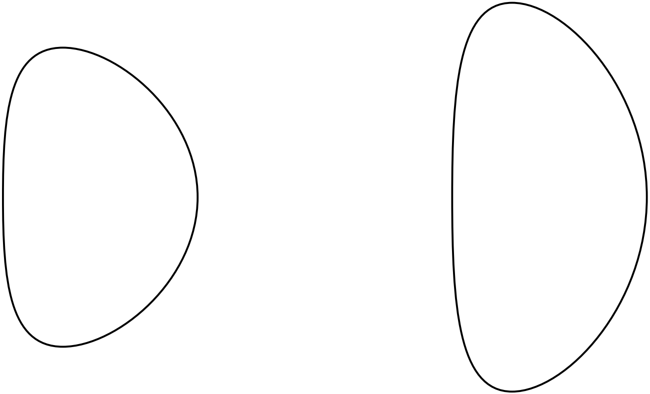
Small κ



Large κ

Elongation

- Can very elongation, κ, of flux surface

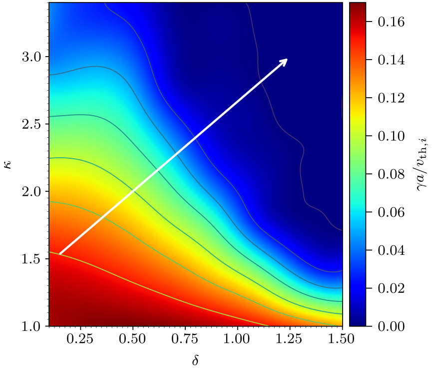


Small κ Large κ

- Can generalise to vector of parameters: $\boldsymbol{p} = \{r, R_0, \Delta, q, \hat{s}, \kappa, \kappa', R_{\rm geo}, \delta, \delta', \beta'\}$ **at no further computational cost!**

Adjoint + Levenberg-Marquardt

- Comparison with finite difference scan when varying δ, and κ
- Use adjoint method to find gradient, and use Levenberg-Marquardt algorithm for optimisation loop



Increasing the temperature gradient

- Second temperature iteration
- $\left.\frac{R_0}{L_{T_i}}\right|_{\text{previous}} = 2.42, \qquad \left.\frac{R_0}{L_{T_i}}\right|_{\text{new}} = 3.49$

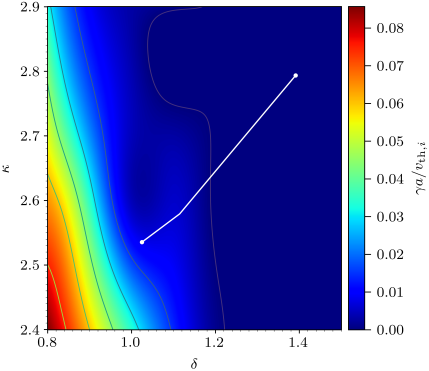


Table of Contents

Summary of adjoint work

- Developed a generalised formalism for calculating derivatives of linear growth rate using adjoint
- Implemented adjoint equations into δf gyrokinetic code `stella` for the case of Miller geometry in an electrostatic, collisionless regime
- Have shown an example case by varying triangularity and elongation, for which the adjoint method has show significant improvements in terms of computational cost

Backup slides: Constraint equations

$$
\begin{aligned}
\hat{G}_{\boldsymbol{k},\nu} &= \gamma_{\boldsymbol{k},0} g_{\boldsymbol{k},\nu,0} + v_{th,\nu} v_\parallel \, \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z \left[\frac{\partial g_{\boldsymbol{k},\nu,0}}{\partial z} + \frac{Z_\nu}{T_\nu} \frac{\partial \left\langle \chi_{\boldsymbol{k},0} \right\rangle_{\boldsymbol{R}_\nu}}{\partial z} e^{-v_\nu^2} \right] \\
&\quad + i\omega_{\star,\boldsymbol{k},\nu} e^{-v_\nu^2} \left\langle \chi_{\boldsymbol{k},0} \right\rangle_{\boldsymbol{R}_\nu} + i\omega_{d,\boldsymbol{k},\nu} \left[g_{\boldsymbol{k},\nu,0} + \frac{Z_\nu}{T_\nu} \left\langle \chi_{\boldsymbol{k},0} \right\rangle_{\boldsymbol{R}_\nu} \; e^{-v_\nu^2} \right] \\
&\quad - v_{th,\nu} \mu_\nu \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} B_0 \frac{\partial g_{\boldsymbol{k},\nu,0}}{\partial v_\parallel} + 2 \frac{Z_\nu}{m_\nu} \mu_\nu \hat{\boldsymbol{b}} \cdot B_0 e^{-v_\nu^2} J_{0,\boldsymbol{k},\nu} A_{\parallel,\boldsymbol{k},0} - \hat{C}_{\boldsymbol{k},\nu}[g_{\boldsymbol{k},\nu,0}] \\
\hat{Q}_{\boldsymbol{k}} &= \sum_\nu Z_\nu n_\nu \left\{ \frac{2B_0}{\sqrt{\pi}} \int \mathrm{d}^2\hat{v} \; J_{0,\boldsymbol{k},\nu} g_{\boldsymbol{k},\nu,0} + \frac{Z_\nu}{T_\nu} \left(\Gamma_{0,\boldsymbol{k},\nu} - 1 \right) \; \phi_{\boldsymbol{k},0} + \frac{1}{B_0} \Gamma_{1,\boldsymbol{k},\nu} \delta B_{\parallel,\boldsymbol{k},0} \right\} \; , \\
\hat{M}_{\boldsymbol{k}} &= \; - \frac{\beta}{\left(k_\perp \rho_r\right)^2} \sum_\nu Z_\nu n_\nu v_{th,\nu} \frac{2B}{\sqrt{\pi}} \int \mathrm{d}^2\hat{v} \; v_\parallel J_{0,\boldsymbol{k},\nu} g_{\boldsymbol{k},\nu,0} \\
&\quad + \left[1 + \frac{\beta}{\left(k_\perp \rho_r\right)^2} \sum_\nu \frac{Z_\nu n_\nu}{m_\nu} \Gamma_{0,\boldsymbol{k},\nu} \right] A_{\parallel,\boldsymbol{k},0} \qquad (14) \\
\hat{N}_{\boldsymbol{k}} &= 2\beta \sum_\nu n_\nu T_\nu \; \frac{2B_0}{\sqrt{\pi}} \int \mathrm{d}^2\hat{v} \mu_\nu \frac{J_{0,\boldsymbol{k},\nu}}{a_{\boldsymbol{k},\nu}} g_{\boldsymbol{k},\nu,0} + \left[\frac{\beta}{2B_0} \sum_\nu Z_\nu n_\nu \Gamma_{1,\boldsymbol{k},\nu} \right] \phi_{\boldsymbol{k},0} \\
&\quad + \left[1 + \frac{\beta}{2B_0} \sum_\nu Z_\nu n_\nu T_\nu \Gamma_{2,\boldsymbol{k},\nu} \right] \delta B_{\parallel,\boldsymbol{k},0} \qquad (15)
\end{aligned}
$$

Backup slides: Constraint equations

$$\begin{aligned} \mathrm{d}_{\boldsymbol{p}}\mathcal{L} =& \langle \partial_{\boldsymbol{p}} \hat{G}_\nu, \lambda_\nu \rangle_{z,v_\nu} + \langle \partial_{\boldsymbol{p}} \hat{Q}, \xi \rangle_z + \langle \partial_{\boldsymbol{p}} \hat{M}, \zeta \rangle_z + \langle \partial_{\boldsymbol{p}} \hat{N}, \sigma \rangle_z \\ =& \, 0 \end{aligned} \tag{16}$$

$$\mathrm{d}_{\boldsymbol{p}}\mathcal{L} = \underbrace{\langle \partial_{\boldsymbol{p}} \hat{G}_\nu, \lambda_\nu \rangle_{z,v_\nu} + \langle \partial_{\boldsymbol{p}} \hat{Q}, \xi \rangle_z + \langle \partial_{\boldsymbol{p}} \hat{M}, \zeta \rangle_z + \langle \partial_{\boldsymbol{p}} \hat{N}, \sigma \rangle_z}_{\text{only contains partial derivatives}} \tag{17}$$

$$\begin{aligned} \mathrm{d}_{\boldsymbol{p}}\mathcal{L} =& \langle \partial_{\boldsymbol{p}} \hat{G}_\nu, \lambda_\nu \rangle_{z,v_\nu} + \langle \partial_{\boldsymbol{p}} \hat{Q}, \xi \rangle_z + \langle \partial_{\boldsymbol{p}} \hat{M}, \zeta \rangle_z + \langle \partial_{\boldsymbol{p}} \hat{N}, \sigma \rangle_z \\ & \hat{G}_\nu = \gamma_0 g_\nu + \hat{L}_\nu \\ & \mathcal{L} = 0, \quad \mathrm{d}_{\boldsymbol{p}}\mathcal{L} = 0 \end{aligned} \tag{18}$$

Backup slides: Adjoint Equations

$$\gamma_0^* \acute{\lambda}_\nu + v_{th,\nu} v_\parallel \; \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} z \frac{\partial \acute{\lambda}_\nu}{\partial z} - v_{th,\nu} \; \mu_\nu \; \hat{\boldsymbol{b}} \cdot \boldsymbol{\nabla} B \frac{\partial \acute{\lambda}_\nu}{\partial v_\parallel} - i\omega_{d,\nu} \acute{\lambda}_\nu + Z_\nu n_\nu J_{0,\nu} \xi - \frac{\beta}{(k_\perp \rho_r)^2} Z_\nu n_\nu v_{th,\nu} J_{0,\nu} v_\parallel \zeta + 2\beta T_\nu \mu_\nu \frac{J_{1,\nu}}{a_\nu} \sigma - \hat{C}_\nu[\acute{\lambda}_\nu] = 0 \; , \tag{19}$$

$$\bar{\eta}\xi + \sum_\nu \frac{2B}{\sqrt{\pi}} \int \mathrm{d}^2\hat{v} \left[i\omega_{*,\nu} + \frac{Z_\nu}{T_\nu} \gamma_0^* \right] J_{0,\nu} \acute{\lambda}_\nu = 0 \; , \tag{20}$$

$$\zeta - \sum_\nu \frac{2B}{\sqrt{\pi}} \int \mathrm{d}^2\hat{v} \; (2 v_{th,\nu} v_\parallel) \left[i\omega_{*,\nu} + \frac{Z_\nu}{T_\nu} \gamma_0^* \right] J_{0,\nu} \acute{\lambda}_\nu = 0 \; , \tag{21}$$

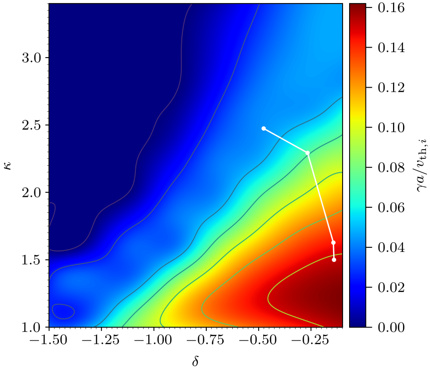
$$\sigma - \sum_\nu \frac{2B}{\sqrt{\pi}} \int \mathrm{d}^2\hat{v} \; \left(4\mu_\nu \frac{J_{1,\nu}}{a_\nu} \right) \left[i\omega_{*,\nu} + \frac{Z_\nu}{T_\nu} \gamma_0^* \right] J_{0,\nu} \acute{\lambda}_\nu = 0 \; . \tag{22}$$

Backup slides: Adjoint for Miller geometry

Define vector $\boldsymbol{p} \coloneqq \{r, R_0, \Delta, q, \hat{s}, \kappa, \kappa', R_{\mathrm{geo}}, \delta, \delta', \beta'\}$

- Minor radius - r
- Major radius - R_0
- Shafranov shift - Δ
- Safety factor - $q = \frac{1}{2\pi} \int_0^{2\pi} d\theta \frac{\boldsymbol{B} \cdot \boldsymbol{\nabla} \zeta}{\boldsymbol{B} \cdot \boldsymbol{\nabla} \theta}$
- Magnetic shear - $\hat{s} \doteq \frac{r}{q} q'$
- Elongation - κ, and κ'
- Proxy for reference magnetic field - $R_{\mathrm{geo}} = \frac{I(r)}{a B_{\mathrm{ref}}}$
- Triangularity - δ, and δ'
- Plasma beta derivative - $\beta' = -\frac{4\pi p'}{B_{\mathrm{ref}}^2}$

Backup slides: Negative triangularity

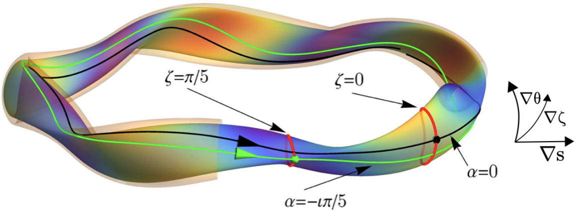


- LM algorithm has difficulty with finding local minima rather than global minima

Backup: Full Flux Surface

Future work - Full Flux Surface (FFS)

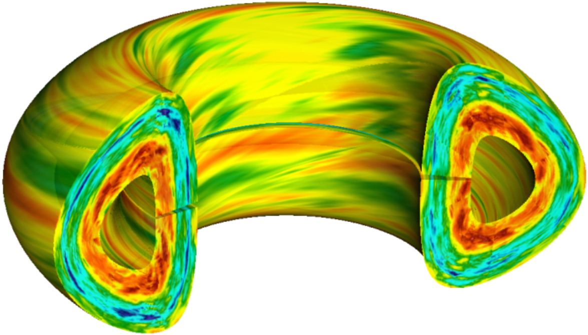
- Currently `stella` uses flux-tube approximation
- Different field lines are decoupled



González-Jerez et al. 2021

Future work - Full Flux Surface (FFS)

- FFS `stella` allows non-linearly coupling of different field lines
- Explore how zonal flows, $k_y = 0$ affect stability



Future work - Full Flux Surface (FFS)

To do:

- Benchmark explicit, adiabatic version by taking $\rho_* \to 0$
- Make FFS `stella` implicit
- Investigate implications, if any, of zonal modes