Stellarator Optimization with DESC

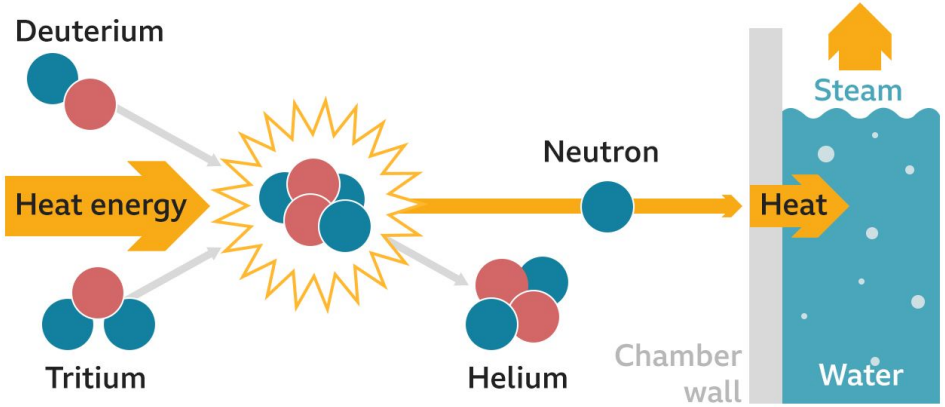
R. Conlin, D. W. Dudt, D. Panici, E. Kolemen

August 1, 2022
2022 Conference on Computational Physics

PPPL
Princeton Plasma Control
control.princeton.edu
andlinger center
for energy+the environment
PRINCETON UNIVERSITY

Nuclear fusion could help solve the global energy and environmental crises

1	2	3	4
Hydrogen atoms are heated	Fusion reaction	Helium, neutron and energy released	Neutron energy heats water

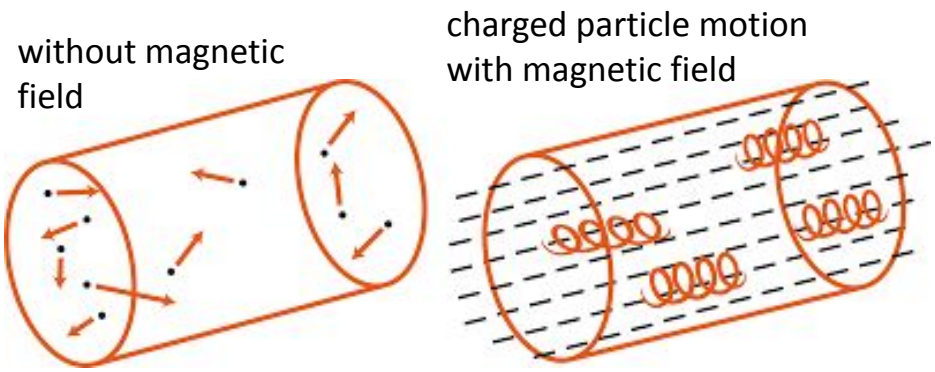


BBC

- Sustainable fuel supply
- No greenhouse gas emissions
- No long-lived radioactive waste
- No risk of runaway reactions
- No risk of nuclear proliferation
- Firm generation source

How to confine a plasma

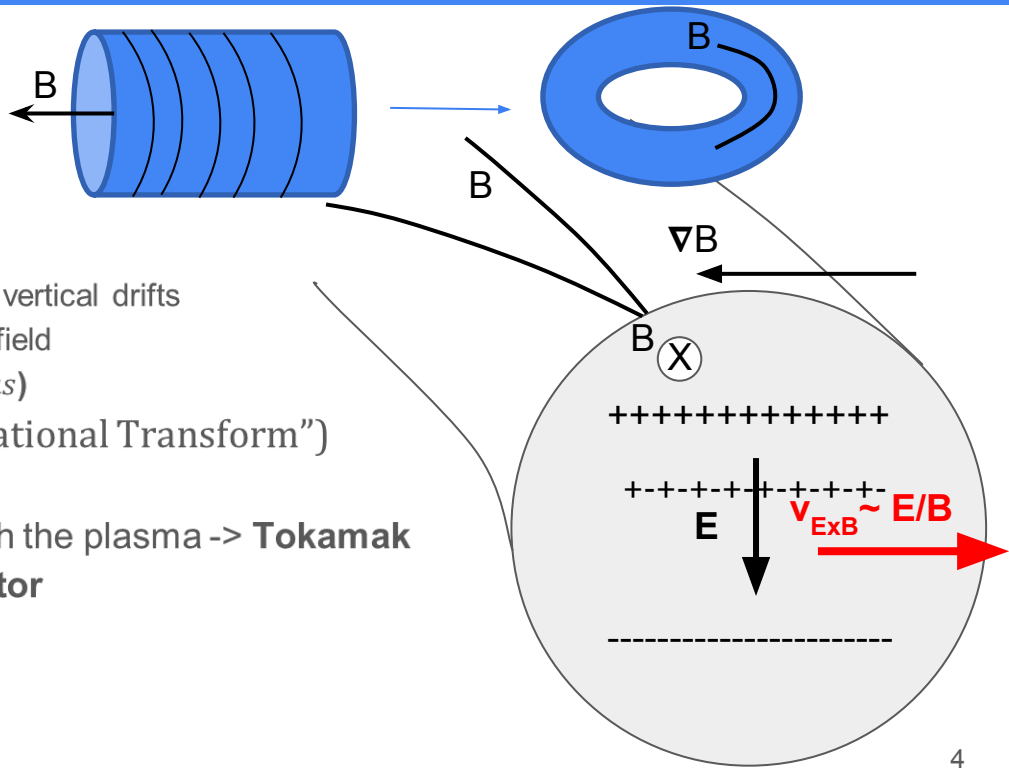
- Fusion reaction requires extremely high temperatures (O(1e8 K))
- How to confine hot fusion plasma in place?
- Magnetic confinement - use Lorentz forces
- Charged particles orbit magnetic field lines



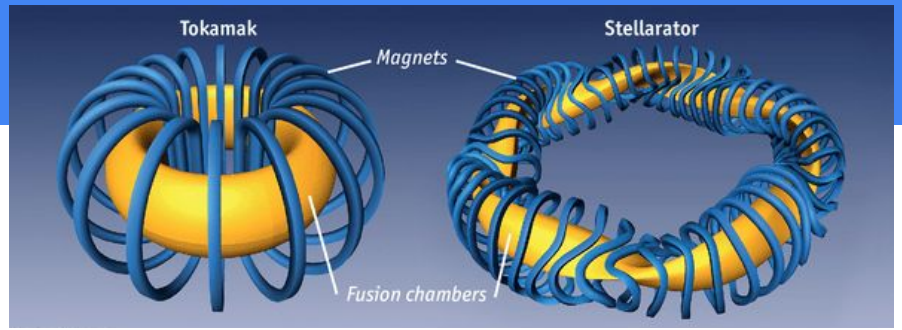
Jason Ginsberg

Magnetic Confinement Geometry

- Simplest Magnetic field is a solenoid
  - Problem: Parallel Confinement
- Solution: Plug Ends by 'biting its tail'
  - Toroidal confinement
- However, confinement still an issue!
  - Curvature and gradient introduced in **B** leads to vertical drifts
  - Causes charge separation in $O(ms)$, leads to **E** field
  - **E** x **B** drift causes particles to exit plasma in $O(\mu s)$
- Solution: Make B field **twist poloidally** ("Rotational Transform")
  - How to create this twist?
    - By driving a **toroidal current** through the plasma -> **Tokamak**
    - By twisting external coils -> **Stellarator**



Magnetic Confinement Devices: Tokamak vs. Stellarator



https://www.economist.com/science-and-technology/2015/10/24/stellar-work

Tokamaks

- Axisymmetric
  - Simpler geometry
  - Guaranteed particle confinement
    - Due to Noether's Theorem
- Requires substantial plasma current
  - Must be driven
  - NOT steady state!
- Source of free energy for instabilities -> disruptions

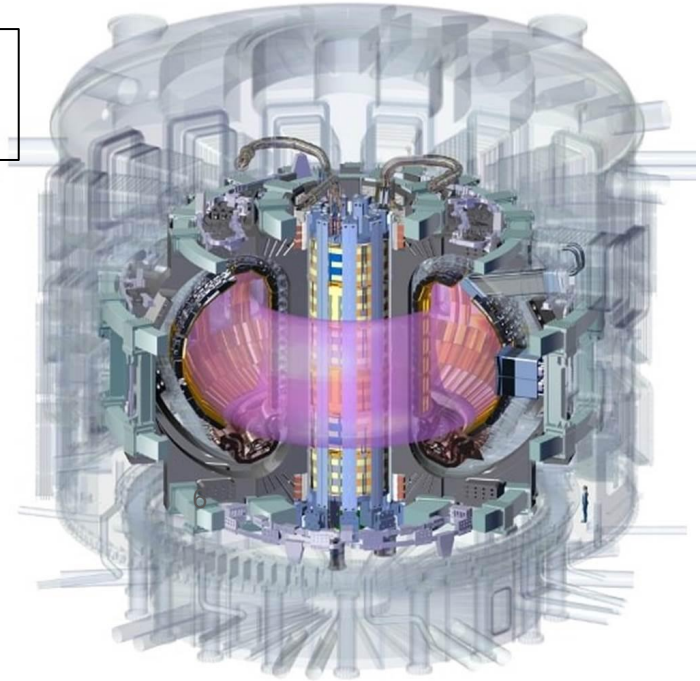
Stellarators

- Inherently 3-D
  - Complex geometry and coils
  - Confinement not guaranteed
    - certain fields exist which recover this (Quasisymmetry)
  - **Larger design space**
- Does not need plasma current
  - Steady state
  - No disruptions

Plasma Equilibria: What and Why?

<u>Plasma Equilibrium</u>: Configuration of magnetic fields that describes a plasma in **steady-state**

- Reactor Design and Optimization
- Experimental Reconstruction
- Necessary for ANY plasma physics studies
  - Particle Transport
  - Stability



https://www.ansys.com/news-center/press-releases/ansys-enables-iter-organization-design-worlds-largest-highly-sustainable-nuclear-fusion-power-plant

Plasma Model – Ideal MHD

Mass: $$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

Momentum: $$\rho \frac{d\mathbf{v}}{dt} = \mathbf{J} \times \mathbf{B} - \nabla p$$

Energy: $$\frac{d}{dt}\left(\frac{p}{\rho^{\gamma}}\right) = 0$$

Ohm's law: $$\mathbf{E} + \mathbf{v} \times \mathbf{B} = 0$$

Maxwell: $$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J}$$

$$\nabla \cdot \mathbf{B} = 0$$

Freidberg *Ideal MHD* (2014)

ρ	Ion Mass Density
$\mathbf{v}$	Ion Flow Velocity
p	Pressure
$\mathbf{B}$	Magnetic Field
$\mathbf{J}$	Current Density

Plasma Model – Ideal MHD Equilibrium

Mass:	$\cancel{\frac{\partial \rho}{\partial t}} + \nabla \cdot (\cancel{\rho \mathbf{v}}) = 0$
Momentum:	$\cancel{\rho \frac{d\mathbf{v}}{dt}} = \mathbf{J} \times \mathbf{B} - \nabla p$
Energy:	$\cancel{\frac{d}{dt}\left(\frac{p}{\rho^{\gamma}}\right)} = 0$
Ohm's law:	$\mathbf{E} + \cancel{\mathbf{v}} \times \mathbf{B} = 0$
Maxwell:	$\nabla \times \mathbf{E} = -\cancel{\frac{\partial \mathbf{B}}{\partial t}}$
	$\nabla \times \mathbf{B} = \mu_0 \mathbf{J}$
	$\nabla \cdot \mathbf{B} = 0$

Freidberg *Ideal MHD* (2014)

$$\frac{\partial}{\partial t} \to 0$$

$$\boldsymbol{v} \to 0$$

$$\boldsymbol{J} \times \boldsymbol{B} = \nabla p$$

$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J}$$

$$\nabla \cdot \boldsymbol{B} = 0$$

Ideal MHD Equilibrium - What Do the Equations Tell Us?

- Physically, plasma is in equilibrium with **pressure gradient** balanced by the **J x B force**
- **B, J** lie on surfaces of constant pressure
  - **Flux Surfaces** $\rightarrow \boldsymbol{B} \cdot \boldsymbol{n} = 0$

$$\boxed{\mathbf{B} \cdot \nabla p = 0} \quad \boxed{\mathbf{J} \cdot \nabla p = 0}$$

$$\boldsymbol{J} \times \boldsymbol{B} = \nabla p$$

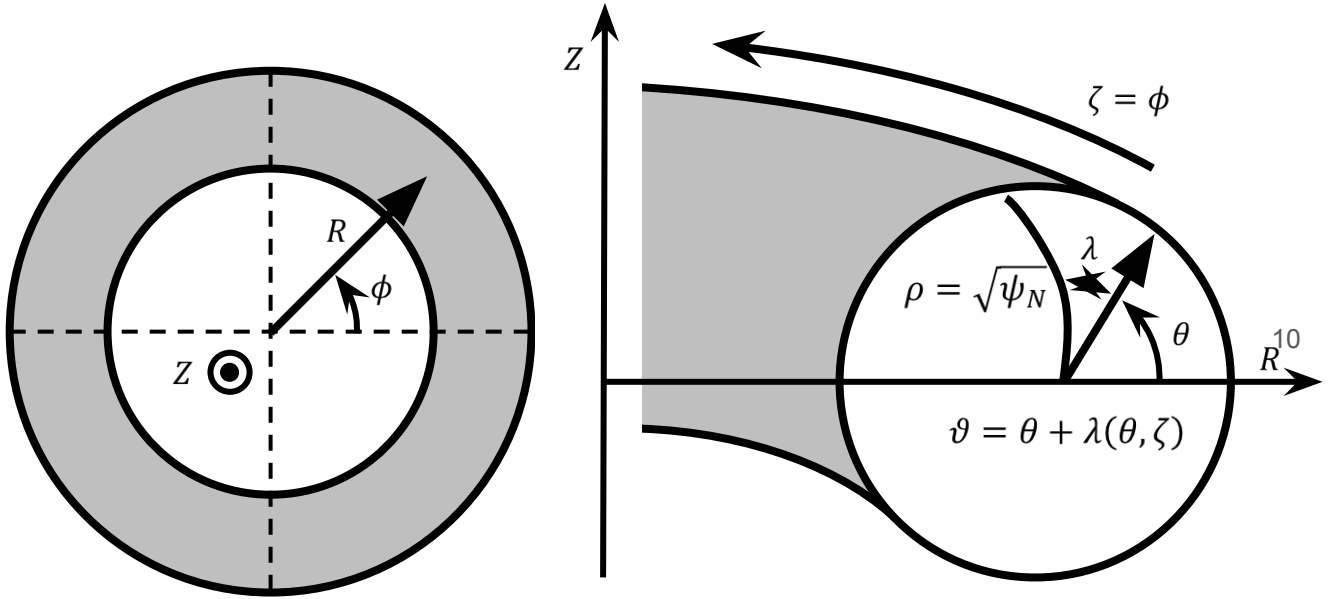
$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J}$$

$$\nabla \cdot \boldsymbol{B} = 0$$

- Mathematically, is a coupled system of nonlinear PDEs
- **Goal of Equilibrium Solving**: Find the magnetic field **B** that satisfies these equations, given some BCs and inputs

Solve the “inverse” equilibrium problem for the shapes of the flux surfaces

Instead of solving for $\boldsymbol{B}$ directly, we solve for the straight field-line coordinates map: $(R, \phi, Z) \mapsto (\rho, \theta, \zeta)$



Magnetic field form

$$\boldsymbol{B} = B^{\zeta}\left(\iota \boldsymbol{e}_{\vartheta} + \boldsymbol{e}_{\zeta}\right)$$

assumes

$$\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0, \boldsymbol{B} \cdot \boldsymbol{\nabla}\rho = 0$$

Coordinate map:

$$R(\rho, \theta, \zeta)$$
$$Z(\rho, \theta, \zeta)$$
$$\lambda(\rho, \theta, \zeta)$$

Solution is represented with global Fourier-Zernike[1,2] spectral basis functions

spectral coefficients

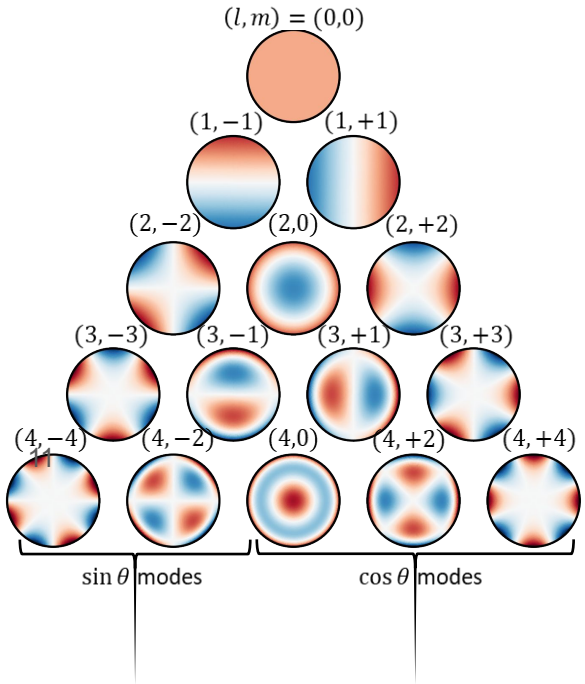
$$R(\rho,\theta,\zeta) = \sum_{lmn} R_{lmn} \mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$

Zernike polynomials

Fourier series

$$\mathcal{Z}_l^m(\rho,\theta) = \begin{cases} \mathcal{R}_l^{|m|}\cos(|m|\theta) \text{ for } m \geq 0 \\ \mathcal{R}_l^{|m|}\sin(|m|\theta) \text{ for } m < 0 \end{cases}$$

one-sided Jacobi polynomials



- Similar discretization for $Z(\rho,\theta,\zeta)$ and $\lambda(\rho,\theta,\zeta)$

[1]Zernike, *Mon. Not. R. Astron. Soc.* (1934).
[2]Loomis, *ASTM STP* (1978).

The global Fourier-Zernike basis is well suited for the toroidal domain

- Periodic boundary conditions for poloidal & toroidal dimensions
- Satisfies analyticity conditions at the magnetic axis[3,4]:

$$f(\rho,\theta) = \sum_m \rho^m \left(a_{m,0} + a_{m,2}\rho^2 + \cdots\right)\cos(m\theta) + \sum_m \rho^m \left(b_{m,0} + b_{m,2}\rho^2 + \cdots\right)\sin(m\theta)$$

- Minimizes the total number of unknown variables
- Exponential convergence (if solution exists and is smooth)

[3]Boyd et al., *J. Comput. Phys.* (2011).
[4]Lewis et al., *J. Math. Phys.* (1990).

The MHD equilibrium equation is solved using a pseudo-spectral collocation method

Substituting the spectral expansions into the original PDE

$$\boldsymbol{F} \equiv (\boldsymbol{\nabla} \times \boldsymbol{B}) \times \boldsymbol{B} - \mu_0 \boldsymbol{\nabla} p = \boldsymbol{0}$$

reduces it to a system of nonlinear algebraic equations

equilibrium force balance equations evaluated in real space

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{0}$$

errors evaluated at a set of collocation points

spectral coefficients

boundary conditions

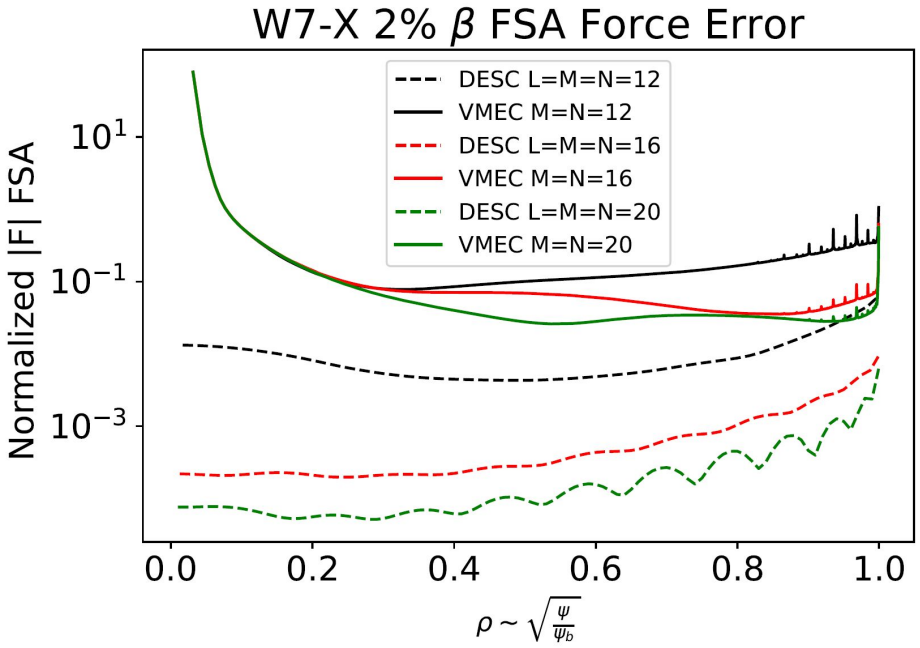
which is then solved by a Gauss-Newton method (super-linear convergence)

$$\boldsymbol{x}^* = \mathrm{argmin}_{\boldsymbol{x}}\left(\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})\|^2\right) \qquad \dim(\boldsymbol{f}) \geq \dim(\boldsymbol{x})$$

DESC achieves lower error than VMEC

Comparison to VMEC [Hirshman, 1983]

- Orders of magnitude lower error, especially near the axis
- For more details, see D. Panici, *Quick and Accurate 3D MHD Equilibria with DESC*
  - this session



Utilize JAX for automatic differentiation and JIT compilation

- JAX is developed by Google, using the same backend as TensorFlow
- Automatic differentiation provides exact derivatives of arbitrary order

  Jacobian matrix required for Newton method: $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1} \boldsymbol{f}(\boldsymbol{x})$

- Just-in-time (JIT) compilation improves speed and memory usage using Accelerated Linear Algebra (XLA)
- Runs on both CPU & GPU
- Easy to implement

```
import jax.numpy as jnp
```

https://github.com/google/jax

Automatic differentiation enables efficient perturbations about equilibrium solutions

- Neighboring equilibrium with different boundary conditions $\boldsymbol{c}^* = \boldsymbol{c} + \Delta\boldsymbol{c}$:

1st-order Taylor expansion:

$$\boldsymbol{f}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}\Delta\boldsymbol{c}$$

$\boldsymbol{f}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) \rightarrow \boxed{0}$, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) \rightarrow \boxed{0}$

$$\Delta\boldsymbol{x} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}\Delta\mathbf{c}$$

Jacobians computed with AD

new equilibrium solution

$$\boldsymbol{x}^* = \boldsymbol{x} + \Delta\boldsymbol{x}$$

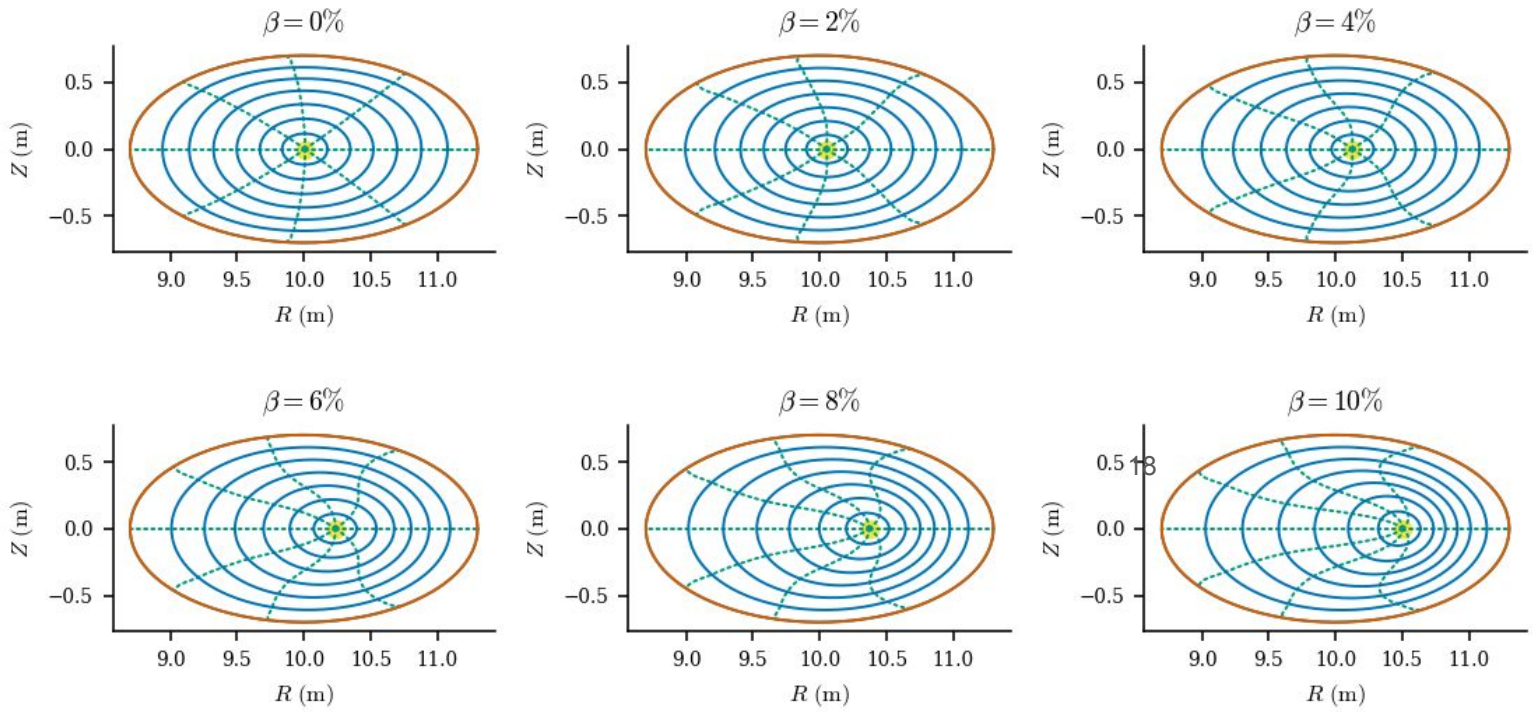
- These perturbations reveal branches of equilibria solutions
- Extended to higher-order approximations with little additional cost

(only Jacobian-vector products required, not full Hessian matrices)

Autodiff allows efficient computation of high order expansions

- Computing high order perturbations requires high order derivatives such as $\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}, \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x} \partial \boldsymbol{c}}$ etc
- Materializing full 2nd derivative tensor $\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}$ would take 100s-1000s GB of ram
- "Halley's method": only need directional derivatives: $\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2} \Delta \boldsymbol{x}_1 \Delta \boldsymbol{x}_1$
  - aka "Jacobian vector products"
- Can be computed using automatic differentiation with significantly reduced memory
- Implemented up to 3rd order expansion in the code

Example: perturbations compute parameter scans "for free"



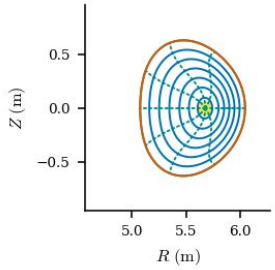
Conlin et al. *Nuclear Fusion* (under review)

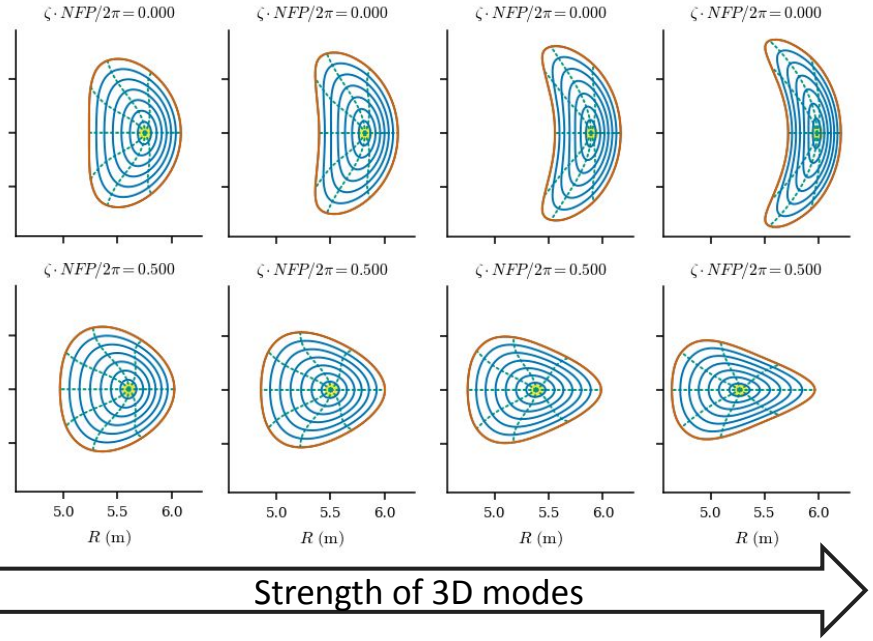
Continuation method example: from axisymmetry to 3D stellarator boundary

Intermediate solutions

Initial solution

axisymmetric boundary (tokamak)





Final solution

strongly shaped stellarator

Strength of 3D modes

Conlin et al. *Nuclear Fusion* (under review)

Can optimize parameters while maintaining the equilibrium constraint

- Let $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c})$ be a physics/engineering objective that we wish to optimize

Taylor expansion:

$$\boldsymbol{g}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c}) + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \Delta\boldsymbol{x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{c}} \Delta\boldsymbol{c}$$

$\boxed{0}$

equilibrium perturbation

least-squares problem:

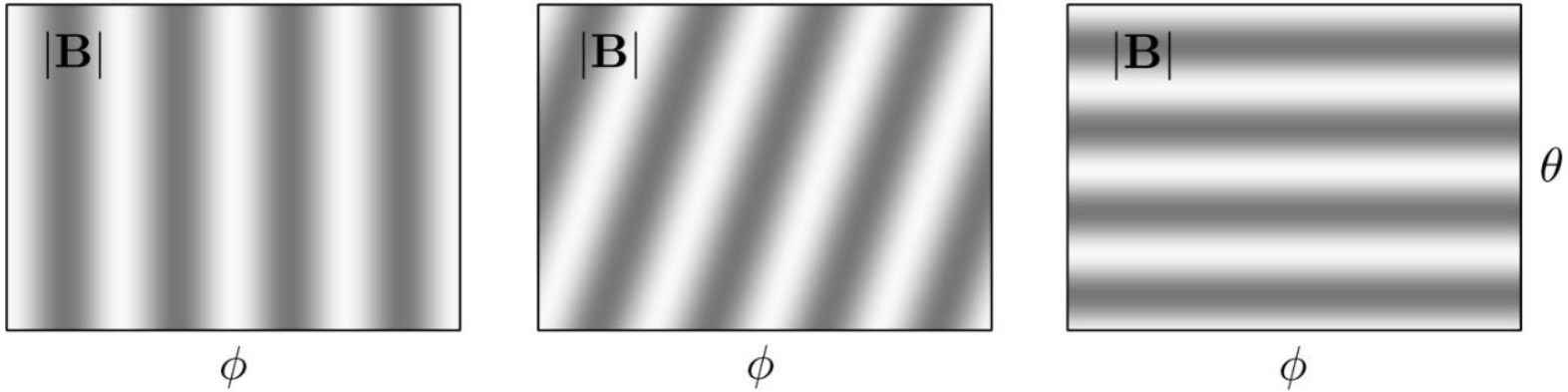
$$\left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{c}}\right] \Delta\boldsymbol{c} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c})$$

- Yields the optimal perturbation to improve the objective $\boldsymbol{c}^* = \boldsymbol{c} + \Delta\boldsymbol{c}$
- Extended to higher-order approximations with little additional cost
- Used in an adaptive Gauss-Newton trust-region optimization method

Quasisymmetry

Quasisymmetry is a minimal property of the magnetic field for the dynamics of charged particles to exhibit an approximately conserved momentum (in gyro-ordering) [Rodriguez 2022]

Can be described by the appearance of magnitude of **B** in *Boozer coordinates*



Quasi-poloidal Quasi-helical Quasi-axial

Multiple Quasi-Symmetry (QS) metrics

- Boozer Coordinates

$$\boldsymbol{B} = I(\psi)\boldsymbol{\nabla}\vartheta_B + G(\psi)\boldsymbol{\nabla}\zeta_B + K(\psi,\vartheta_B,\zeta_B)\boldsymbol{\nabla}\psi$$

$$|\boldsymbol{B}| = B(\psi, M\vartheta_B - N\zeta_B)$$

$$\boldsymbol{f}_B = \left\{B_{mn}\middle|\frac{m}{n} \neq \frac{M}{N}\right\}$$

- Two-Term

$$f_C = (M\iota - N)(\boldsymbol{B}\times\boldsymbol{\nabla}\psi)\cdot\boldsymbol{\nabla}B - (MG + NI)\boldsymbol{B}\cdot\boldsymbol{\nabla}B$$

$$\boldsymbol{f}_C = \{f_C(\theta_i,\zeta_j)|\theta_i \in [0,2\pi), \zeta_j \in [0, 2\pi/NFP)\}$$

- Triple Product

$$f_T = \boldsymbol{\nabla}\psi \times \boldsymbol{\nabla}B \cdot \boldsymbol{\nabla}(\boldsymbol{B}\cdot\boldsymbol{\nabla}B)$$

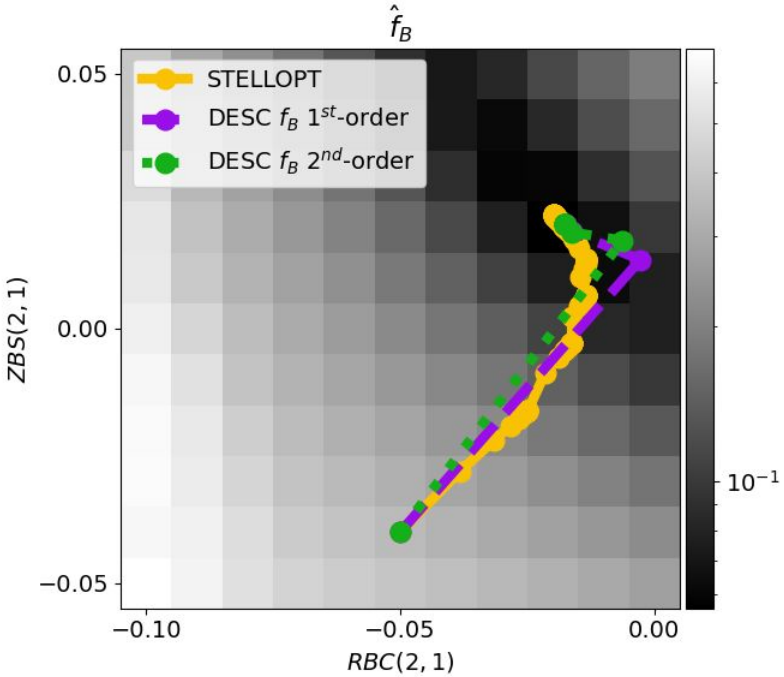
$$\boldsymbol{f}_T = \{f_T(\rho_i,\theta_j,\zeta_k)|\rho_i \in [0,1], \theta_j \in [0,2\pi), \zeta_k \in [0, 2\pi/NFP)\}$$

Optimization example: optimizing two boundary modes to improve quasi-symmetry

Benchmark code: STELLOPT[6]

- Levenberg-Marquardt optimization algorithm
- Finite differences through VMEC equilibria + other code to compute quasi-symmetry cost

"1st-order" expansion is 2nd-order in the scalar optimization cost



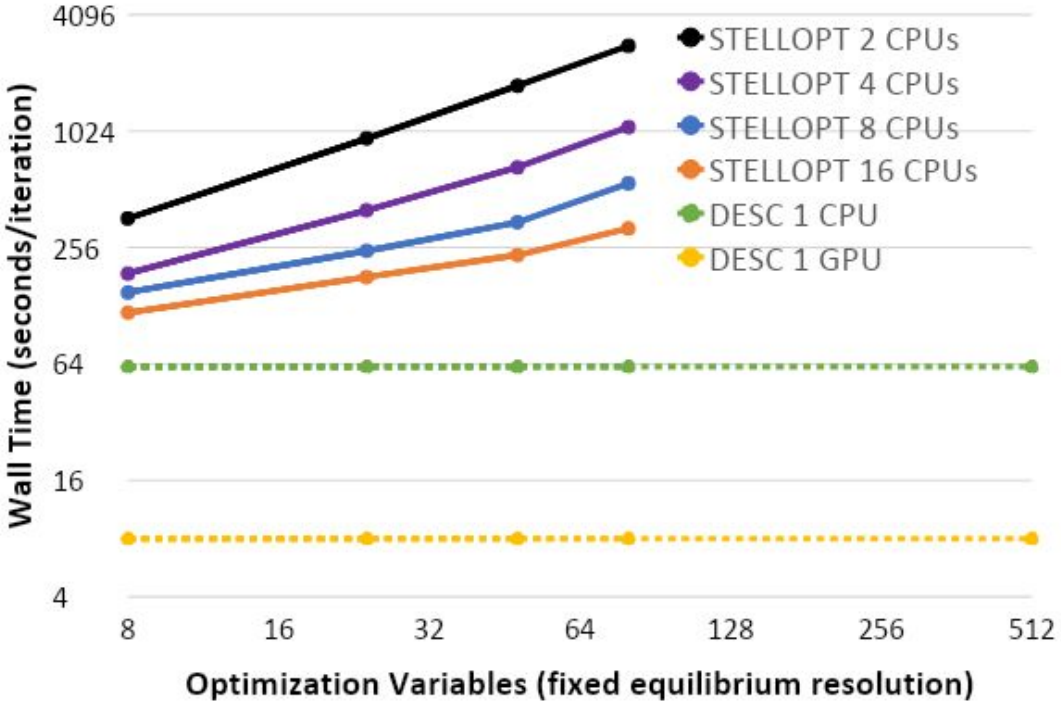
Dudt et al. *Nuclear Fusion* (under review)

[6]Spong et al., *Phys. Plasmas* (1998).

DESC achieves orders of magnitude faster stellarator optimization times

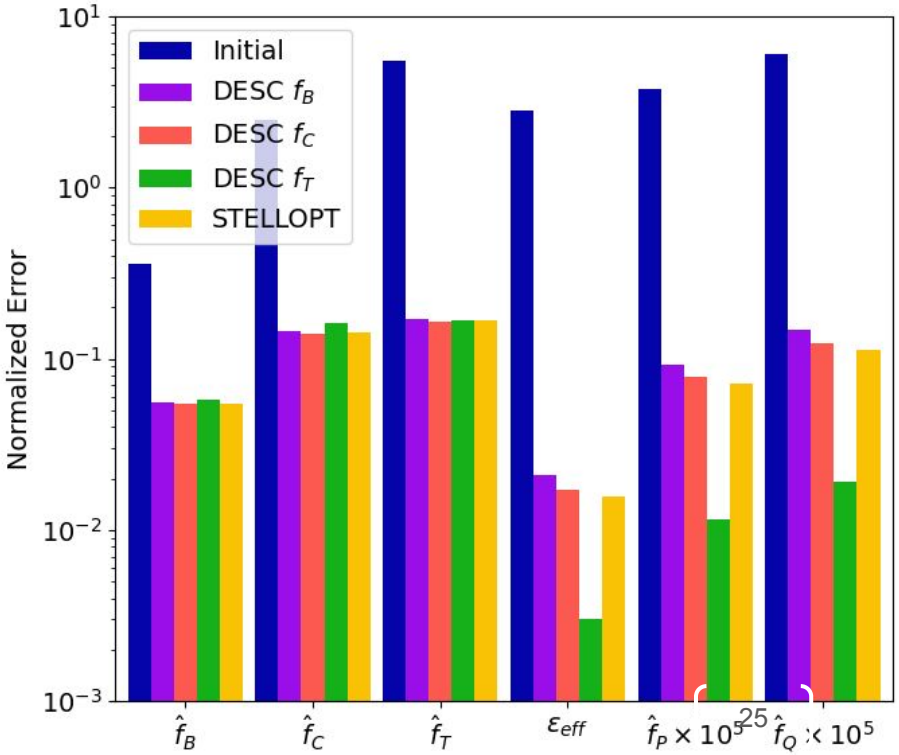
- Finite difference gradients scale with the number of optimization variables
- Automatic differentiation time is only dependent on spectral resolution

CPU/GPU time per iteration (80 variables)	
STELLOPT	~ 1.4 hrs
DESC (CPU)	~ 1 min
DESC (GPU)	~ 8 sec



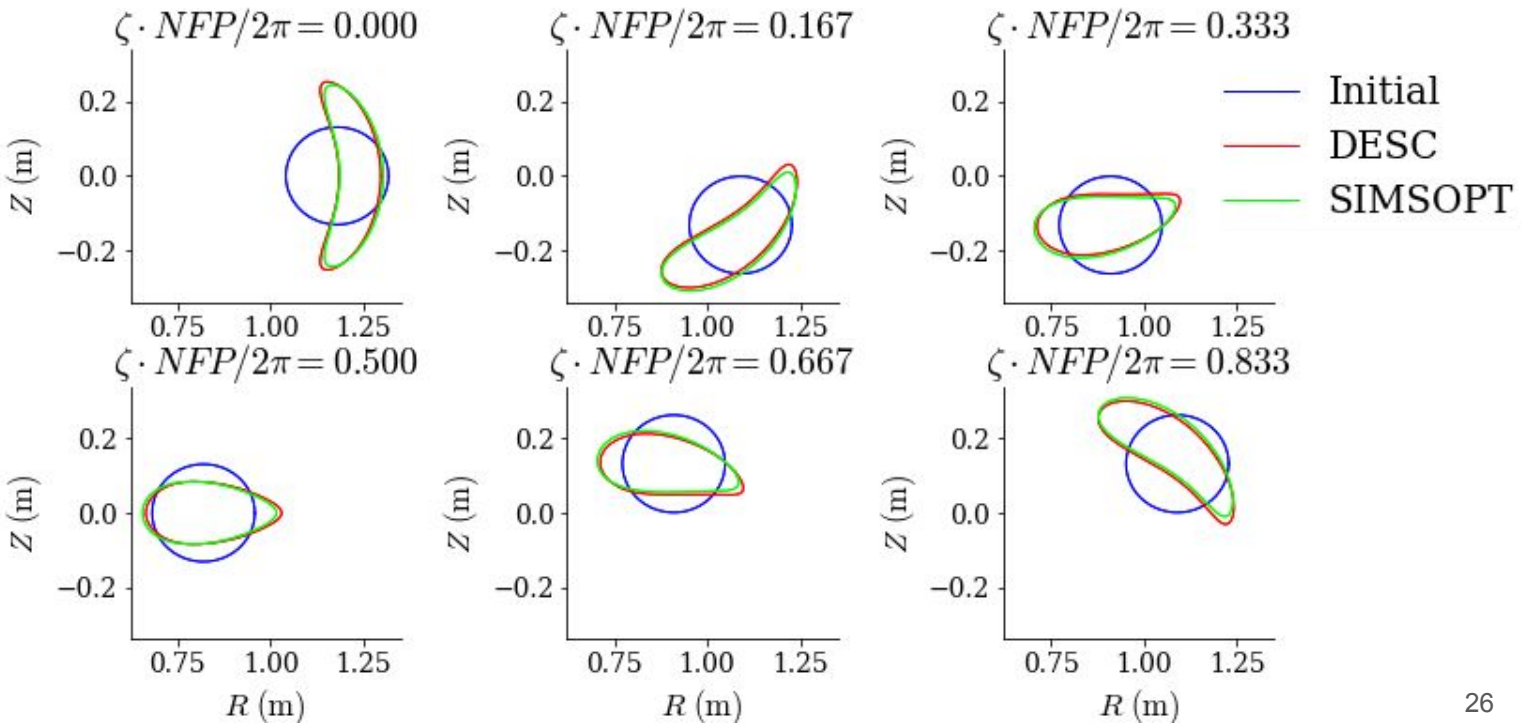
QS has a complex correlation with transport

- Neoclassical ripple ε_{eff} computed from NEO
- Particle & heat fluxes f_P & f_Q computed from SFINCS
- Similar levels of QS don't always mean similar levels of transport
- Triple product metric may give better results



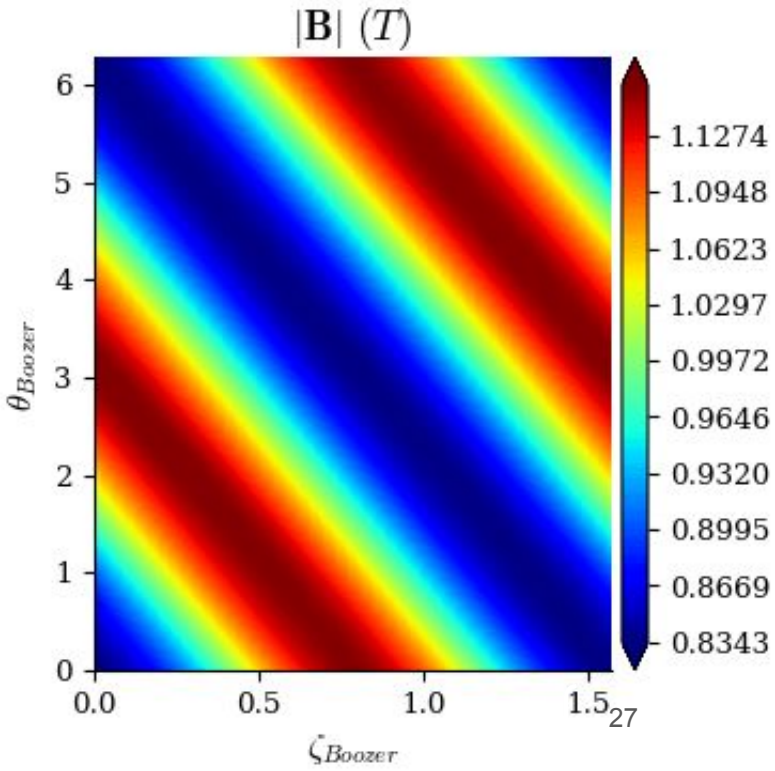
Reproduce “precise QH” SIMSOPT result

Example QH optimization adapted from Landreman and Paul, *Phys. Rev. Lett.* (2022)



Reproduce “precise QH” SIMSOPT result

- Achieved same order of magnitude of QS error
- Compute time:
  - SIMPSOPT (CPU): ~ 5 hrs
  - DESC (CPU) : 22 min
  - DESC (GPU): 7 min



Summary

✔ Equilibrium solver is more accurate than VMEC

✔ Optimization is faster than STELLOPT

✔ Automatic differentiation enables advanced analysis tools

✔ Easy to install and use

Software

- Open-source repository: `https://github.com/PlasmaControl/DESC`
- Python package: `pip install desc-opt`

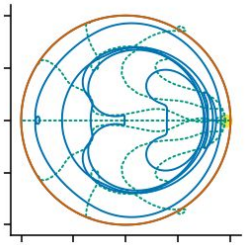
Papers

- The DESC Stellarator Code Suite Part I https://arxiv.org/abs/2203.17173
- The DESC Stellarator Code Suite Part II https://arxiv.org/abs/2203.15927
- The DESC Stellarator Code Suite Part III https://arxiv.org/abs/2204.00078

Backup

Trust region method regularizes perturbation step

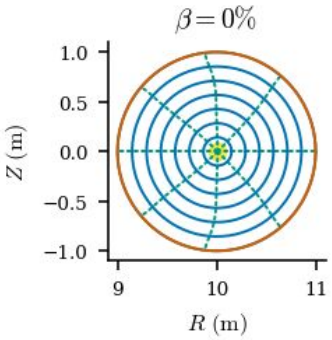
- When perturbations are large or curvature is strong unconstrained step can be too large, leading to non-physical results
  - Don't want exact solution to approximate model

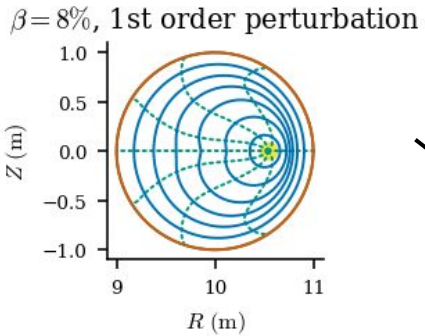


- Solution: constrain step size to lie within "trust region" in which we trust the Taylor approximation
  - $\min_{\Delta x} \left| \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \Delta x - b \right| \quad s.t. \ |\Delta x| < r$
- Subproblem solved using univariate Newton's method
  - Cost roughly one SVD of $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ or 2-3 Cholesky factorizations of $\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$
- Shrinking trust region for higher order expansion also allows enforcing of assumed asymptotic scaling
  - $|\Delta x_2| \ll |\Delta x_1|$

2nd order perturbations significantly reduce number of needed iterations

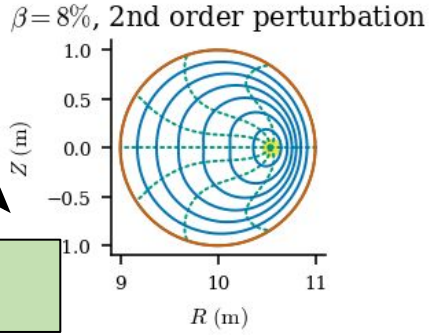
```
eq1 = eq.perturb(dp=delta_p,
                 order=1)
```

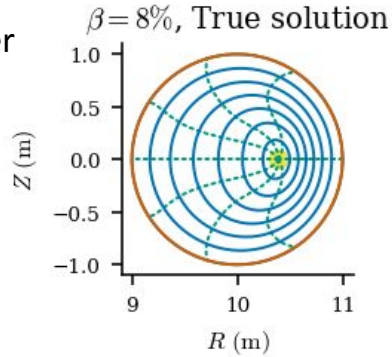





```
eq1.solve(ftol=1e-2)
```

59 optimizer steps





30 optimizer steps

```
eq2 = eq.perturb(dp=delta_p,
                 order=2)
```

```
eq2.solve(ftol=1e-2)
```

Continuation methods enable exploration of the stellarator phase space

- Once a single equilibrium solution is found, varying an input parameter reveals a family of solutions
  - Pressure scans, boundary perturbations, etc.
- Provides an efficient way to navigate the solution landscape
- Exposes the connection between discrete equilibrium solutions

(Hirshman and Whitson, 1983)

VMEC – Theory

$$W = \int_V \left(\frac{B^2}{2\mu_0} + \frac{p}{\gamma - 1} dV \right)$$

$$X_j = \{R, \lambda, Z\}, j = 1,2,3$$

$$X_j = \sum_{m,n} X_j^{mn} \mathrm{e}^{(i(mu - nv))}$$

First variation, with t as variational parameter

$$\frac{dW}{dt} = \int_V (F_j^{mn})^* \frac{\partial X_j^{mn}}{\partial t} dV$$

Steepest Descent direction: change X_j^{mn} until $\frac{dW}{dt} = 0$ i.e. stationary point is reached

$$\frac{\partial X_j^{mn}}{\partial t} = F_j^{mn}$$

$$\frac{\partial^2 X_j^{mn}}{\partial t^2} + \frac{1}{\tau} \frac{\partial X_j^{mn}}{\partial t} = F_j^{mn}$$

Change to 2nd order for better convergence

VMEC Algorithm

(Hirshman and Whitson, 1983)

Initialization

Inputs

$R_b(s=1,u,v)$,
$Z_b(s=1,u,v)$,
$p(s), \iota(s), \psi_a$

Fourier Series

$R_{b,mn}, Z_{b,mn}$

Scale Boundary as Initial Guess for Surface Geometry

$R_{mn}(s) \sim s\, R_{b,mn}$

$Z_{mn}(s) \sim s\, Z_{b,mn}$

Main Algorithm

Repeat until Desired Resolution

Compute **B** and necessary derivatives from $R(s,u,v), Z(s,u,v)$

Compute F_j^{mn}

$$\frac{\partial^2 X_j^{mn}}{\partial t^2} + \frac{1}{\tau}\frac{\partial X_j^{mn}}{\partial t} = F_j^{mn}$$

Repeat until converged

Interpolate solution onto a finer radial mesh

$$R(s,u,v) = \sum_{m=0,n=-N}^{M,N} R_{mn,c}(s)cos(mu - nvN_{FP}) + R_{mn,s}(s)sin(mu - nvN_{FP})$$

$$\lambda(s,u,v) = \sum_{m=0,n=-N}^{M,N} \lambda_{mn,c}(s)cos(mu - nvN_{FP}) + \lambda_{mn,s}(s)sin(mu - nvN_{FP})$$

$$Z(s,u,v) = \sum_{m=0,n=-N}^{M,N} Z_{mn,c}(s)cos(mu - nvN_{FP}) + Z_{mn,s}(s)sin(mu - nvN_{FP})$$

What DESC Solves

Inputs: $\mathrm{R_b}(\theta,\zeta), \mathrm{Z_b}(\theta,\zeta), p(\rho), \iota(\rho)$

$$\mathbf{e}_\rho = \begin{bmatrix} \partial_\rho R \\ 0 \\ \partial_\rho Z \end{bmatrix} \quad \mathbf{e}_\theta = \begin{bmatrix} \partial_\theta R \\ 0 \\ \partial_\theta Z \end{bmatrix} \quad \mathbf{e}_\zeta = \begin{bmatrix} \partial_\zeta R \\ R \\ \partial_\zeta Z \end{bmatrix}$$

$$\sqrt{g} = \mathbf{e}_\rho \cdot \mathbf{e}_\theta \times \mathbf{e}_\zeta$$

$$B^\theta = \frac{\psi'}{\sqrt{g}}\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)$$

$$B^\zeta = \frac{1}{\sqrt{g}}\psi'\left(1 + \frac{\partial \lambda}{\partial \theta}\right)$$

$$J^\rho = \frac{1}{\mu_0\sqrt{g}}\left(\frac{\partial B_\zeta}{\partial \theta} - \frac{\partial B_\theta}{\partial \zeta}\right)$$

$$J^\theta = \frac{1}{\mu_0\sqrt{g}}\left(\frac{\partial B_\rho}{\partial \zeta} - \frac{\partial B_\zeta}{\partial \rho}\right)$$

$$J^\zeta = \frac{1}{\mu_0\sqrt{g}}\left(\frac{\partial B_\theta}{\partial \rho} - \frac{\partial B_\rho}{\partial \theta}\right)$$

$\mathbf{B}(\rho,\theta,\zeta), \mathbf{J}(\rho,\theta,\zeta)$

$$F_\rho = \sqrt{g}(J^\zeta B^\theta - J^\theta B^\zeta) + p'$$

$$F_\beta = \sqrt{g} J^\rho$$

$\mathbf{F}(\rho,\theta,\zeta)$

R, Z, λ and their derivatives are evaluated on a collocation grid in (ρ,θ,ζ) , then multiplied to calculate **F** on this grid

This leads to a system of equations comprised of the force balance error evaluated at the collocation nodes, which we want to make equal to zero -> Can use root-finding or least-squares to solve

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$$

x is the spectral coefficients of *R,Z,λ*, which is what we are changing to minimize **f**

DESC Algorithm

Initialization

Inputs

$R_b(\rho = 1, \theta, \zeta)$,
$Z_b(\rho = 1, \theta, \zeta)$,
$p(\rho), \iota(\rho), \psi_a$

Fourier Series $R_{b,mn}, Z_{b,mn}$

Scale Boundary as Initial Guess for Surface Geometry

$R_{mn}(\rho) \sim \rho R_{b,mn}$
$Z_{mn}(\rho) \sim \rho Z_{b,mn}$

Main Algorithm

Repeat until Desired Resolution

Compute **B** and **J** from $R(\rho, \theta, \zeta), Z(\rho, \theta, \zeta), \lambda(\rho, \theta, \zeta)$

Compute F_ρ, F_β on collocation grid

$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$, use Newton Method to find $\Delta \boldsymbol{x}$

Repeat until converged

Increase Collocation Grid and/or Spectral Resolution

$$R(\rho, \theta, \zeta) = \sum_{m=-M, n=-N, l=0}^{M,N,L} R_{lmn} \mathcal{Z}_l^m(\rho, \theta) \mathcal{F}^n(\zeta)$$

$$\lambda(\rho, \theta, \zeta) = \sum_{m=-M, n=-N, l=0}^{M,N,L} \lambda_{lmn} \mathcal{Z}_l^m(\rho, \theta) \mathcal{F}^n(\zeta)$$

$$Z(\rho, \theta, \zeta) = \sum_{m=-M, n=-N, l=0}^{M,N,L} Z_{lmn} \mathcal{Z}_l^m(\rho, \theta) \mathcal{F}^n(\zeta)$$

$$\boldsymbol{x} = [R_{lmn}, Z_{lmn}, \lambda_{lmn}]$$

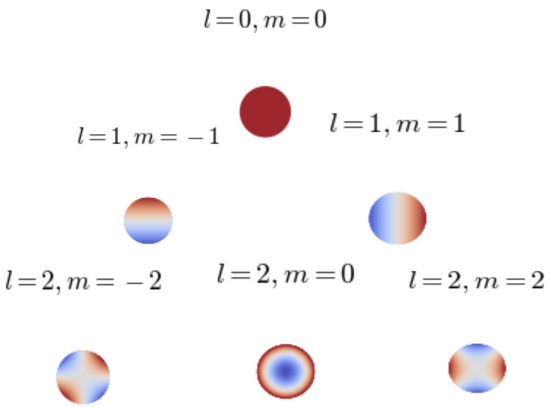
DESC - Fourier-Zernike Spectral Basis

$$R(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} R_{lmn}\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$

$$\lambda(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} \lambda_{lmn}\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$

$$Z(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} Z_{lmn}\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$

Zernike Basis in (*r,θ*)



$$\mathcal{Z}_m^l(\rho,\theta) = \begin{cases} \mathcal{R}_l^{|m|}(\rho)cos(|m|\theta) & \text{for } m \geq 0 \\ \mathcal{R}_l^{|m|}(\rho)sin(|m|\theta) & \text{for } m < 0 \end{cases}$$

Zernike Polynomials

$$\boxed{\mathcal{R}_l^{|m|}(\rho) = \sum_{s=0}^{(l-|m|)/2} \frac{(-1)^s(l-s)!}{s!\,[(l+|m|)/2-s]!\,[(l-|m|)/2+s]!}\rho^{l-2s}}$$

Radial Polynomial is of degree *l-2s*, and *l=m,m+2,…, L*

$$\mathcal{F}^n(\zeta) = \begin{cases} cos(|n|N_{FP}\zeta)\text{for } n \geq 0 \\ sin(|n|N_{FP}\zeta)\text{for } n < 0 \end{cases}$$

Toroidal Fourier Series

[1] Lazerson et al. (2020)
[2] Paul, Landreman and Antonsen (2021)

Equilibrium Optimization with DESC Automatic Differentiation (AD)

- Once an equilibrium solution is found, it is desirable to **optimize** for some **objective**
  - Particle confinement, Quasisymmetry, etc.
- Inputs to an equilibrium are pressure profile, rotational transform profile, and boundary shape

Equilibrium Solution $\boldsymbol{x}^* = [R_{lmn}, Z_{lmn}, \lambda_{lmn}]$

Inputs $\boldsymbol{c} = [p, i, R_b, Z_b]$

- Want to optimize the objective **g(x*,c)** with respect to inputs, **c**
  - Requires knowing how the figure of merit changes with respect to inputs, **c**, i.e. $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{c}}$
  - **But g is not an explicit function of c**
  - **Instead an eqn (argmin of g)**
- **Challenge: How to get gradient of g with respect to c?**
  - Typically find derivatives with finite-differences[1], **requires many VMEC equilibrium solves -> expensive**
  - Analytic gradients have been formulated recently, but only for limited number of objectives[2]

DESC Automatic Differentiation with JAX gives **fast, accurate derivative information** for **arbitrary objectives**

Optimization Code	Computation Time
STELLOPT (8 CPUs)	~ 2 hours
STELLOPT (16 CPUs)	~ 1.5 hours
STELLOPT (32 CPUs)	~ 1 hour
DESC (1 CPU)	< 30 minutes
DESC (1 GPU)	< 10 minutes

Courtesy of Daniel Dudt

Plasma Model – Ideal MHD

Mass:	$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$
Momentum:	$\rho \frac{d\mathbf{v}}{dt} = \mathbf{J} \times \mathbf{B} - \nabla p$
Energy:	$\frac{d}{dt}\left(\frac{p}{\rho^{\gamma}}\right) = 0$
Ohm's law:	$\mathbf{E} + \mathbf{v} \times \mathbf{B} = 0$
Maxwell:	$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$
	$\nabla \times \mathbf{B} = \mu_0 \mathbf{J}$
	$\nabla \cdot \mathbf{B} = 0$

Simplest macroscopic plasma fluid model, assumes low-frequency, long wavelength, neglects e$^-$ inertia

$$\rho = m_i n_i$$

$$\mathbf{v} = \mathbf{u}_i$$

$$L \gg \lambda_D$$

$$n_e \approx n_i$$

(Freidberg 2014)

Plasma Model – Ideal MHD Derivation

Low-Frequency
Neglect e⁻ inertia
Long Wavelength

$$\omega \ll \omega_p e, \Omega_e$$

$$m_e \to 0$$

$$L \gg \lambda_D$$

Vlasov-Maxwell Equations for ions and electrons

$$\frac{df_\alpha}{dt} \equiv \frac{\partial f_\alpha}{\partial t} + \mathbf{v}\cdot\nabla f_\alpha + \frac{Z_\alpha e}{m_\alpha}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\cdot\nabla_v f_\alpha = \left(\frac{\partial f_\alpha}{\partial t}\right)_c$$

$$\nabla\times\mathbf{E} = -\frac{\partial\mathbf{B}}{\partial t} \qquad \nabla\cdot\mathbf{E} = \frac{\sigma}{\varepsilon_0}$$

$$\nabla\times\mathbf{B} = \mu_0\mathbf{J} + \frac{1}{c^2}\frac{\partial\mathbf{E}}{\partial t} \qquad \nabla\cdot\mathbf{B} = 0$$

Take 0th,1st,2nd moments

Implicit assumption: enough collisions so that fluid moments are accurate reflections (i.e. plasma is near Maxwellian)

Two-Fluid Equations

$$\left(\frac{dn_\alpha}{dt}\right)_\alpha + n_\alpha\nabla\cdot\mathbf{u}_\alpha = 0$$

$$m_\alpha n_\alpha\left(\frac{d\mathbf{u}_\alpha}{dt}\right)_\alpha - Z_\alpha e n_\alpha(\mathbf{E}+\mathbf{u}_\alpha\times\mathbf{B}) + \nabla\cdot\mathbf{P}_\alpha = \mathbf{R}_\alpha$$

$$n_\alpha\left[\frac{d}{dt}\left(\frac{1}{2}m_\alpha u_\alpha^2 + \frac{3}{2}T_\alpha\right)\right]_\alpha - Z_\alpha e n_\alpha \mathbf{u}_\alpha\cdot\mathbf{E} + \nabla\cdot(\mathbf{u}_\alpha\cdot\mathbf{P}_\alpha + \mathbf{q}_\alpha) = Q_\alpha + \mathbf{u}_\alpha\cdot\mathbf{R}_\alpha$$

(Freidberg 2014)

Single-Fluid Equations + Ohm's Law

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0$$

$$\rho\frac{d\mathbf{v}}{dt} - \mathbf{J}\times\mathbf{B} + \nabla p = -\nabla\cdot(\mathbf{\Pi}_i + \mathbf{\Pi}_e)$$

$$\frac{d}{dt}\left(\frac{p_i}{\rho^\gamma}\right) = \frac{2}{3\rho^\gamma}(Q_i - \nabla\cdot\mathbf{q}_i - \mathbf{\Pi}_i : \nabla\mathbf{v})$$

$$\frac{d}{dt}\left(\frac{p_e}{\rho^\gamma}\right) = \frac{2}{3\rho^\gamma}\left[Q_e - \nabla\cdot\mathbf{q}_e - \mathbf{\Pi}_e : \nabla\left(\mathbf{v} - \frac{\mathbf{J}}{en}\right)\right] + \frac{1}{en}\mathbf{J}\cdot\nabla\left(\frac{p_e}{\rho^\gamma}\right)$$

$$\mathbf{E} + \mathbf{v}\times\mathbf{B} = \frac{1}{en}(\mathbf{J}\times\mathbf{B} - \nabla p_e - \nabla\cdot\mathbf{\Pi}_e + \mathbf{R}_e)$$

Collision dominated -
L >> ion gyroradius -
Resistivity is negligible -
e⁻ -ion equil time small -

Viscosity negligible, neglect anisotropic pressure
Simplify Ohm's Law $\boldsymbol{J}\times\boldsymbol{B}, \nabla p_e$
$\boldsymbol{R_e} \sim \eta\boldsymbol{J} \to 0$ (high collisionality)
$p_e \sim p_i$, simplifies energy eqns

Ideal MHD

Mass: $\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0$

Momentum: $\rho\frac{d\mathbf{v}}{dt} = \mathbf{J}\times\mathbf{B} - \nabla p$

Energy: $\frac{d}{dt}\left(\frac{p}{\rho^\gamma}\right) = 0$

Ohm's law: $\mathbf{E} + \mathbf{v}\times\mathbf{B} = 0$

Maxwell: $\nabla\times\mathbf{E} = -\frac{\partial\mathbf{B}}{\partial t}$

$\nabla\times\mathbf{B} = \mu_0\mathbf{J}$

$\nabla\cdot\mathbf{B} = 0$

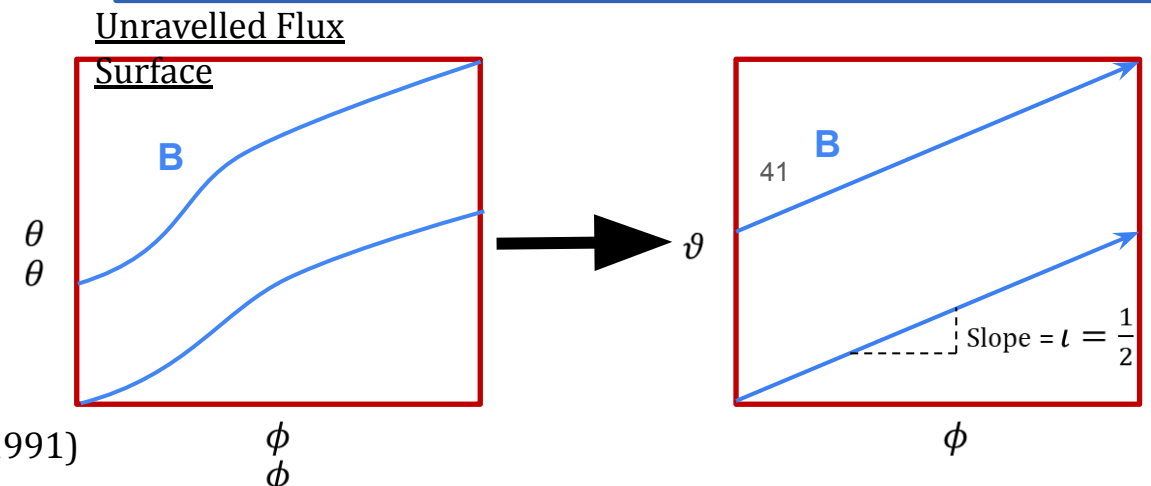
Straight-Field-Line (SFL) Coordinates

Assumes **B** lies on Nested Flux Surfaces, so
$\boldsymbol{B} \cdot \nabla\rho = 0 \rightarrow \boldsymbol{B} \cdot \hat{\boldsymbol{n}} = 0$

- Magnetic Field on a flux surface appears **straight** in special, 'Straight-Field-Line' coordinates, $(u^1, u^2, u^3) = (\rho, \vartheta, \phi)$
  - Curvilinear, non-orthogonal
  - Especially simple form for **B:**

$$\boldsymbol{B} = -\iota\nabla\rho \times \nabla\phi + \nabla\rho \times \nabla\vartheta = \frac{1}{\sqrt{g}}(\iota\boldsymbol{e}_{\vartheta} + \boldsymbol{e}_{\phi})$$

	Flux Surface Label
	Poloidal Angle
	SFL Poloidal Angle
	Geometric Toroidal Angle



D'haeseleer et al. *Flux Coordinates* (1991)

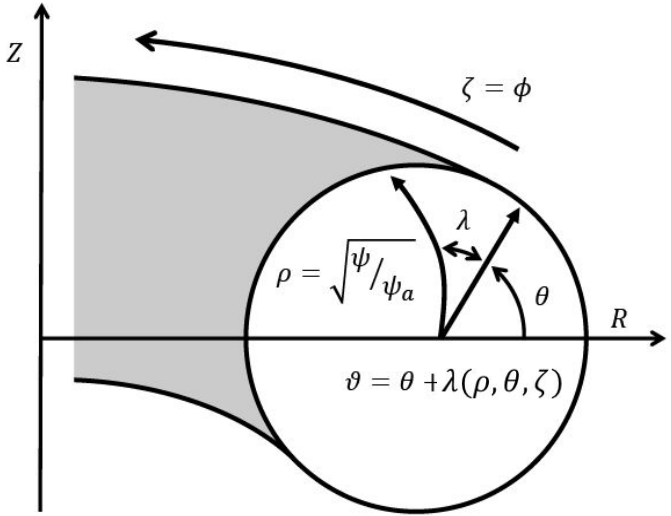
Computational Coordinate System

Straight-Field-Line/Magnetic (ρ, ϑ, ζ)

- **Curvilinear, non-orthogonal**
- **Magnetic Field is Straight**
- **But, must know magnetic field already!**

Computational Coordinates (ρ, θ, ζ)

- **Curvilinear, non-orthogonal**
- **Magnetic Field is NOT Straight**
- $\vartheta = \theta + \lambda(\rho, \theta, \zeta)$



$$\mathbf{e}_\rho = \begin{bmatrix} \partial_\rho R \\ 0 \\ \partial_\rho Z \end{bmatrix} \quad \mathbf{e}_\theta = \begin{bmatrix} \partial_\theta R \\ 0 \\ \partial_\theta Z \end{bmatrix} \quad \mathbf{e}_\zeta = \begin{bmatrix} \partial_\zeta R \\ R \\ \partial_\zeta Z \end{bmatrix}$$

$$\mathbf{B} = \frac{1}{\sqrt{g}}\left(\iota - \frac{\partial \lambda}{\partial \zeta}\right)\mathbf{e}_\theta + \frac{1}{\sqrt[4]{g}}\left(1 + \frac{\partial \lambda}{\partial \theta}\right)\mathbf{e}_\zeta$$

Must include λ as an **unknown** now, which is the function which when added to θ makes it an SFL angle ϑ

(Hirshman and Whitson, 1983)

VMEC - Coordinate System

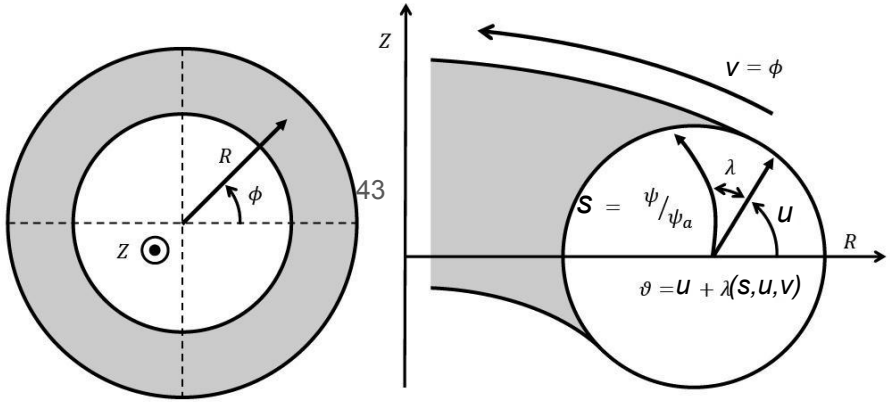
	Flux Surface Label
	Poloidal Angle
	SFL Poloidal Angle
	Geometric Toroidal Angle

Geometry represented as **Fourier Series** in (u, v) **on each discrete surface**

$$R(s,u,v) = \sum_{m=0,n=-N}^{M,N} R_{mn,c}(s)cos(mu - nvN_{FP}) + R_{mn,s}(s)sin(mu - nvN_{FP})$$

$$\lambda(s,u,v) = \sum_{m=0,n=-N}^{M,N} \lambda_{mn,c}(s)cos(mu - nvN_{FP}) + \lambda_{mn,s}(s)sin(mu - nvN_{FP})$$

$$Z(s,u,v) = \sum_{m=0,n=-N}^{M,N} Z_{mn,c}(s)cos(mu - nvN_{FP}) + Z_{mn,s}(s)sin(mu - nvN_{FP})$$



DESC - Coordinate System

(Dudt and Kolemen 2020)

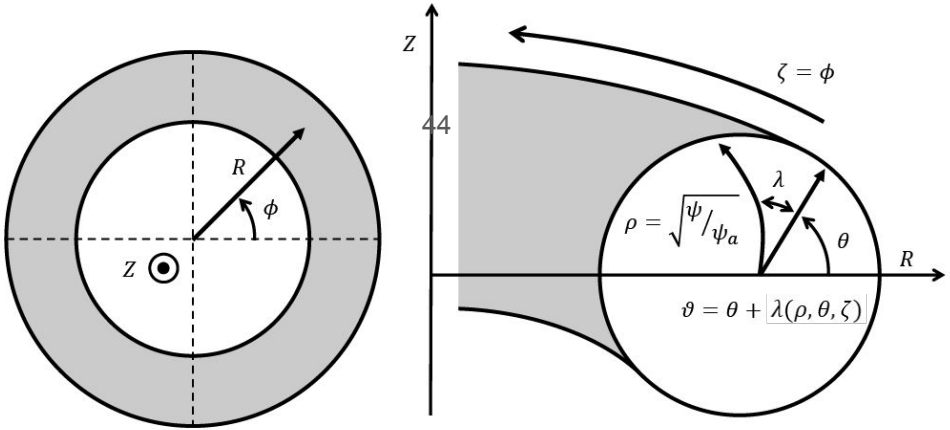
	Flux Surface Label
	Poloidal Angle
	SFL Poloidal Angle
	Geometric Toroidal Angle

Geometry represented **continuously** with global basis functions

$$R(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} R_{lmn}\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$

$$\lambda(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} \lambda_{lmn}\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$

$$Z(\rho,\theta,\zeta) = \sum_{m=-M,n=-N,l=0}^{M,N,L} Z_{lmn}\mathcal{Z}_l^m(\rho,\theta)\mathcal{F}^n(\zeta)$$



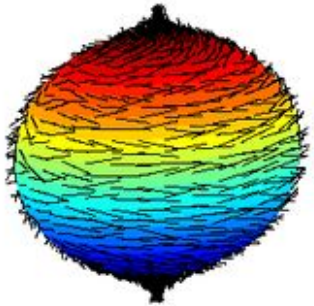
Poincare-Hopf Theorem : Mathematical Reason for Toroidal Geometry

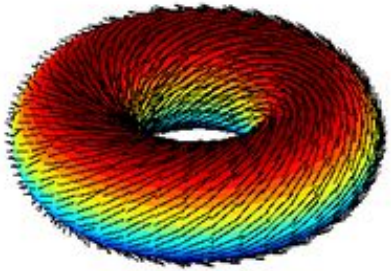
$$\boldsymbol{J} \times \boldsymbol{B} = \nabla p \longrightarrow \mathbf{B} \cdot \nabla p = 0$$

- No non-vanishing **tangent vector fields** on **even-dimensional n-spheres** (i.e. cannot have a vector field tangent everywhere to a sphere in 3D while being continuous and non-zero)
- So a sphere in 3D cannot be a flux surface (remember that a 2-sphere is the normal sphere in 3D we think of)

Cannot comb hair flat on a coconut

- ealiz : allows this is a **torus**

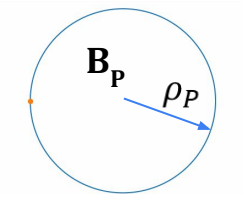




Source: https://en.wikipedia.org/wiki/Hairy_ball_theorem

Axisymmetry and Noether's Theorem

$\rho_P = \frac{mv_t}{qB_P}$: Poloidal Gyroradius



- Axisymmetry guarantees particles confinement (in absence of collis
- Consider single particle Lagrangian, axisy $\frac{\partial L}{\partial \phi} = 0 \rightarrow \frac{\partial L}{\partial \dot{\phi}} = p_\phi = const$

$$L = \frac{m}{2}(\dot{R}^2 + R^2\dot{\phi}^2 + \dot{Z}^2) - q\Phi_E + (A_R\dot{R} + A_\phi R\dot{\phi} + A_Z\dot{Z})$$

$$p_\phi := \partial L/\partial\dot{\phi} = mR^2\dot{\phi} + qRA_\phi$$

$$\frac{mR^2\dot{\phi}}{qRA_\phi} \sim \frac{mv_t}{qB_P R} \sim \frac{\rho_P}{R} \ll 1$$

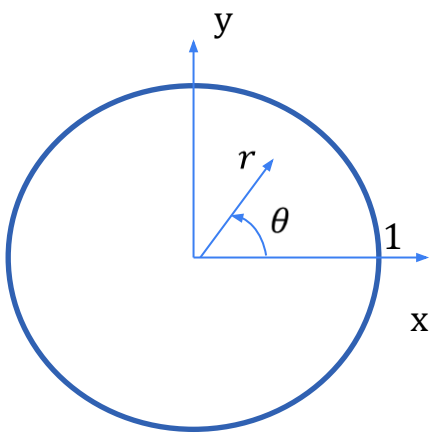
$$p_\phi \approx RA_\phi = constant$$

For Strongly Magnetized Plasmas (true in fusion devices)

RA_ϕ **ux function** (i.e. is constant on a magnetic surface), so this means that a particle will stay within a small distance of the flux surface it is on -> **particle is confined to the surface!**

[1](Lewis and Bellan, 1990)

Analyticity Constraint at Polar Axis Proof



- Assume $f(r,\theta)$ is a physical scalar, regular at r=0
- Expand in a Fourier Series: $\sum_{m=-\infty}^{\infty} a_{m(r)} e^{im\theta} = \sum_{-\infty}^{\infty} f_m(r,\theta)$
  - Where the Fourier coefficients are a function of polar radius *r*
- Assume each $f_m(r,\theta)$ is a regular function of *(x,y)* at r=0
- Notice that $e^{im\theta}$ is NOT regular at r=0 (it is multi-valued)
- But, $\left[re^{\pm im\theta}\right]^{|m|} = [x \pm iy]^{|m|}$ is a regular function of (x,y) b/c it is a <u>polynomial</u> in (x,y)
- We can rewrite $f(r,\theta)$ as

$$
\begin{aligned}
f_m(r,\theta) &= a_m(r) e^{im\theta} \\
&= \frac{a_m(r)}{r^{|m|}} r^{|m|} e^{i\theta m} \\
&= \frac{a_m(r)}{r^{|m|}} \left[re^{\pm i\theta}\right]^{|m|} \begin{cases} + & m > 0 \\ - & m < 0 \end{cases}
\end{aligned}
$$

Regular at r=0

$$\boxed{f_m(r,\theta)} = \boxed{\frac{a_m(r)}{r^{|m|}}} \boxed{[x \pm iy]^{|m|}}$$

Must be regular at r=0!

Regular at r=0

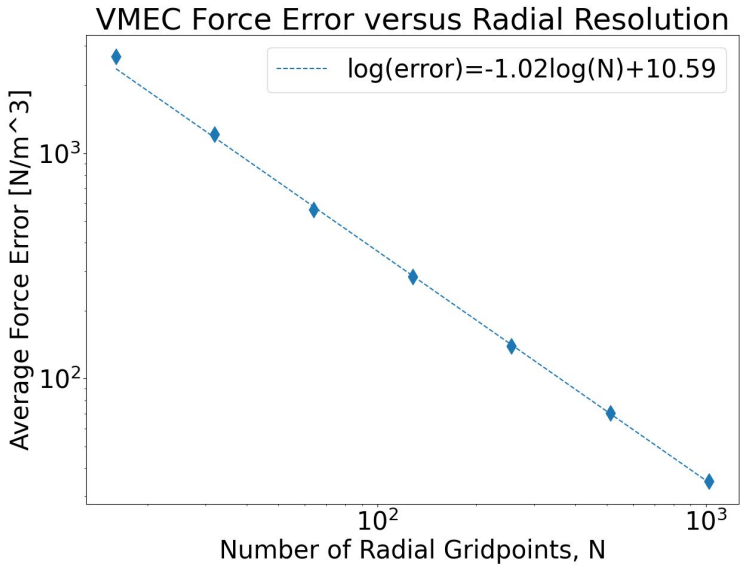
$$\boxed{\lim_{r\to 0} \frac{a_m(r)}{r^{|m|}} < \infty}$$

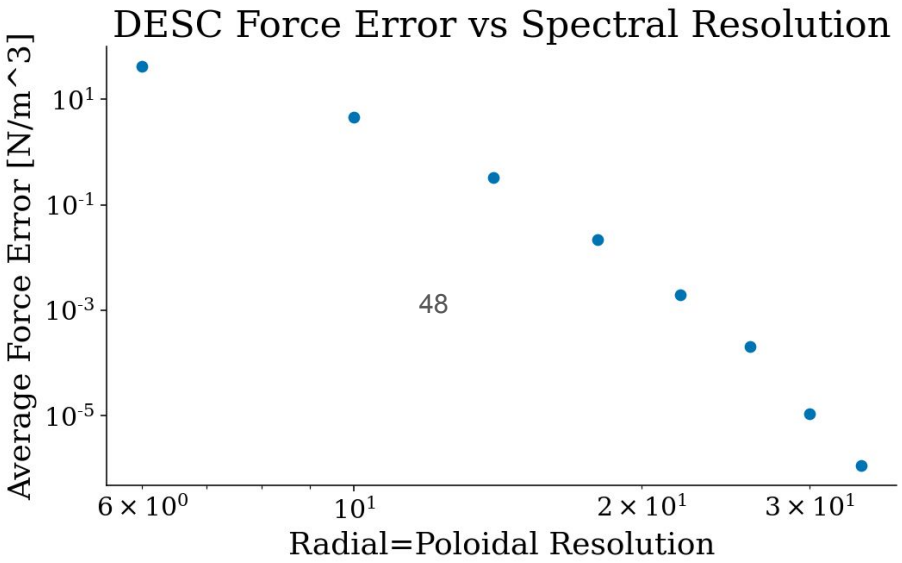
$a_m(r)$ must scale **at least** as $r^{|m|}$

$a_m(r) \sim r^{|m|} + r^{|m|+2} \ldots$

DESC compares well to VMEC – Convergence on log-log scale

	Angular Convergence	Radial Convergence
DESC	Exponential	Exponential (coupled to Poloidal)
VMEC	Exponential	





Automatic Differentiation

- Essentially, is the **chain rule**

$$
\begin{aligned}
y &= f(g(h(x))) = f(g(h(w_0))) = f(g(w_1)) = f(w_2) = w_3 \\
w_0 &= x \\
w_1 &= h(w_0) \\
w_2 &= g(w_1) \\
w_3 &= f(w_2) = y
\end{aligned}
$$

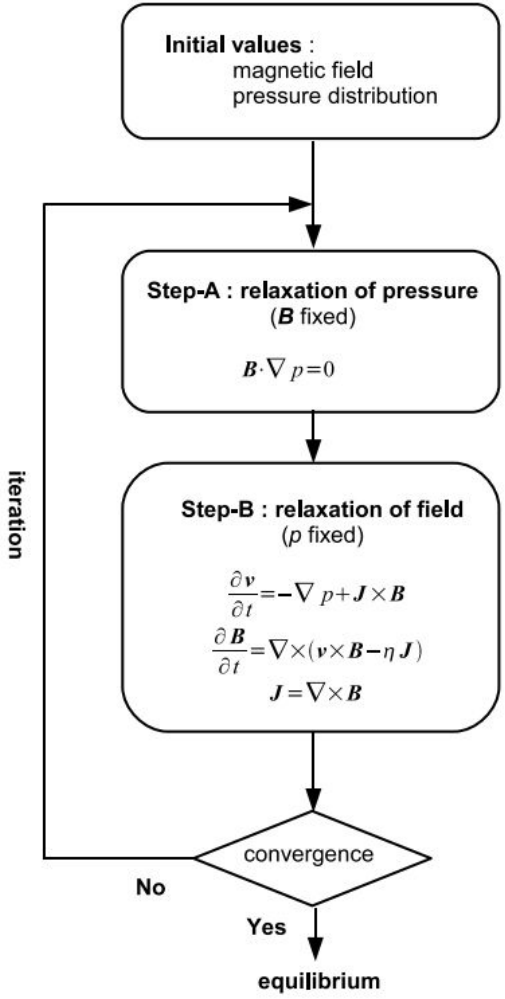
the chain rule gives

$$
\frac{dy}{dx} = \frac{dy}{dw_2}\frac{dw_2}{dw_1}\frac{dw_1}{dx} = \frac{df(w_2)}{dw_2}\frac{dg(w_1)}{dw_1}\frac{dh(w_0)}{dx}
$$

https://en.wikipedia.org/wiki/Automatic_differentiation

(Suzuki et al. 2006)

HINT2 Code



Relax pressure parallel to **B**

Fix **p** and let fields evolve until stationary state reached $\frac{\partial}{\partial t} \approx 0$

PIES Code

Reiman et al. *J. Comp. Phys.* (1986)

- Starting from an initial field, update pressure:

$$\boldsymbol{B} \cdot \boldsymbol{\nabla} p = 0$$

Update diamagnetic & Pfirsch-Schlüter currents:

$$\boldsymbol{J}_{\perp} = \boldsymbol{B} \times \boldsymbol{\nabla} p / B^2 \quad \boldsymbol{B} \cdot \boldsymbol{\nabla} \left(\frac{J_{\parallel}}{B} \right) = -\boldsymbol{\nabla} \cdot \boldsymbol{J}_{\perp}$$

Update magnetic field from Ampere's Law:

$$\boldsymbol{\nabla} \times \boldsymbol{B} = \mu_0 \boldsymbol{J}$$

Iterate until a steady state is reached

(Hudson et al. 2012)

SPEC Code

- Assume the plasma consists of nested volumes of stochasticity, separated by strongly irrational flux surfaces

Solve for $\boldsymbol{B}$ in each volume where $\boldsymbol{\nabla} p = \boldsymbol{0}$:

$$\boldsymbol{\nabla} \times \boldsymbol{B} = \mu_\alpha \boldsymbol{B}$$

Determine the positions of the flux surfaces by ensuring that the total pressure is continuous across each interface:

$$\left[\!\left[\frac{|\boldsymbol{B}|^2}{2\mu_0} + \frac{p}{\gamma - 1} \right]\!\right] = 0$$

DESC vs STELLOPT QS Optimization

- STELLOPT
  - Finite-difference derivatives
  - Parallelized across CPUs
  - Speed depends strongly on resolution
- DESC
  - First iteration takes longest (JIT compilation), then additional iterations are very fast
  - Resolution limited by GPU memory
  - Speed is independent of number of optimization variables

Optimization Code	Computation Time
STELLOPT (8 CPUs)	~ 2 hours
STELLOPT (16 CPUs)	~ 1.5 hours
STELLOPT (32 CPUs)	~ 1 hour
DESC (1 CPU)	< 30 minutes
DESC (1 GPU)	< 10 minutes

MPOL = NTOR = 8, NS = 65
FTOL = 1E-12
48 optimization variables
Optimize QS on all surfaces

Courtesy of Daniel Dudt

Rotational Transform

- Rotational Transform works by moving particle **poloidally** each time it transits **toroidally**
  - Vertical drift is in same direction, so effectively averages out the vertical drift to zero
- Rotational Transform = $\iota = \frac{\#\ of\ Poloidal\ Turns}{\#\ of\ Toroidal\ Turns}$
- Rotational Transform means there is a **toroidal** and a **poloidal B** field
- How to create this?
  - By driving a **toroidal current** through the plasma -> **Tokamak**
  - By twisting external coils -> **Stellarator**