Updates and improvements to the DESC code for finding and optimizing stellarator equilibria

Rory Conlin

With Daniel Dudt & Dario Panici

Princeton Plasma Control Group

Advisor: Egemen Kolemen

PPPL Stellarator Seminar

May 7, 2021

Why is *another* equilibrium code needed?

<u>Understand the solution space of stellarator equilibria</u>

- **Existing equilibrium codes only find discrete solutions**
- What does the phase space look like?
- Where are the subspaces for quasi-symmetry and the connection to tokamaks?
- How are different equilibria related to each other?

<u>Integrate stellarator optimization tools into equilibrium solver</u>

- **Existing approaches use finite differences or adjoint methods**
- Can we perform computations faster and more accurately?
- How should an equilibrium solver interact with a broader optimization code?

What would be the ideal stellarator optimization suite?

- Incorporate derivative information (sensitivity of solution to input parameters)
- Incorporate optimization objectives (quasi-symmetry, coil complexity, etc.)
- Modular design: easy to add new functionality, interface with other codes
- Eliminate the need for Jacobian approximations, near-axis expansions[1], low-β expansions, etc.

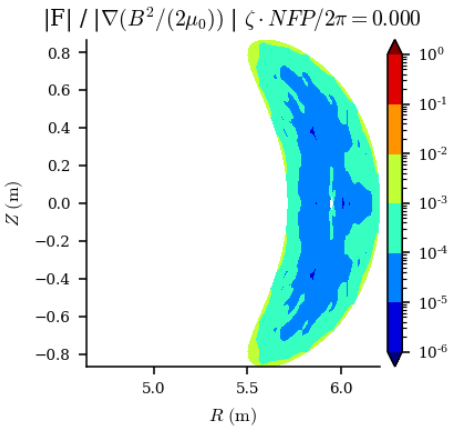
DESC is a modular pseudo-spectral equilibrium solver with automatic differentiation designed for stellarator optimization

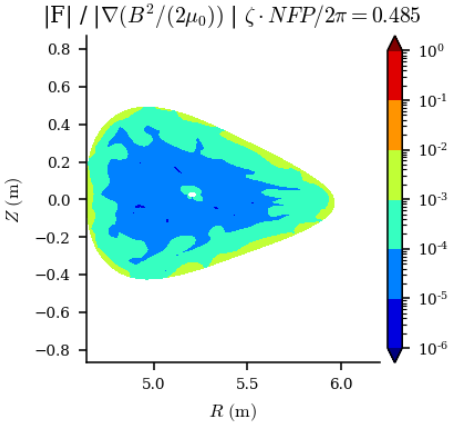
[1]Plunk et. al., *J. Plasma Phys.*, (2019).

What can DESC do?

High resolution, low error equilibria

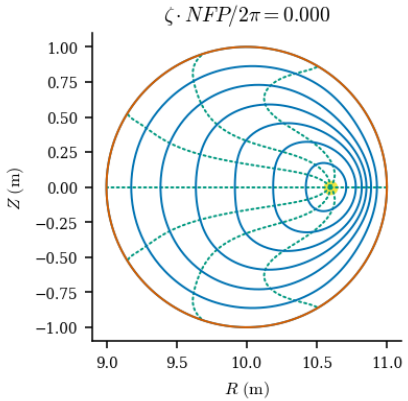
```
# solving an equilibrium
eq = Equilibrium(inputs)
eq.solve()
```



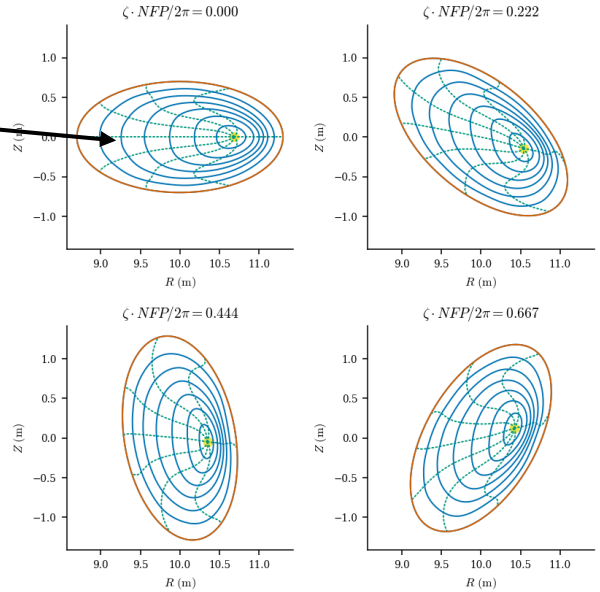


Transform tokamaks into stellarators

```
eq.perturb(dRb=delta_R_bdry,
           dZb=delta_Z_bdry,
           order=2)
```



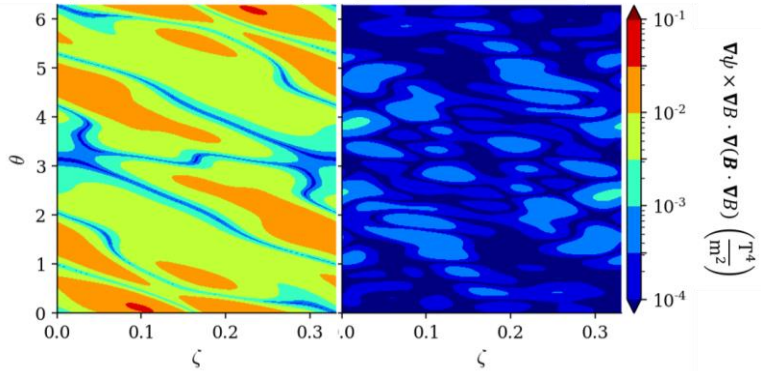
1 step



Optimize Quasi-symmetry

```
eq.perturb(objective=QS_fun,
           dRb=R_bdry_modes,
           dZb=Z_bdry_modes)
```

Before After



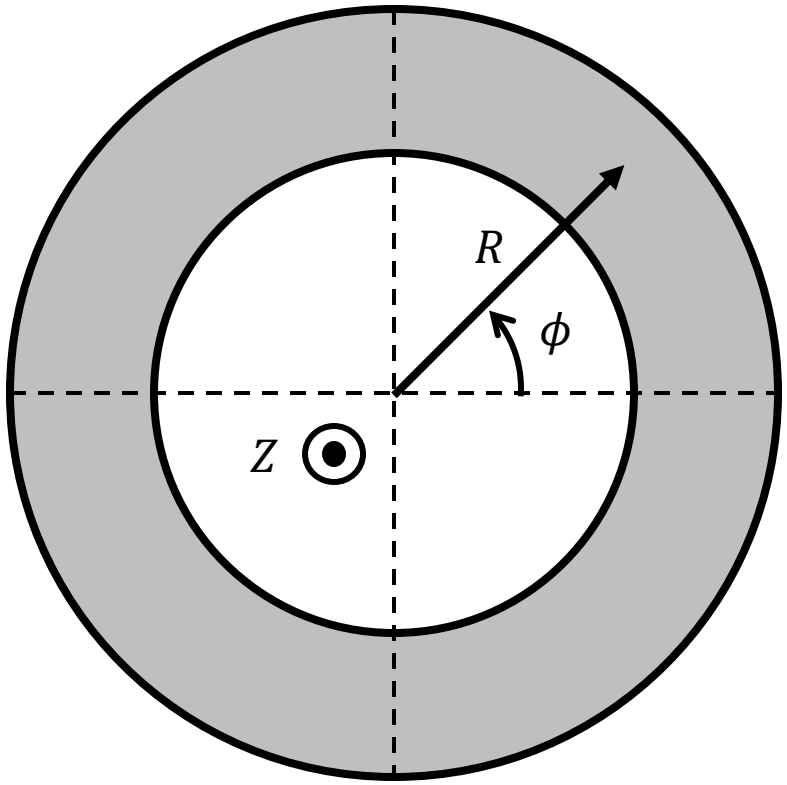
And more...

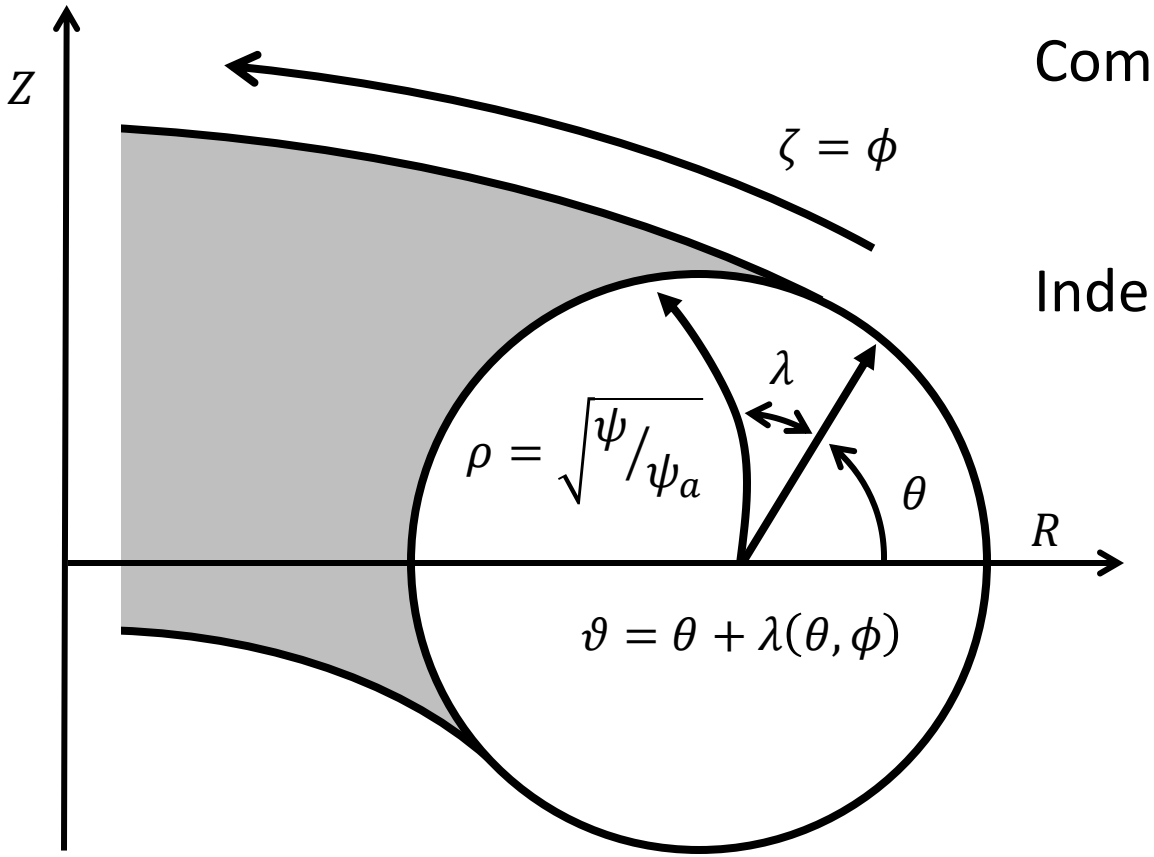
DESC makes similar assumptions as VMEC

- Ideal MHD
- Nested flux surfaces
- Fixed-boundary given by the inputs: $R^b(\theta, \phi), Z^b(\theta, \phi)$
- Free functions and scale given by the inputs: $p(\rho), \iota(\rho), \psi_a$

- Flexible design allows for many different input options
  - Showing the setup like VMEC for convenience
  - Input/Output compatibility with VMEC

Solving the "inverse" equilibrium problem





Computational domain:
(ρ, θ, ζ)

Independent variables:
$R(\rho, \theta, \zeta)$
$Z(\rho, \theta, \zeta)$
$\lambda(\rho, \theta, \zeta)$

toroidal coordinates: (R, ϕ, Z)

straight field-line coordinates[1]: (ρ, ϑ, ζ)

[1]Grimm et. al., *Methods in Computational Physics: Advances in Research and Applications*, (1976).

Magnetic field is written in flux coordinates

- Assume nested flux surfaces: $\boldsymbol{B} \cdot \boldsymbol{\nabla}\rho = 0$

$$\boldsymbol{B} = B^{\theta}\boldsymbol{e}_{\theta} + B^{\zeta}\boldsymbol{e}_{\zeta}$$

- Include the constraint of Gauss's law[1]: $\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$

$$\boldsymbol{B} = \frac{\psi'}{2\pi\sqrt{g}}\Big(\big(\iota - \partial_{\zeta}\lambda\big)\boldsymbol{e}_{\theta} + (1 + \partial_{\theta}\lambda)\boldsymbol{e}_{\zeta}\Big)$$

$$\boldsymbol{B}(\rho,\theta,\zeta) = \boldsymbol{B}(R(\rho,\theta,\zeta), Z(\rho,\theta,\zeta), \lambda(\rho,\theta,\zeta), \iota(\rho))$$

- Use Ampere's Law: $\boldsymbol{\nabla} \times \boldsymbol{B} = \mu_0 \boldsymbol{J}$

$$\boldsymbol{J}(\rho,\theta,\zeta) = \boldsymbol{J}(R(\rho,\theta,\zeta), Z(\rho,\theta,\zeta), \lambda(\rho,\theta,\zeta), \iota(\rho))$$

$$\boldsymbol{e}_{\rho} = \begin{bmatrix} \partial_{\rho}R \\ 0 \\ \partial_{\rho}Z \end{bmatrix}$$

$$\boldsymbol{e}_{\theta} = \begin{bmatrix} \partial_{\theta}R \\ 0 \\ \partial_{\theta}Z \end{bmatrix}$$

$$\boldsymbol{e}_{\zeta} = \begin{bmatrix} \partial_{\zeta}R \\ R \\ \partial_{\zeta}Z \end{bmatrix}$$

[1]D'haeseleer et. al., *Flux Coordinates and Magnetic Field Structure*, (1991).

Equilibrium is obtained when the force balance residuals vanish

$$\boldsymbol{F} \equiv \boldsymbol{J} \times \boldsymbol{B} - \boldsymbol{\nabla} p = \boldsymbol{0}$$

- Substitute in $\boldsymbol{B}$ and $\boldsymbol{J}$[1]:

$$\boldsymbol{F} = F_\rho \boldsymbol{\nabla}\rho + F_\beta \boldsymbol{\beta}$$
$$\boldsymbol{\beta} = \boldsymbol{\nabla}\theta - \iota \boldsymbol{\nabla}\zeta$$
$$F_\rho = \sqrt{g}\left(B^\zeta J^\theta - B^\theta J^\zeta\right) - \frac{\partial p}{\partial \rho}$$
$$F_\beta = \sqrt{g}\, B^\zeta J^\rho$$

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

↓

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

↓

$F_\rho(\rho,\theta,\zeta)$
$F_\beta(\rho,\theta,\zeta)$

↓

≈ 0

[1]Hirshman et. al., *Phys. Fluids*, (1983).

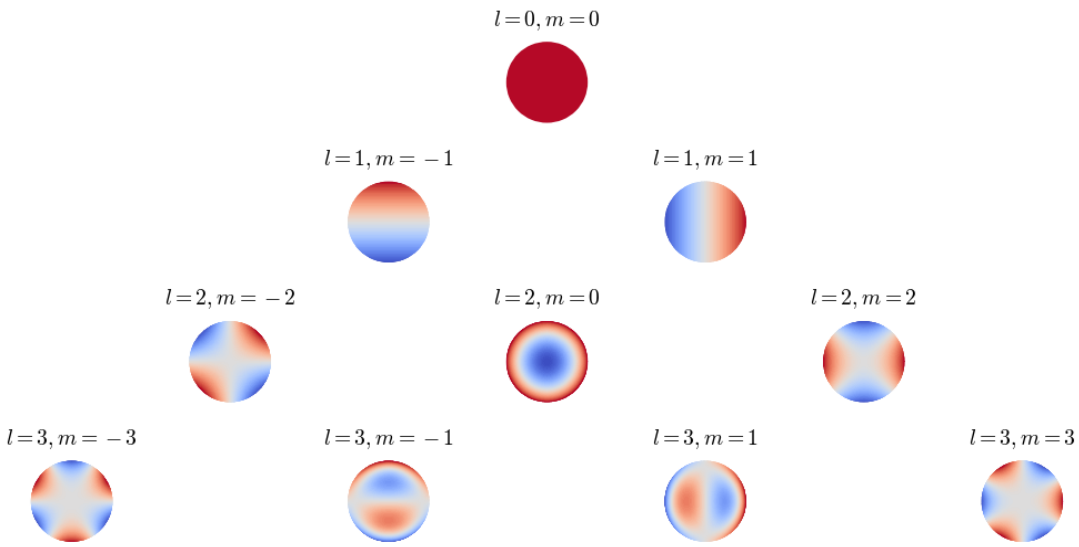
Global Fourier-Zernike[1,2] spectral bases are used for discretization

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

- Similar discretization for $Z(\rho,\theta,\zeta)$ and $\lambda(\rho,\theta,\zeta)$
- Inherently satisfies necessary boundary conditions at the magnetic axis for analytic functions[3,4]:

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

- Number of basis functions scales as $LMN/2$, so it uses about half as many terms as other methods



[1]Zernike, *Mon. Not. R. Astron. Soc.*, (1934).
[2]Loomis, *ASTM STP*, (1978).
[3]Boyd et. al., *J. Comput. Phys.*, (2011).
[4]Lewis et. al., *J. Math. Phys.*, (1990).

Equilibrium force balance is solved as a system of nonlinear equations

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

$$\boldsymbol{f} = \begin{bmatrix} f_\rho \\ f_\beta \\ BC \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} R_{lmn} \\ Z_{lmn} \\ \lambda_{lmn} \end{bmatrix} \quad \boldsymbol{c} = \begin{bmatrix} R^b_{mn} \\ Z^b_{mn} \\ p_l \\ \iota_l \\ \psi_a \end{bmatrix}$$

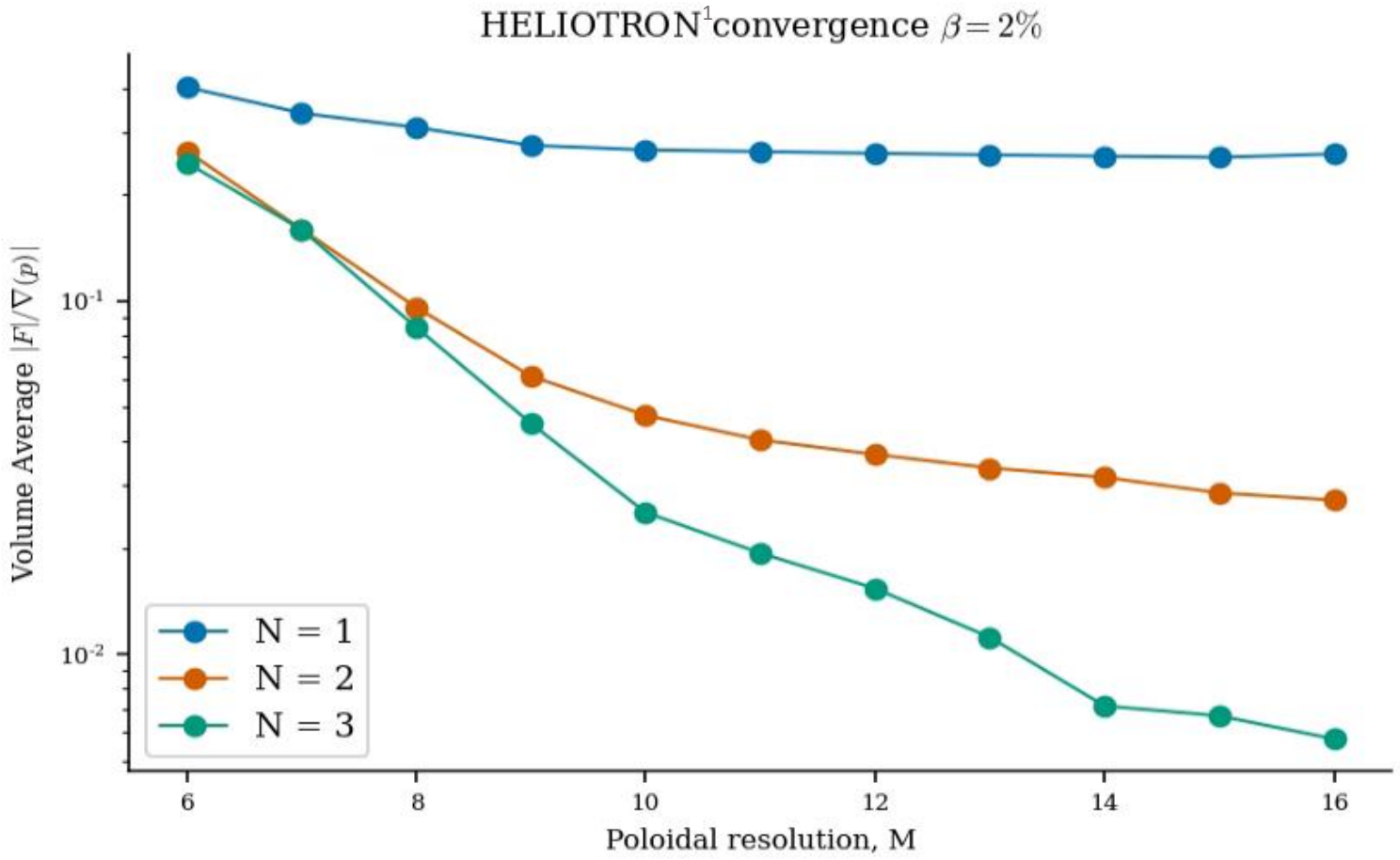
$$\boldsymbol{x}^* = \mathrm{argmin}_{\boldsymbol{x}}(\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})\|^2)$$

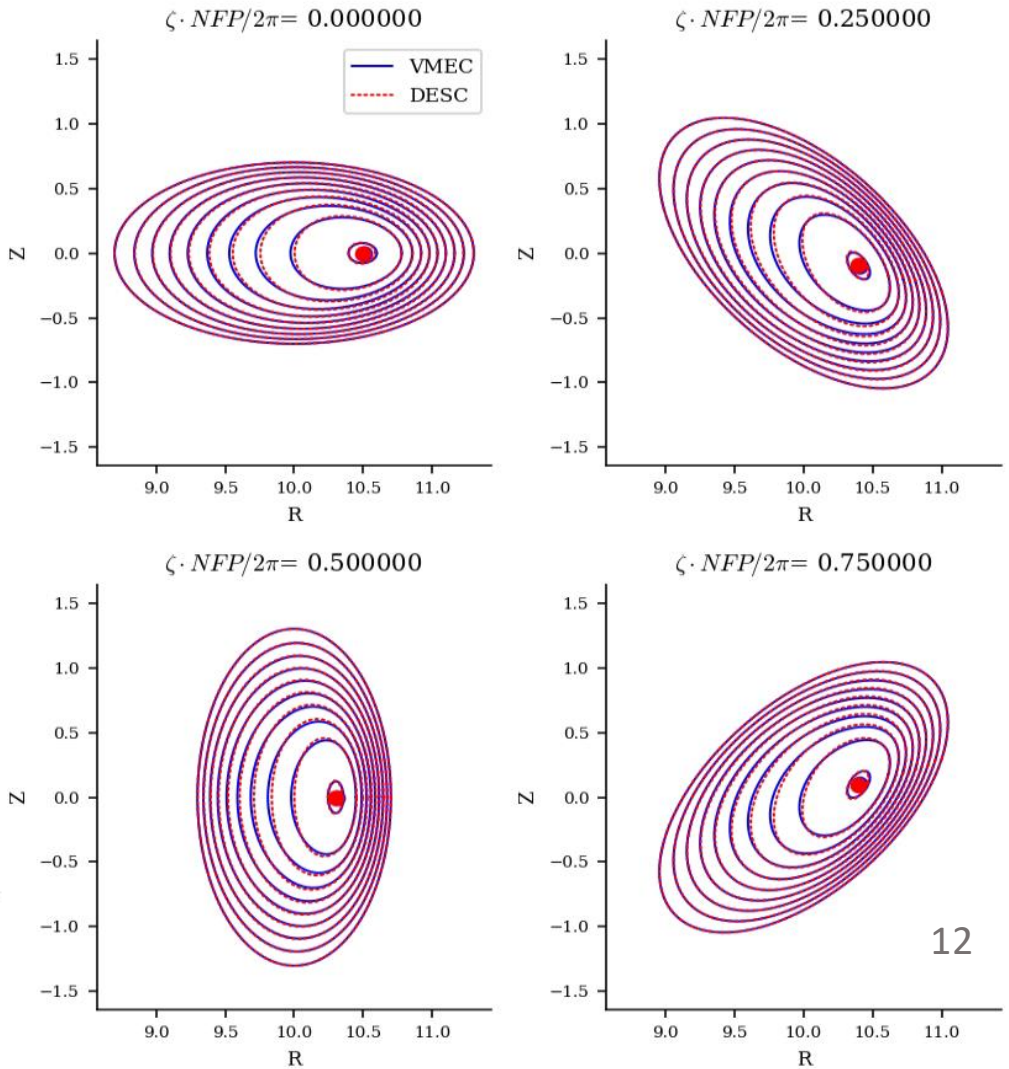
- Pseudo-spectral approach
  - Solution defined by global basis functions
  - Minimize residuals at collocation nodes
  - Exponential convergence if smooth
- Flexible choice of solver
  - Modified Newton methods, least-squares minimization, custom routines, etc.
  - Quadratic convergence near solution

Modern object-oriented code design

- Written in Python3
  - Object-oriented structure is easy to use, extend, and interface with other codes
  - Modular design allows flexibility for different solvers, objectives, etc.
- Uses JAX
  - Developed by Google, same backend as TensorFlow
  - Automatic differentiation for exact derivatives of arbitrary order
  - JIT compilation to CPU & GPU via XLA

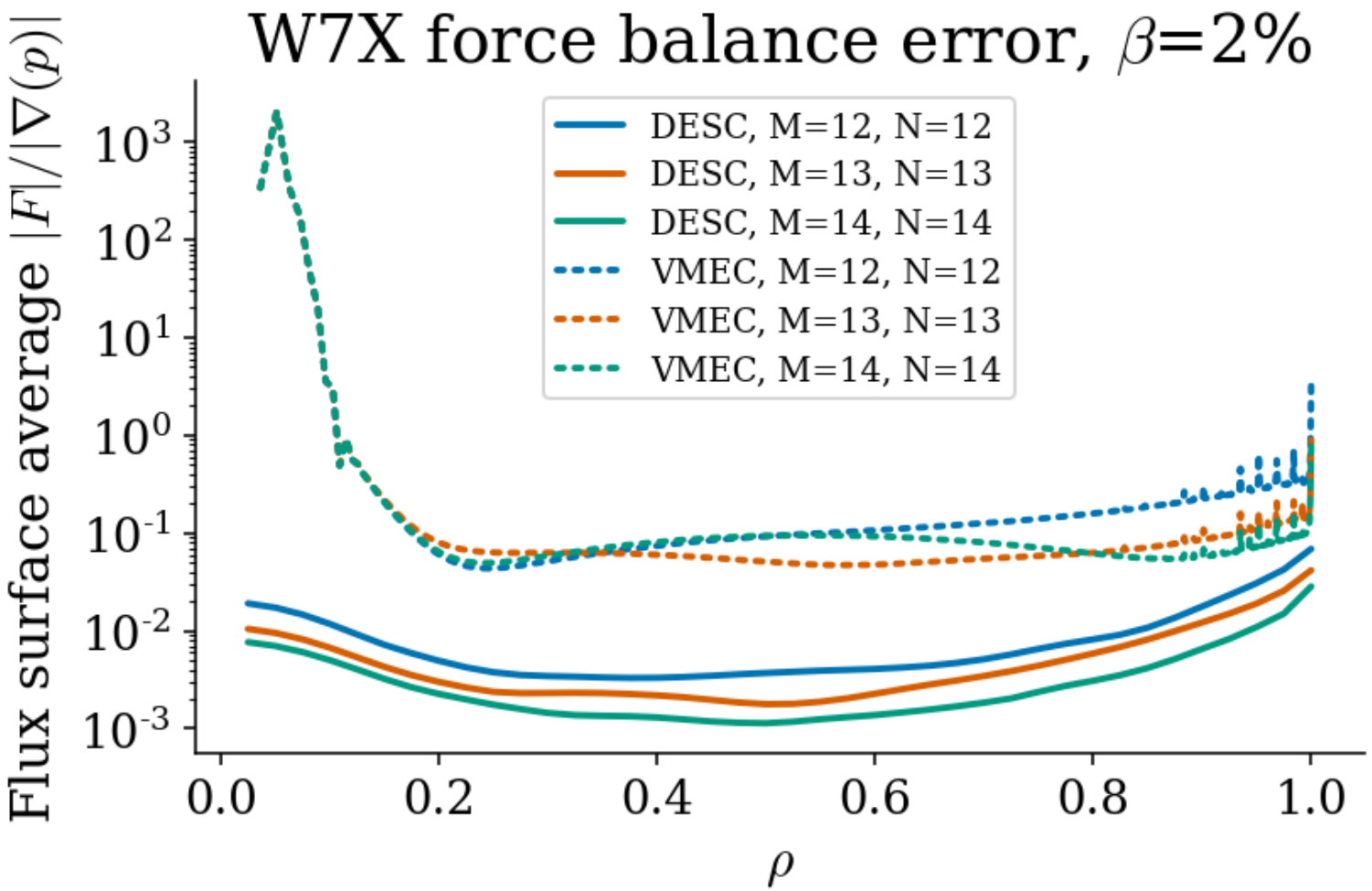
DESC shows strong convergence and agreement with VMEC



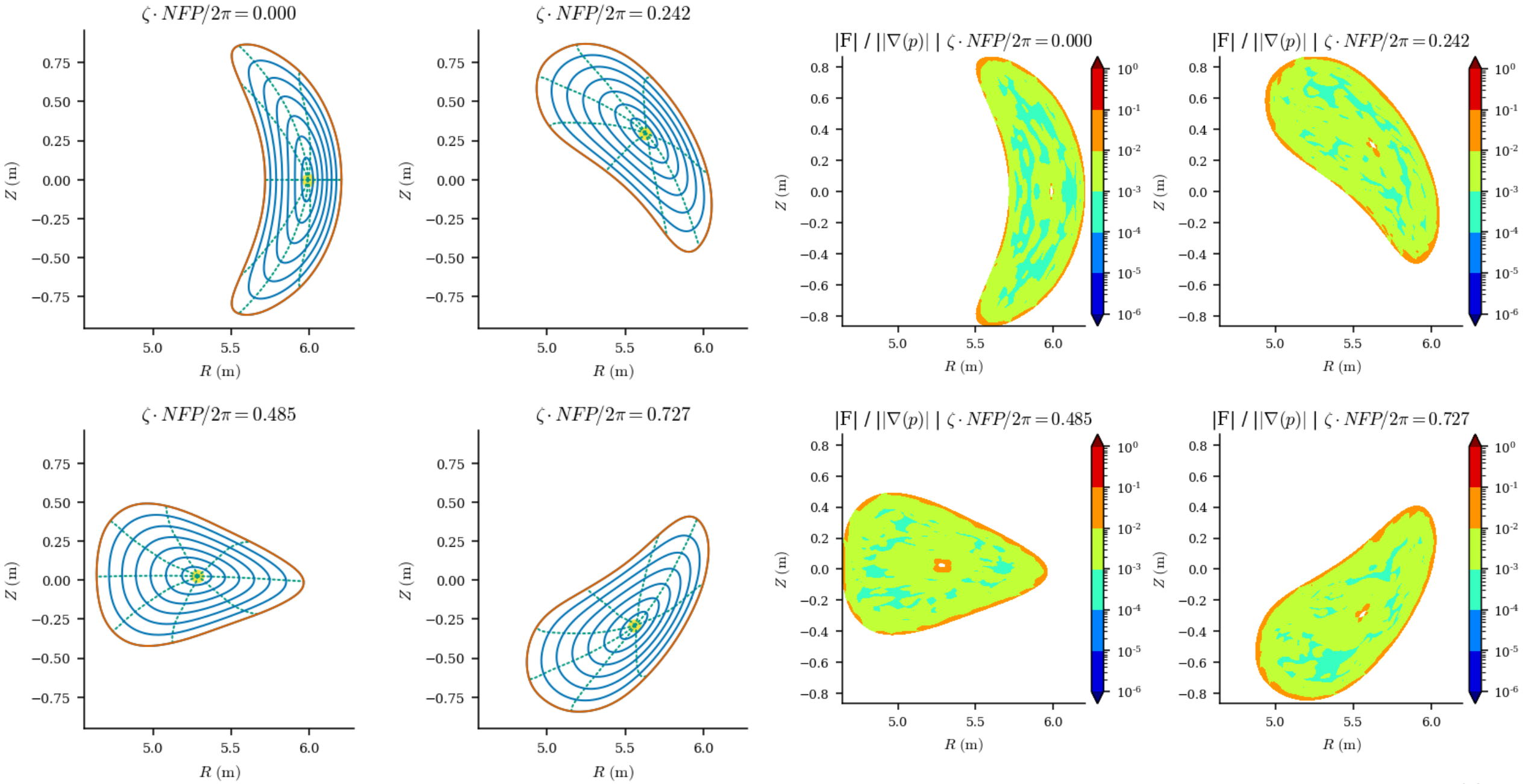


[1]Hirshman et. al., *Phys. Fluids*, (1983).

DESC typically reaches lower force balance errors than VMEC, at similar resolutions



- DESC maintains convergence while VMEC reaches a minimum error threshold
- VMEC usually has poor accuracy near the magnetic axis
- VMEC requires many flux surfaces for high resolution


$\zeta \cdot NFP/2\pi = 0.000$
$\zeta \cdot NFP/2\pi = 0.242$
$\zeta \cdot NFP/2\pi = 0.485$
$\zeta \cdot NFP/2\pi = 0.727$
$|F| \,/\, ||\nabla(p)| \,|\, \zeta \cdot NFP/2\pi = 0.000$
$|F| \,/\, ||\nabla(p)| \,|\, \zeta \cdot NFP/2\pi = 0.242$
$|F| \,/\, ||\nabla(p)| \,|\, \zeta \cdot NFP/2\pi = 0.485$
$|F| \,/\, ||\nabla(p)| \,|\, \zeta \cdot NFP/2\pi = 0.727$
Z (m)
R (m)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}

DESC's equilibrium solver speed is comparable to VMEC

W7-X resolution: M=12, N=12	
DESC on CPU*	195 min
VMEC on CPU*	156 min
DESC on GPU	10 min

*Tests conducted on different hardware, so comparison is only approximate

- Solving a single equilibrium on a CPU (at similar resolutions):
  - Speed is comparable to VMEC
  - DESC achieves lower force errors
- GPU computations are much faster
- Further profiling and speed improvements are in progress
  - Randomized SVD (fbpca)
  - Stochastic/iterative Newton methods
  - Optimal stopping criteria

DESC's stellarator optimization is orders of magnitude faster

- Finite differences (VMEC + STELLOPT[1]):
  - $\sim O$(hours – days)
  - Limited accuracy
- Adjoint method (VMEC + ALPOpt[2]):
  - $\sim O$(minutes – hours)
  - Difficult to implement
  - Not applicable for all objectives / derivative orders
- Automatic differentiation (DESC)
  - $\sim O$(seconds – minutes)
  - Easy to implement (often only 1 extra line of code)
  - Can differentiate arbitrary objectives with respect to arbitrary parameters

Optimization requires derivative information
(How the equilibrium changes as parameters are varied)

[1]Lazerson et. al., *STELLOPT*, (2020).
[2]Paul et. al., *J. Plasma Phys.*, (2021).

Perturbations provide approximations to neighboring solutions

$\boldsymbol{f}$ = force balance error
$\boldsymbol{x}$ = state vector
$\boldsymbol{c}$ = input parameters

- First-order Taylor expansion:

$$\boldsymbol{f}(\boldsymbol{x}+\Delta\boldsymbol{x},\boldsymbol{c}+\Delta\boldsymbol{c}) = \cancelto{\sim 0}{\boldsymbol{f}(\boldsymbol{x},\boldsymbol{c})} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}\Delta\boldsymbol{c} = \boldsymbol{0}$$

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

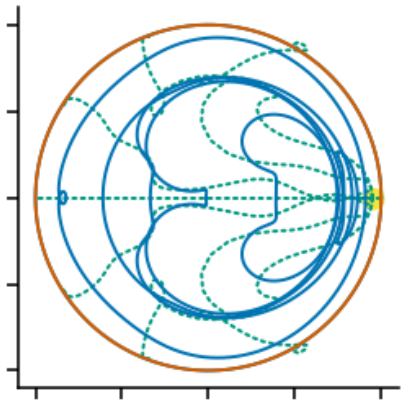
- Derivatives are computed exactly and efficiently with automatic differentiation
- Jacobian matrix $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ is already computed in the final Newton iteration when solving the equilibrium
- Easily extended to higher order : $\Delta x = \varepsilon x_1 + \varepsilon^2 x_2 + \ldots$

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

Trust region method regularizes perturbation step

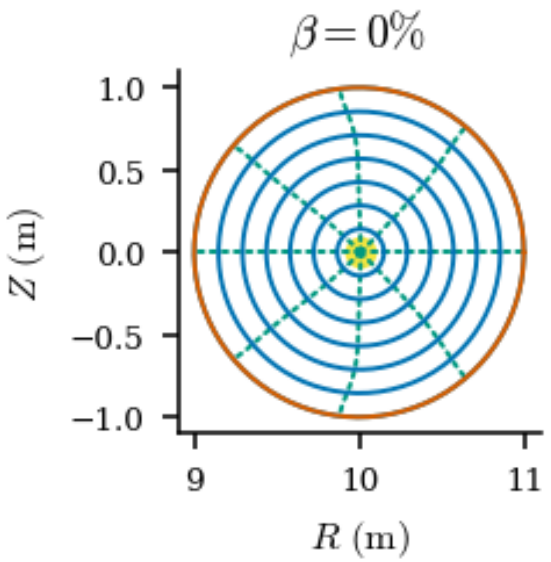
- When perturbations are large or curvature is strong unconstrained step can be too large, leading to non-physical results
  - Don't want exact solution to approximate model

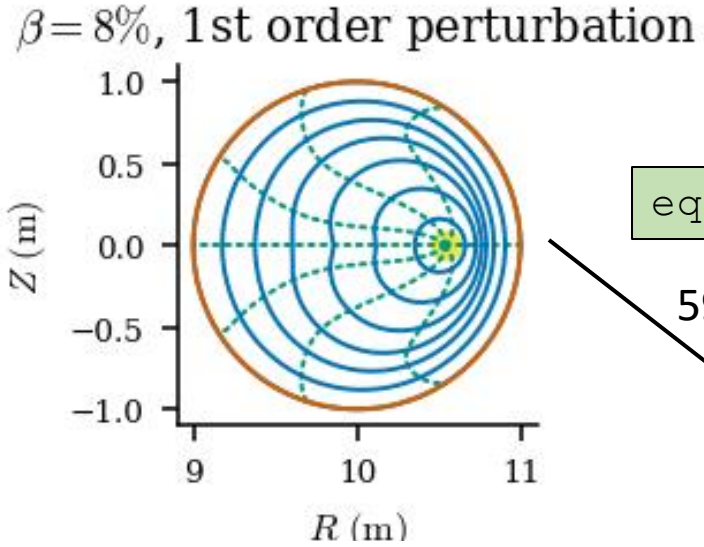


- Solution: constrain step size to lie within "trust region" in which we trust the Taylor approximation
  - $\min_{\Delta x} \left| \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \Delta x - b \right| \quad s.t. \;\; |\Delta x| < r$
- Subproblem solved using univariate Newton's method
  - Cost 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

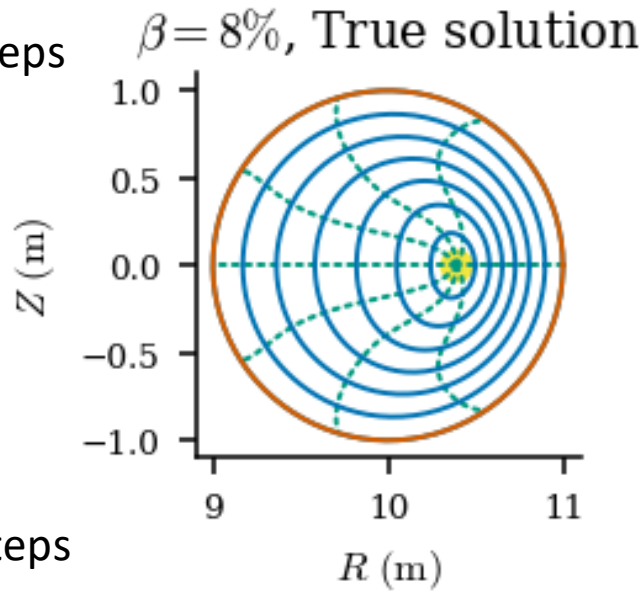
```
eq1 = eq.perturb(dp=delta_p,
                 order=1)
```

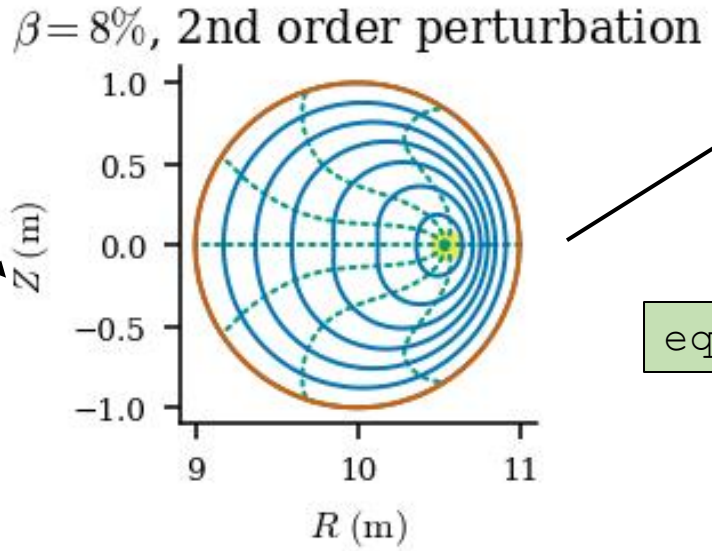





```
eq1.solve(ftol=1e-2)
```

59 optimizer steps





30 optimizer steps

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

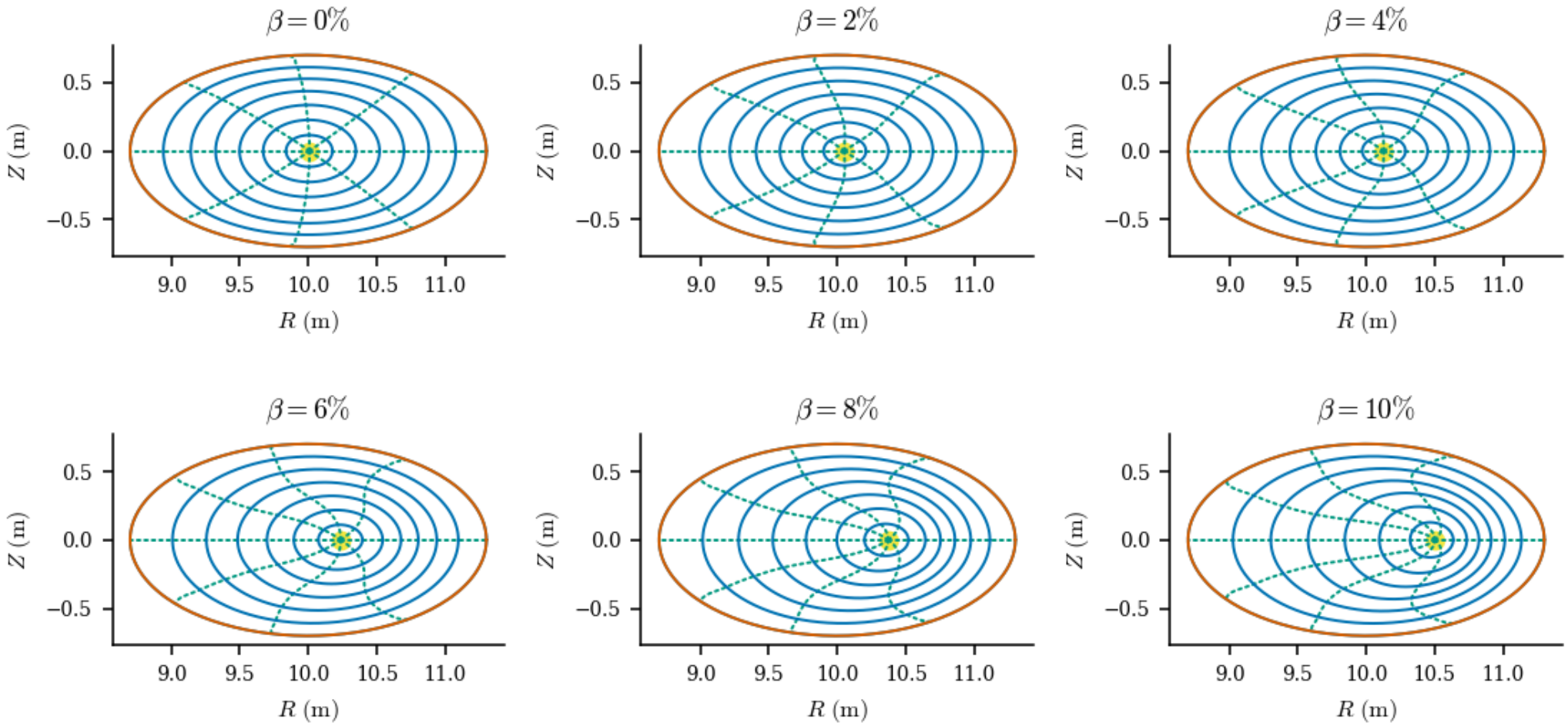
```
eq2 = eq.perturb(dp=delta_p,
                 order=2)
```

Continuation methods enable exploration of the stellarator phase space

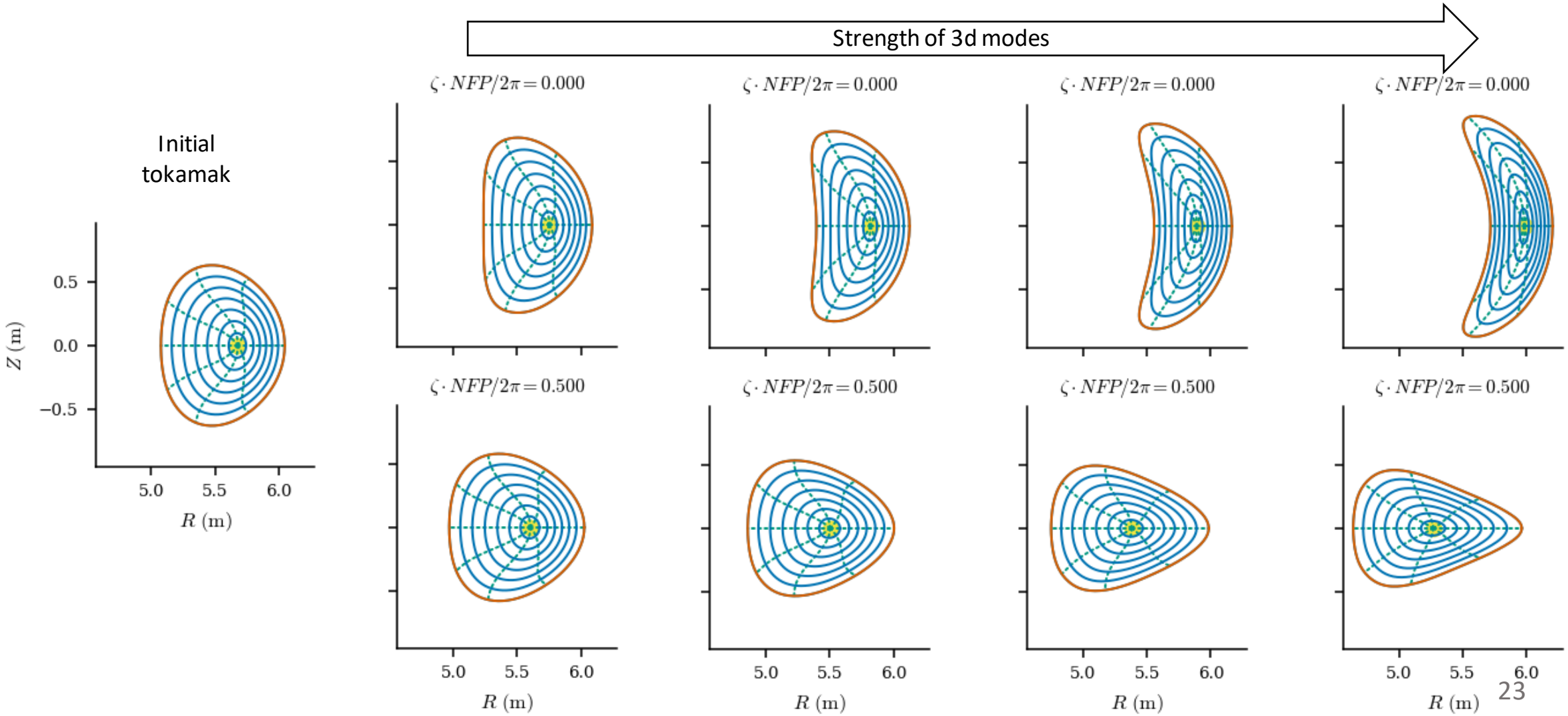
- Once a single equilibrium solution is found, varying an input parameter reveals a family of solutions
  - Pressure scans, boundary perturbations, etc.
- Provides an efficient way to navigate the solution landscape
- Exposes the connection between discrete equilibrium solutions

Perturbations compute parameter scans "for free"

```
delta_p = 1.8e3*np.array([1, -2, 1])
eq_new = eq.perturb(dp=delta_p, order=1)
```



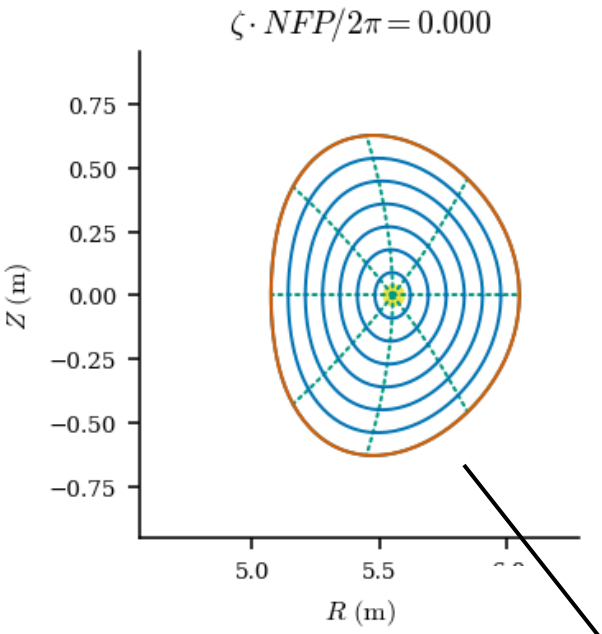
Example: continuously deforming tokamak into stellarator



Continuation Example: W7-X

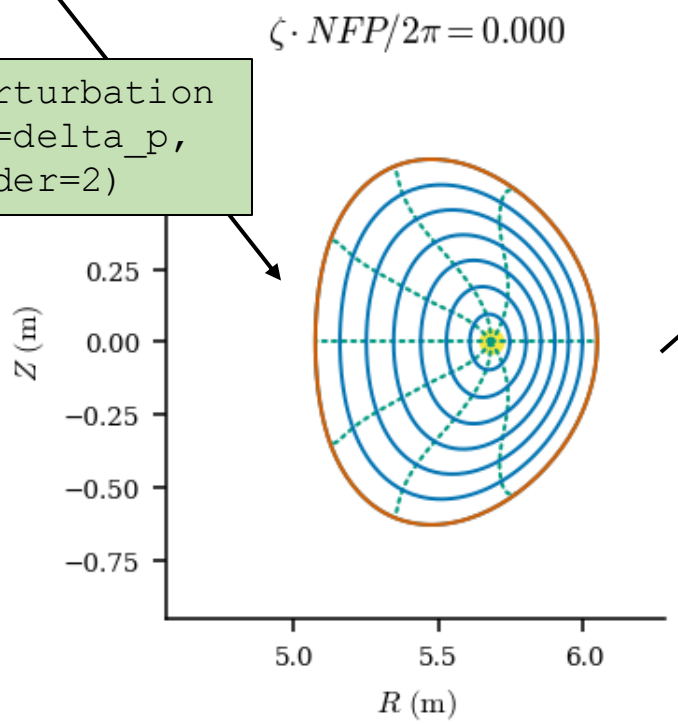
(second-order perturbations)

Vacuum tokamak



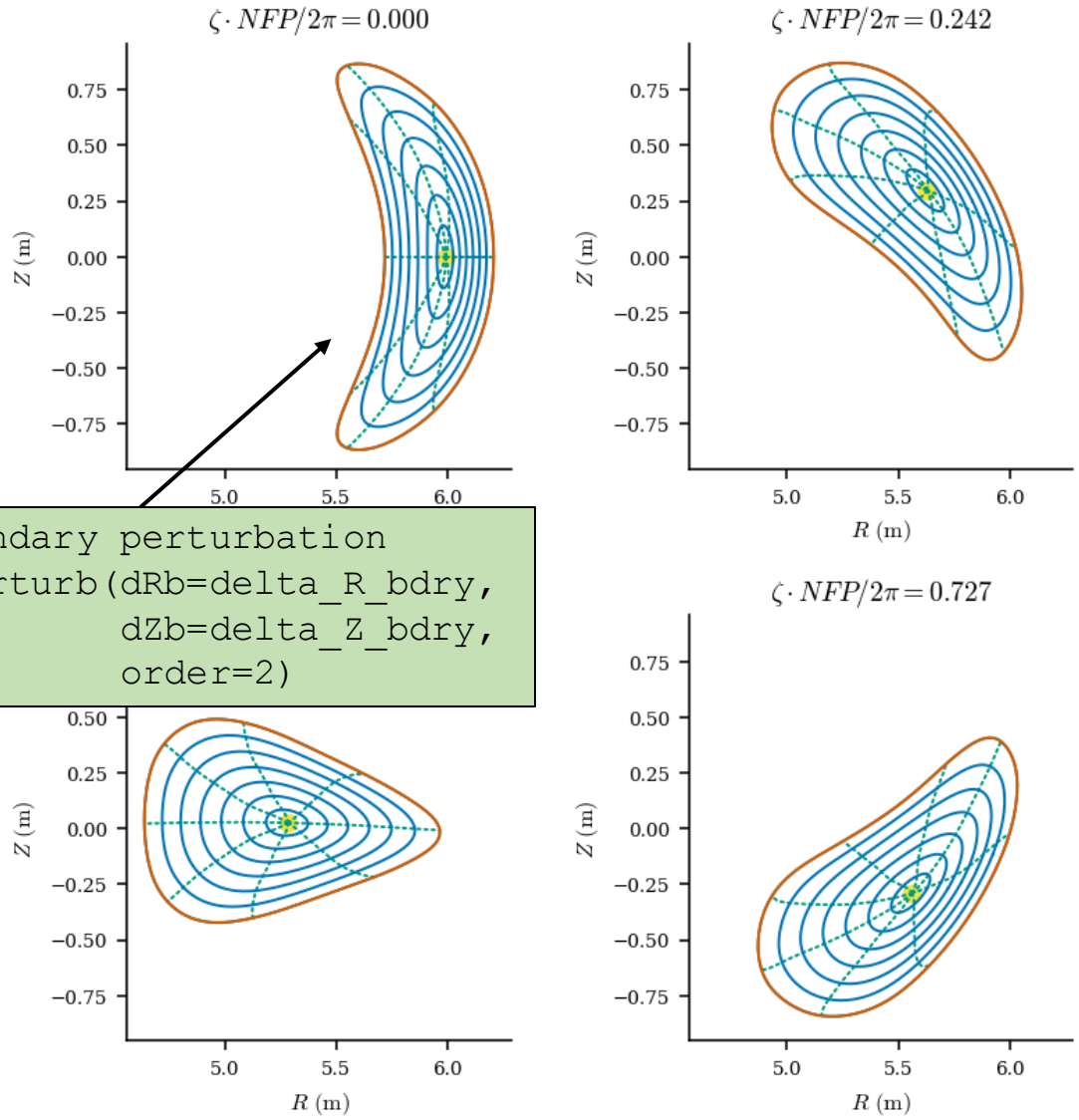

```
# pressure perturbation
eq.perturb(dp=delta_p,
           order=2)
```

Finite-β tokamak




```
# boundary perturbation
eq.perturb(dRb=delta_R_bdry,
           dZb=delta_Z_bdry,
           order=2)
```

Finite-β stellarator



Optimization objectives can be incorporated into the perturbations

- Define an optimization cost function $\boldsymbol{g} \equiv \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c})$
  - Quasi-symmetry, coil complexity, etc.
- First-order Taylor expansion:

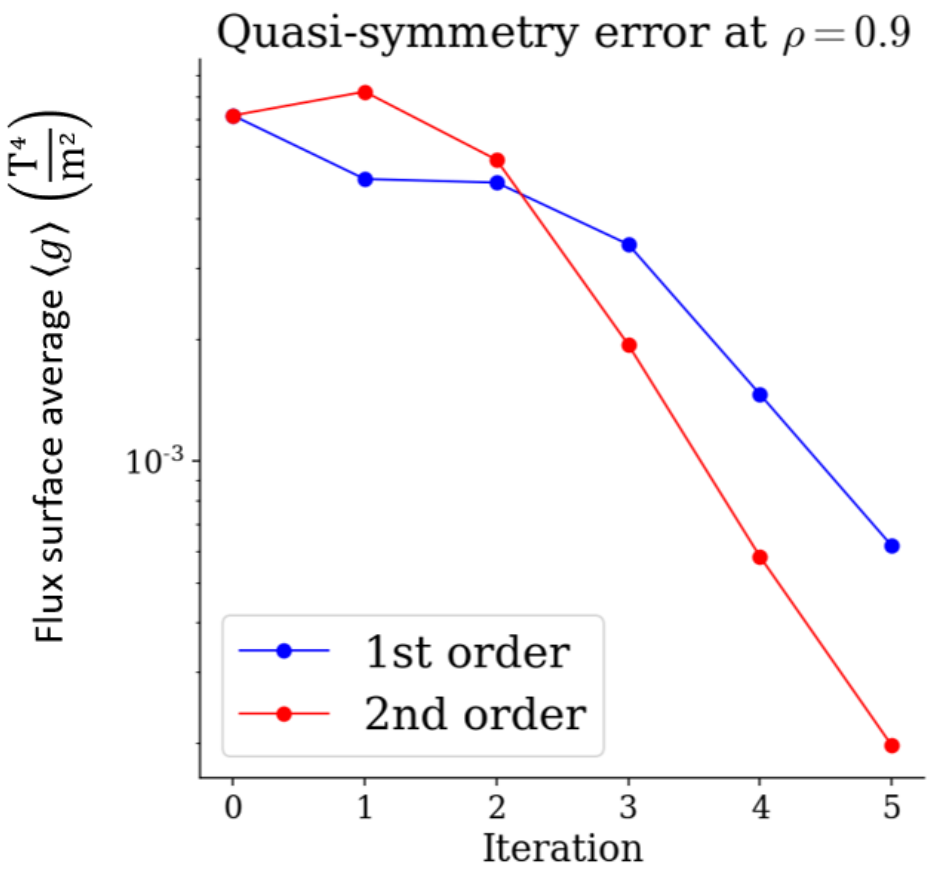
$$\boldsymbol{g}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c}) + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \Delta\boldsymbol{x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{c}} \Delta\boldsymbol{c} = \boldsymbol{0}$$

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

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

- This yields the perturbation $\Delta\boldsymbol{c}$ that most improves the objective while maintaining force balance
- No additional equilibrium solves required!

Example optimization objective: quasi-symmetry

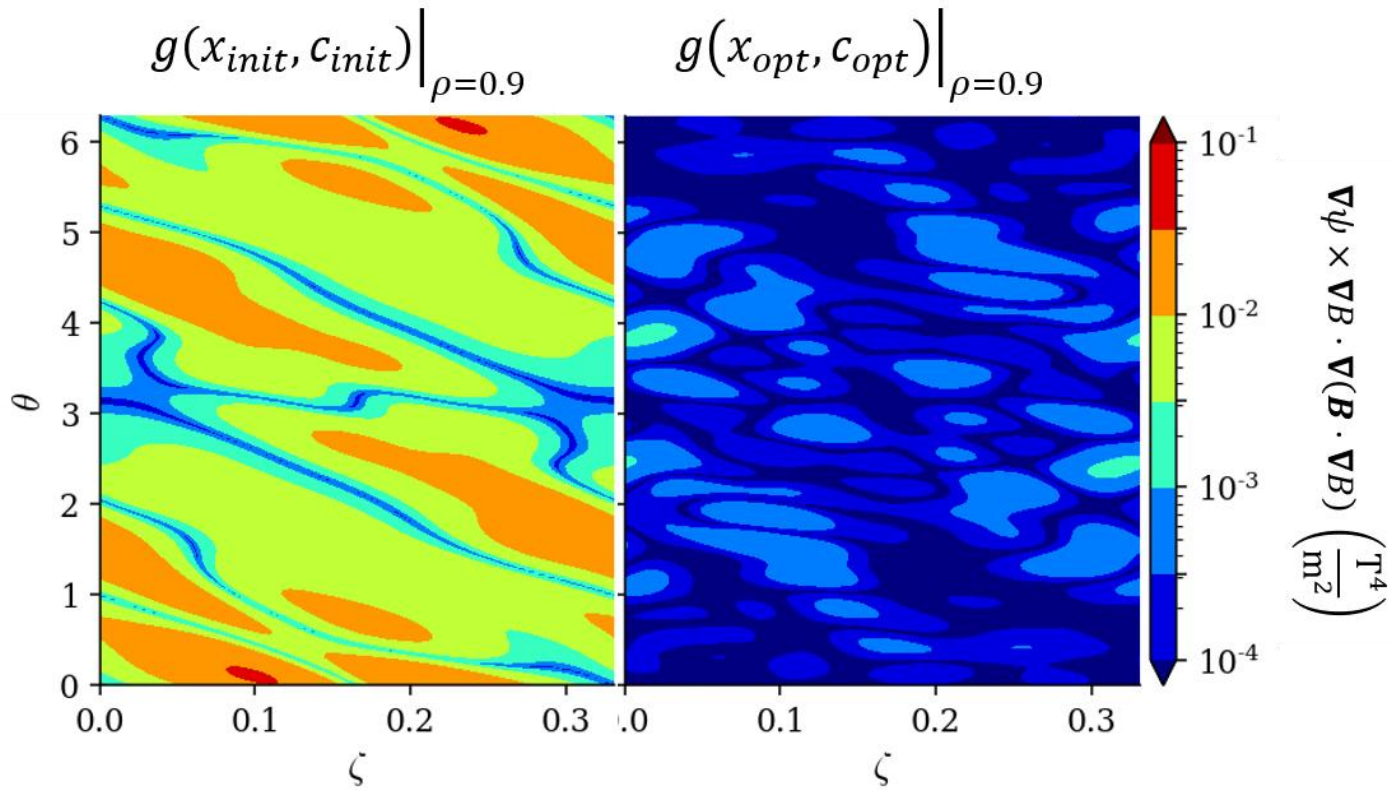


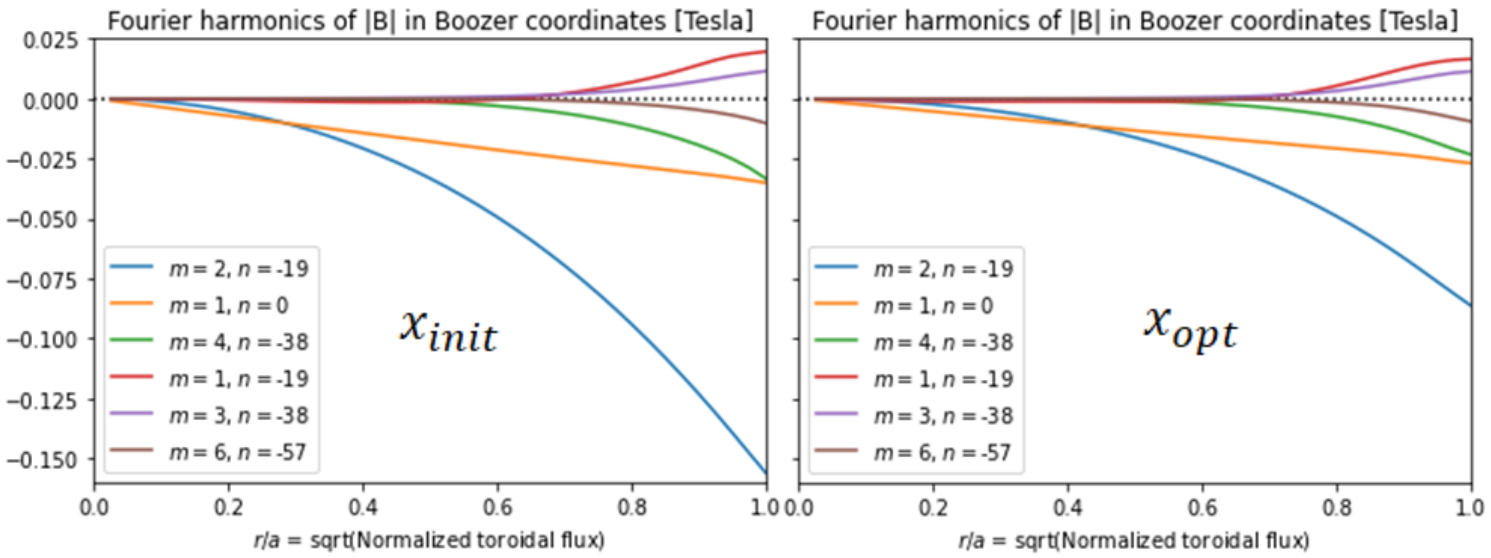
- Optimized the boundary shape for quasi-symmetry at the ρ=0.9 flux surface

$$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c}) \equiv \nabla\psi \times \nabla B \cdot \nabla(\boldsymbol{B} \cdot \nabla B)^{1}$$

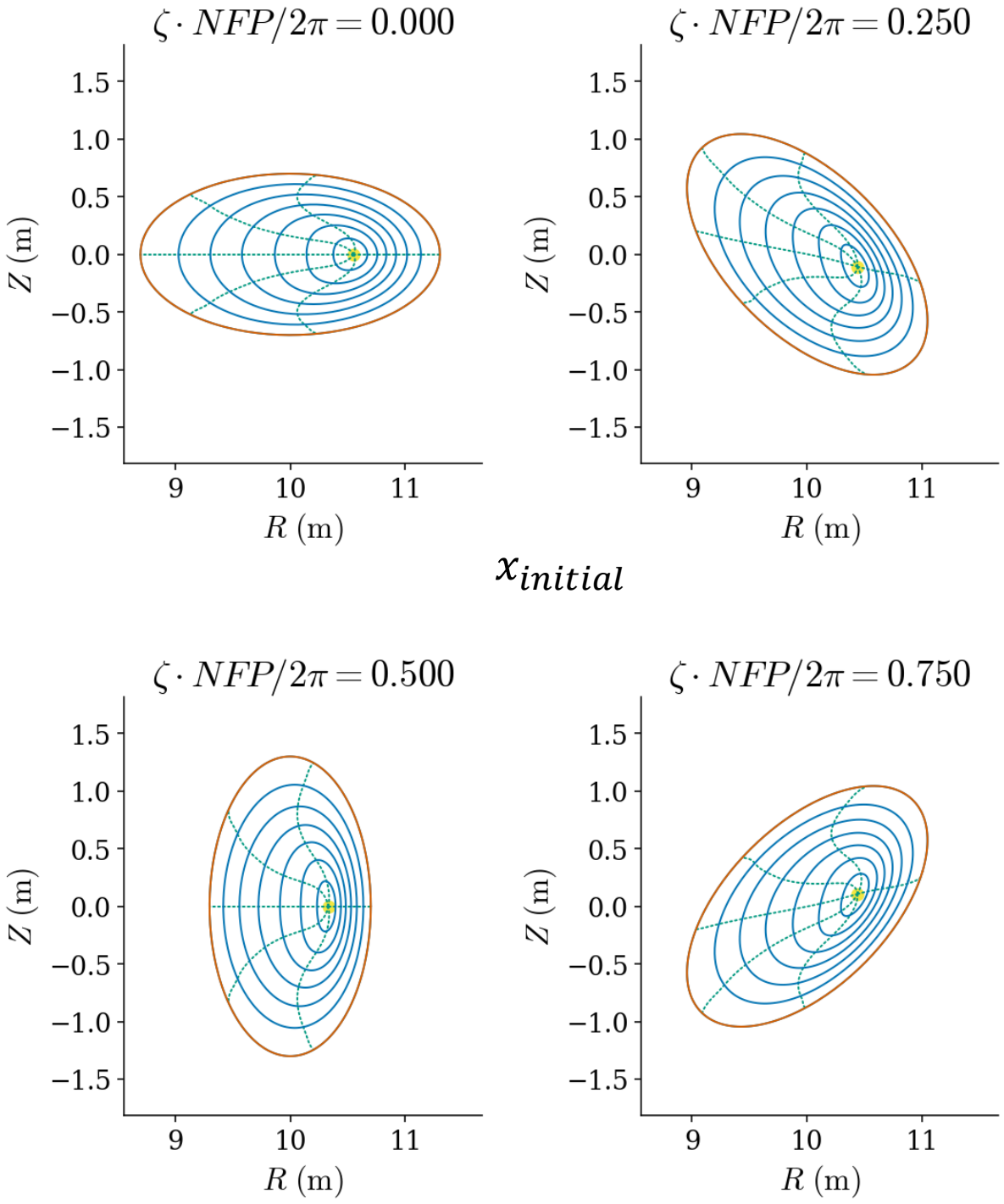
```
# optimization loop
for step in iterations:
    eq.perturb(
        objective=QS_fun,
        dRb=R_bdry_modes,
        dZb=Z_bdry_modes,
    )
# a few Newton iterations to
# make sure it's still converged
eq.solve(ftol=1e-2)
```

[1]Helander, *Rep. Prog. Phys.*, (2014).

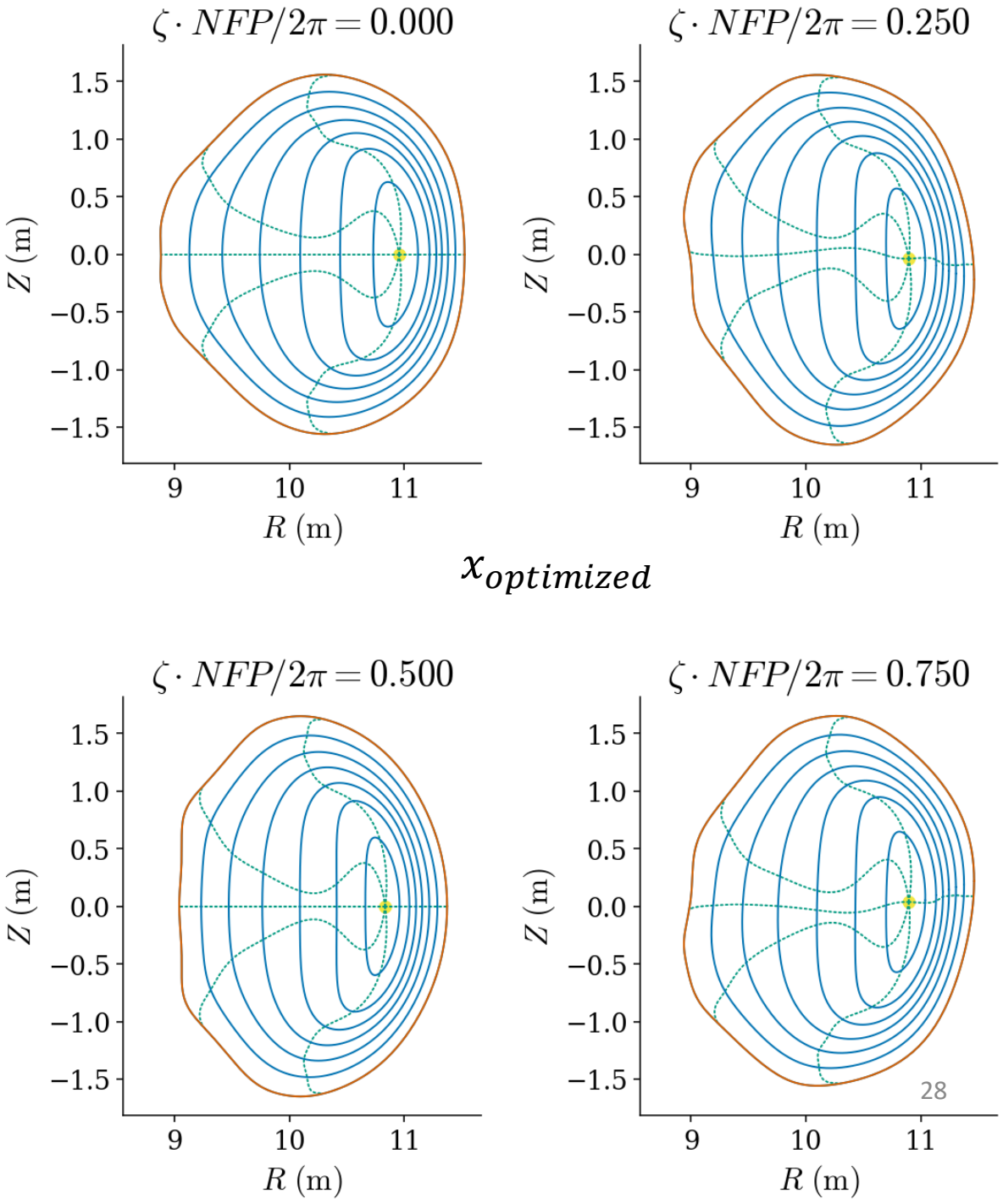




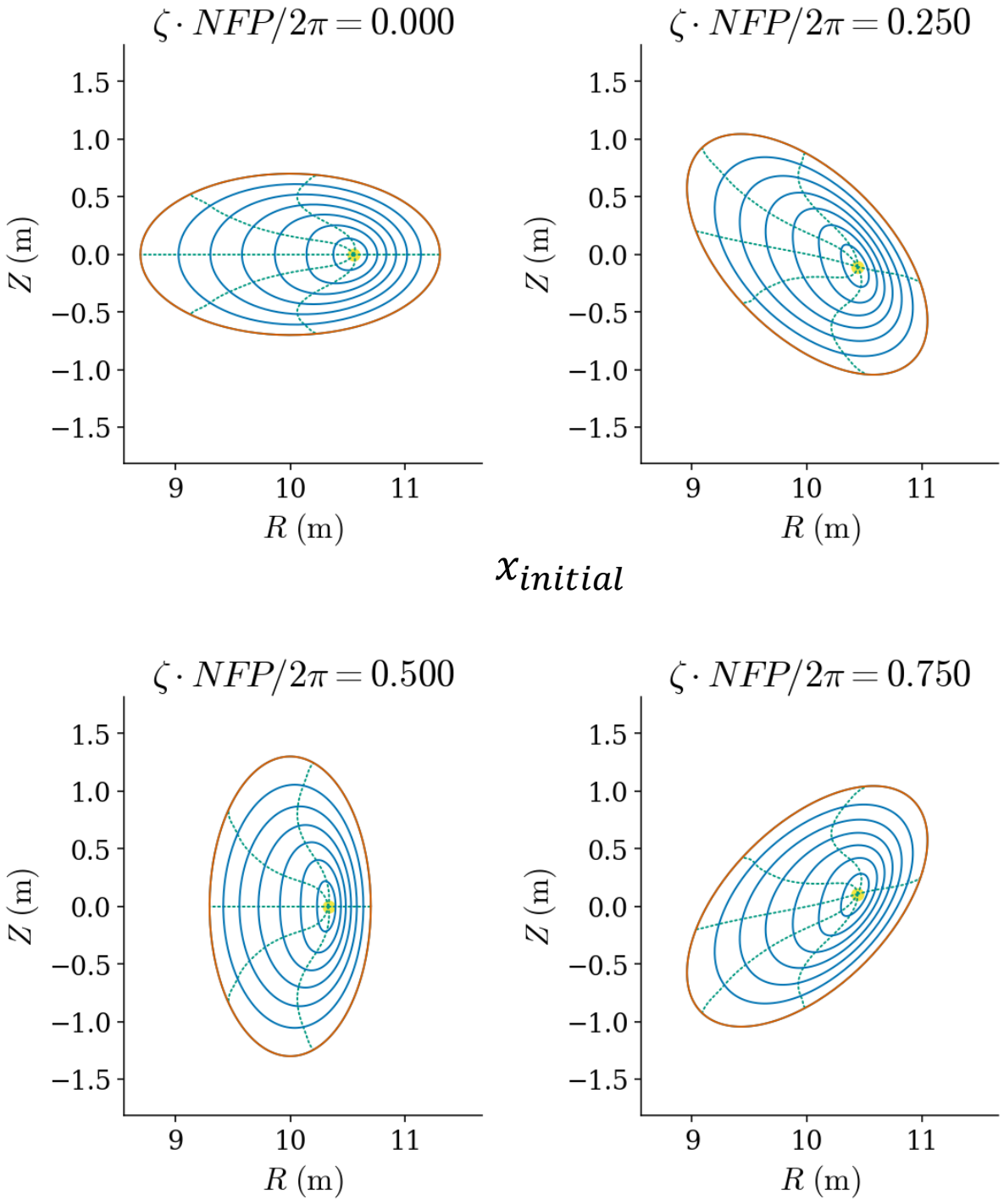
- Optimizing for triple-product quasi-symmetry metric at a single surface improved the Boozer Fourier harmonics throughout the volume



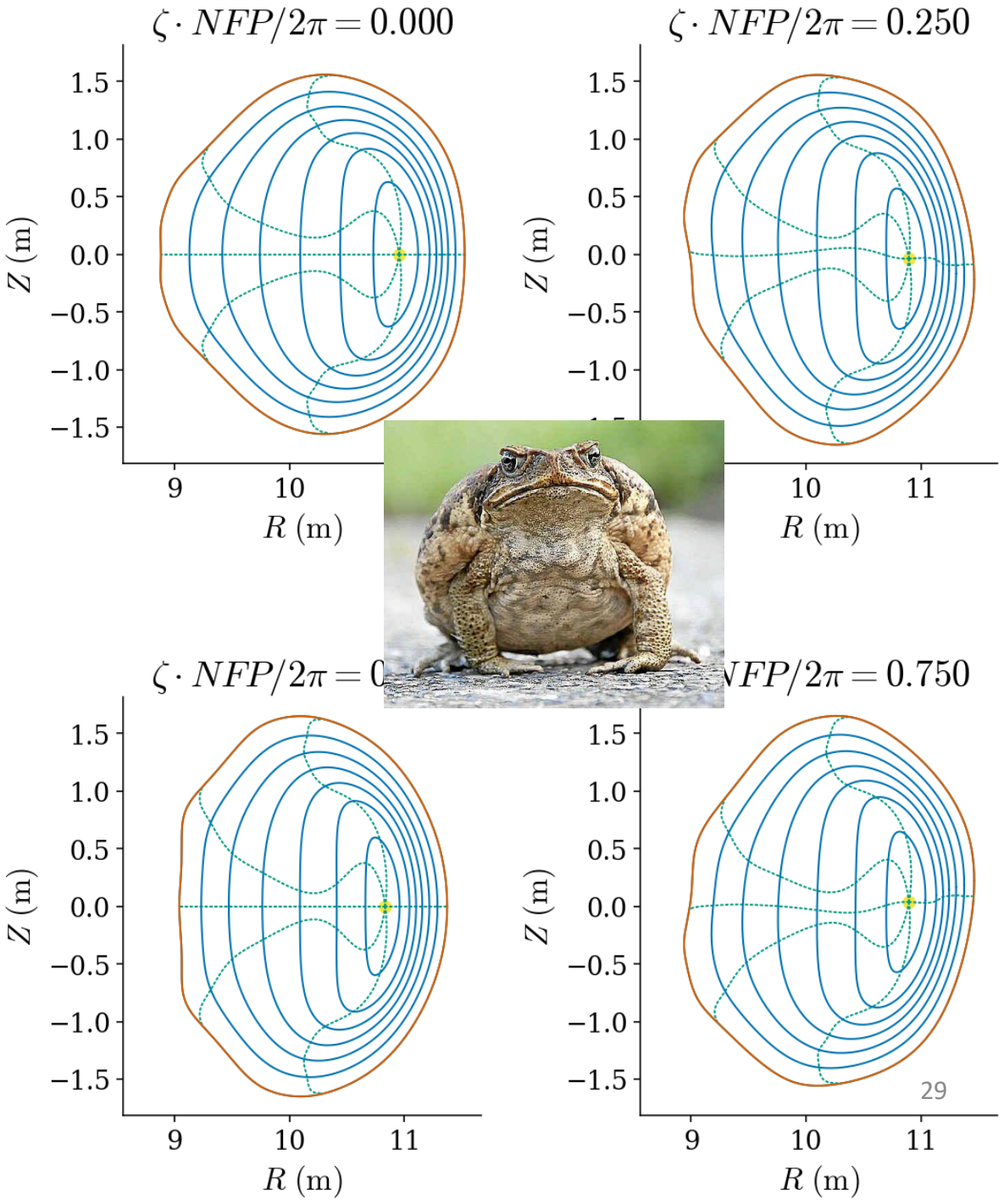
$x_{initial}$



$x_{optimized}$

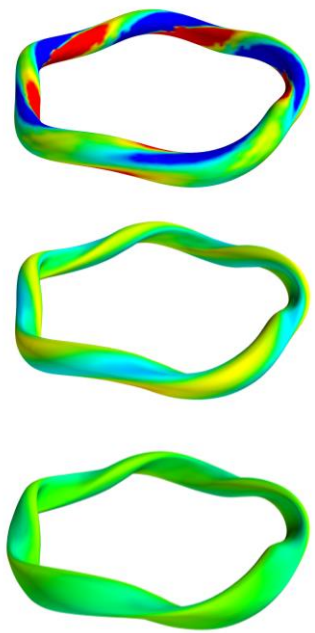

$\zeta \cdot NFP/2\pi = 0.000$
$\zeta \cdot NFP/2\pi = 0.250$
$\zeta \cdot NFP/2\pi = 0.500$
$\zeta \cdot NFP/2\pi = 0.750$
Z (m)
R (m)

$x_{initial}$


$\zeta \cdot NFP/2\pi = 0.000$
$\zeta \cdot NFP/2\pi = 0.250$
$\zeta \cdot NFP/2\pi = 0.750$
Z (m)
R (m)

Free-Boundary extension for DESC (J. Schilling)

- NESTOR[1] provides the free-boundary contribution in VMEC
  - computes normal magnetic field required to cancel the plasma-vacuum-mismatch
- Equations for analytic cancellation of singularities re-derived and implemented in python
  - Verified correctness vs original (Fortran-77) version
- Coupling to DESC is currently being implemented
  - Switching to use JAX for numerics to make use of available GPU power / autodiff



Φ_B on LCFS for free boundary VMEC w/ NESTOR

[1] P. Merkel, J. Comp. Phys **66**, 83 (1986)

MHD Equilibrium Reconstructions using DESC (J. Schilling)

- free-boundary equilibrium solver required to accurately represent realistic experimental conditions (coming soon)
- Prediction of magnetic diagnostic signals[2] needs current density from equilibrium
  - Exact analytic derivatives with DESC should lead to cleaner synthetic diagnostics
- Strong benefits because derivative information is available in DESC
  - no need for finite differencing between equilibria -> fewer evaluations of equilibrium
  - faster reconstruction of experimental data
  - uncertainty quantification

[2] S. P. Hirshman et al, Phys. Plasmas **11**, 595 (2004)

Current & Future work

- Free boundary equilibria
- Equilibrium reconstruction
- Constraining toroidal current profile vs iota
- Alternate boundary conditions
- Couple with FOCUSADD for combined coil + plasma optimization
- Magnetic islands & stochastic regions

DESC was thoughtfully designed to be the ideal stellarator optimization code

- **Force balance** (instead of energy) minimization achieves lower error solutions
- **Newton methods** for solving systems of equations allows quadratic convergence
- **Pseudo-spectral** method allows exponential convergence
- **Zernike polynomial** basis functions properly resolve the magnetic axis
- **Python** code provides a modular framework that is easy to use
- **Automatic differentiation** supplies exact derivatives for solving and optimization
- **GPUs** are the future workhorses of high-performance computing

Thank You!

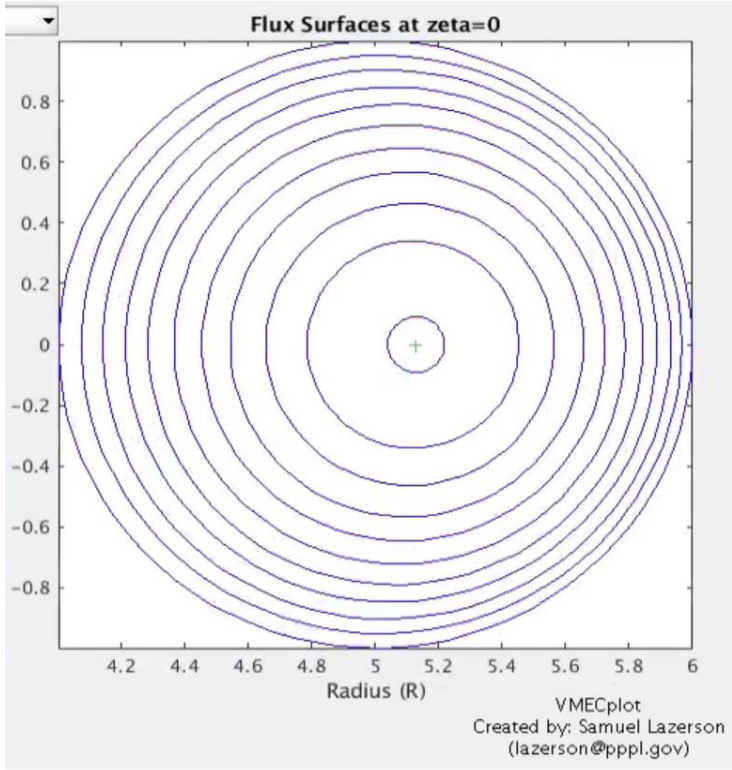
Publication:	Dudt et. al. *Phys. Plasmas*. 2020.
Repository:	https://github.com/ddudt/DESC
Python Package:	`pip install desc-opt`

Flux Coordinates

- Choose the flux surface label proportional to the minor radius: $\rho = \sqrt{\psi_N}$
- Covariant basis vector definitions:

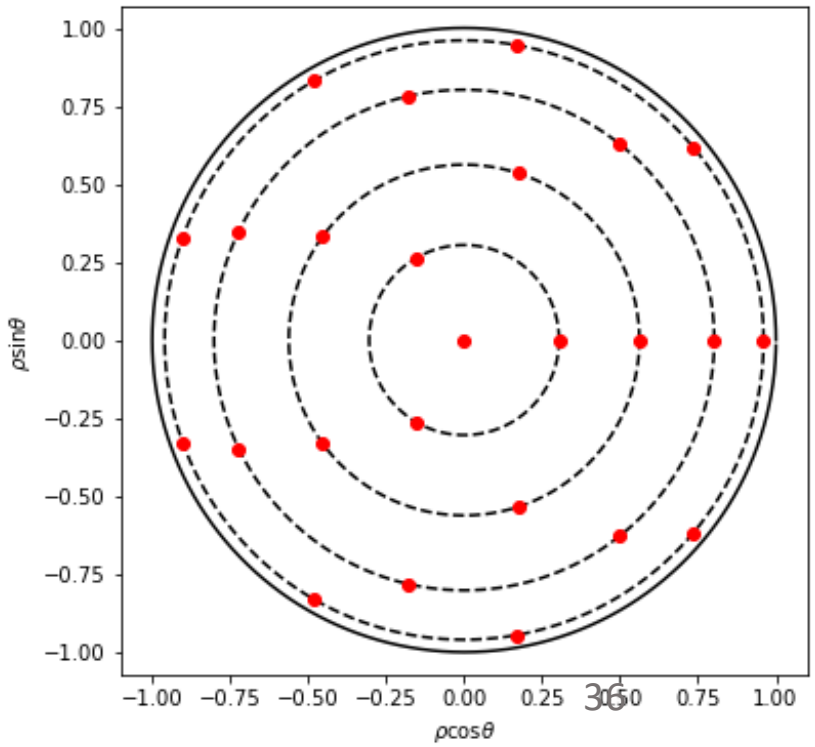
$$\boldsymbol{e}_\rho = \begin{bmatrix} \partial_\rho R \\ 0 \\ \partial_\rho Z \end{bmatrix} \qquad \boldsymbol{e}_\vartheta = \begin{bmatrix} \partial_\theta R \\ 0 \\ \partial_\theta Z \end{bmatrix} \qquad \boldsymbol{e}_\zeta = \begin{bmatrix} \partial_\zeta R \\ R \\ \partial_\zeta Z \end{bmatrix}$$

- Jacobian: $\sqrt{g} = \boldsymbol{e}_\rho \cdot \boldsymbol{e}_\theta \times \boldsymbol{e}_\zeta$



Collocation Nodes

- The computational grid is a finite set of discrete points $(\rho_i, \theta_i, \zeta_i)$
- The force balance errors $f_\rho(\rho, \theta, \zeta)$ & $f_\beta(\rho, \theta, \zeta)$ are minimized at these nodes
- Spectral collocation theory predicts *global* convergence
- Flexibility in choosing the nodes provides:
  - Control of grid refinement
  - Avoidance of rational surfaces



f(x,c) = 0 == equilibrium
x = magnetic fields
c = pressure, boundary, etc

$$
\begin{aligned}
f(x+\Delta x, c+\Delta c) &= f(x,c) + \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial c}\Delta c + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\Delta x \Delta x^T \\
&+ \frac{1}{2}\frac{\partial^2 f}{\partial c^2}\Delta c \Delta c^T + \frac{\partial^2 f}{\partial x \partial c}\Delta x \Delta c^T + O(\Delta x^3)
\end{aligned}
$$

$$
\begin{aligned}
\Delta x &= \epsilon x_1 + \epsilon^2 x_2 \\
\Delta c &= \epsilon c_1
\end{aligned}
$$

Assume perturbation series for x

$$
\epsilon x_1 = -\left(\frac{\partial f}{\partial x}\right)^{-1}\left(f(x,c) + \frac{\partial f}{\partial c}\epsilon c_1\right)
$$

2nd order step computed using directional derivatives in direction given by 1st order terms

$$
\epsilon^2 x_2 = -\left(\frac{\partial f}{\partial x}\right)^{-1}\left(\frac{1}{2}\frac{\partial^2 f}{\partial x^2}\epsilon^2 x_1 x_1^T + \frac{1}{2}\frac{\partial^2 f}{\partial c^2}\epsilon^2 c_1 c_1^T + \frac{\partial^2 f}{\partial x \partial c}\epsilon^2 x_1 c_1^T\right)
$$

Least squares optimization

$$y = \frac{1}{2}\mathbf{f}^T\mathbf{f}$$

$$J = \frac{d\mathbf{f}}{d\mathbf{x}}$$

$$\mathbf{g} = \nabla y = \mathbf{f}^T\frac{d\mathbf{f}}{d\mathbf{x}} = \mathbf{f}^T J$$

$$H = \nabla^2 y = J^T J + \mathbf{f}^T\frac{d^2\mathbf{f}}{d\mathbf{x}^2}$$

- "small residual approximation"
  - $H \sim J^T J$
- even for large $|\mathbf{f}|$, $H \sim J^T J$ is often still useful
- Approximates convex hull of true solution landscape
- Helps to avoid stalling & local minima

Trust region method

$$\min_{\mathbf{p}} ||J\mathbf{p} - \mathbf{f}|| \quad s.t. \quad ||\mathbf{p}|| \leq r$$

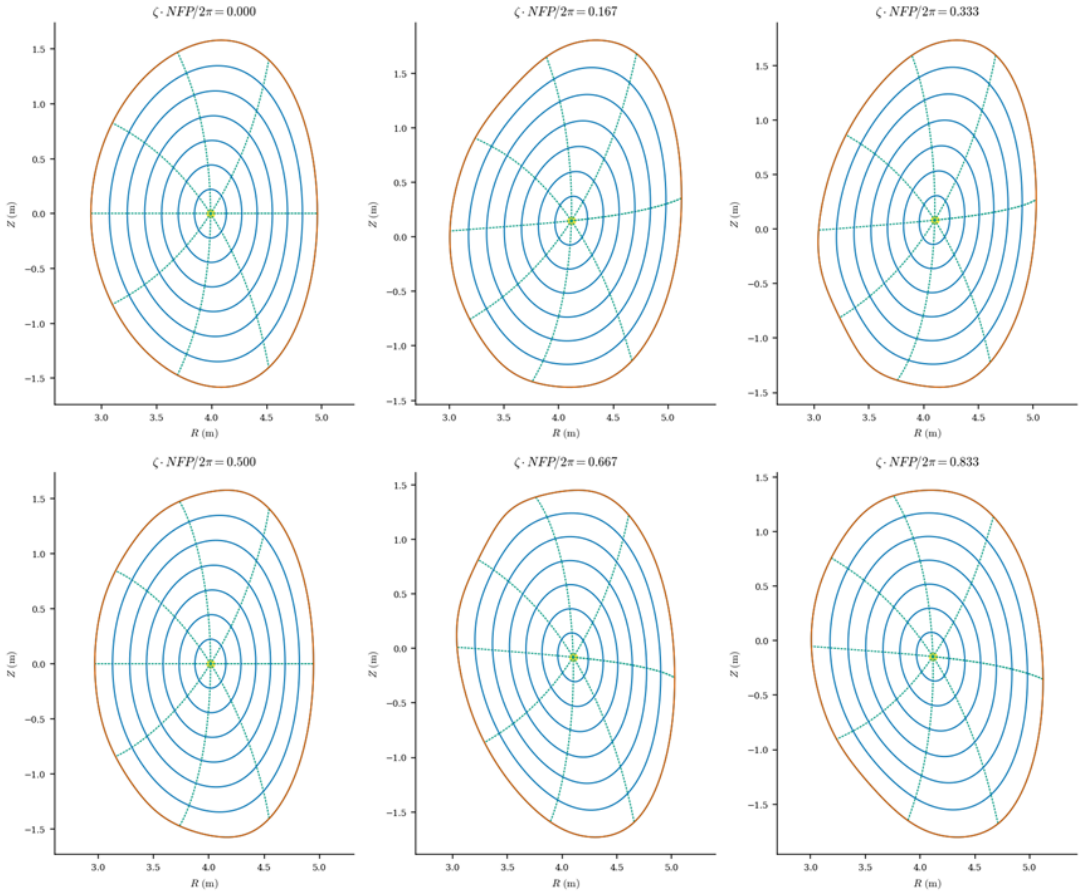
Can be shown to be equivalent to:

$$(J^T J + \alpha I)\mathbf{p} = J^T \mathbf{f}$$

$$\alpha(||\mathbf{p}|| - r) = 0$$

So either full step lies in trust region, or reduces problem to 1d newton's method to find correct value of α

Poincare Section Boundary Condition



- BC: ζ=0 Poincare section from Solov'ev tokamak solution
- Increased pressure profile by 60%
- Boundary deforms non-axisymmetrically to find stellarator equilibrium

Solving force balance does not enforce $\nabla \cdot J = 0$

Ampere's Law: $\boldsymbol{\nabla} \times \boldsymbol{B} = \mu_0 \boldsymbol{J}$

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

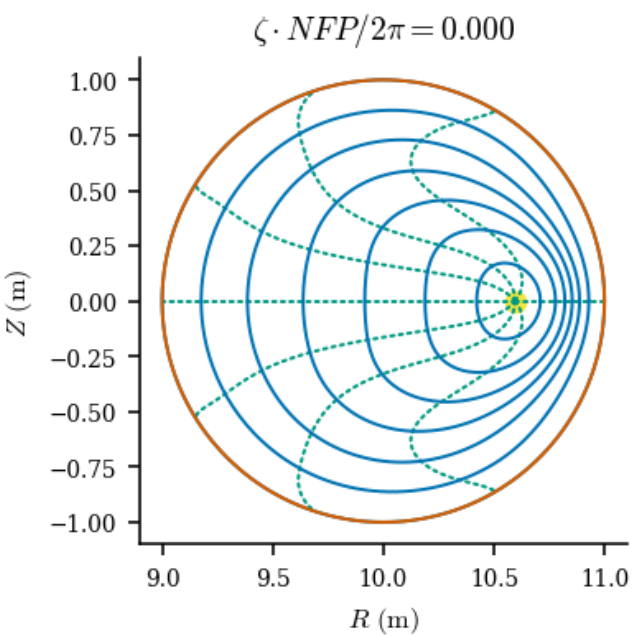
$$J^\theta = \frac{1}{\mu_0\sqrt{g}}\left(\partial_\zeta B_\rho - \partial_\rho B_\zeta\right)$$

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

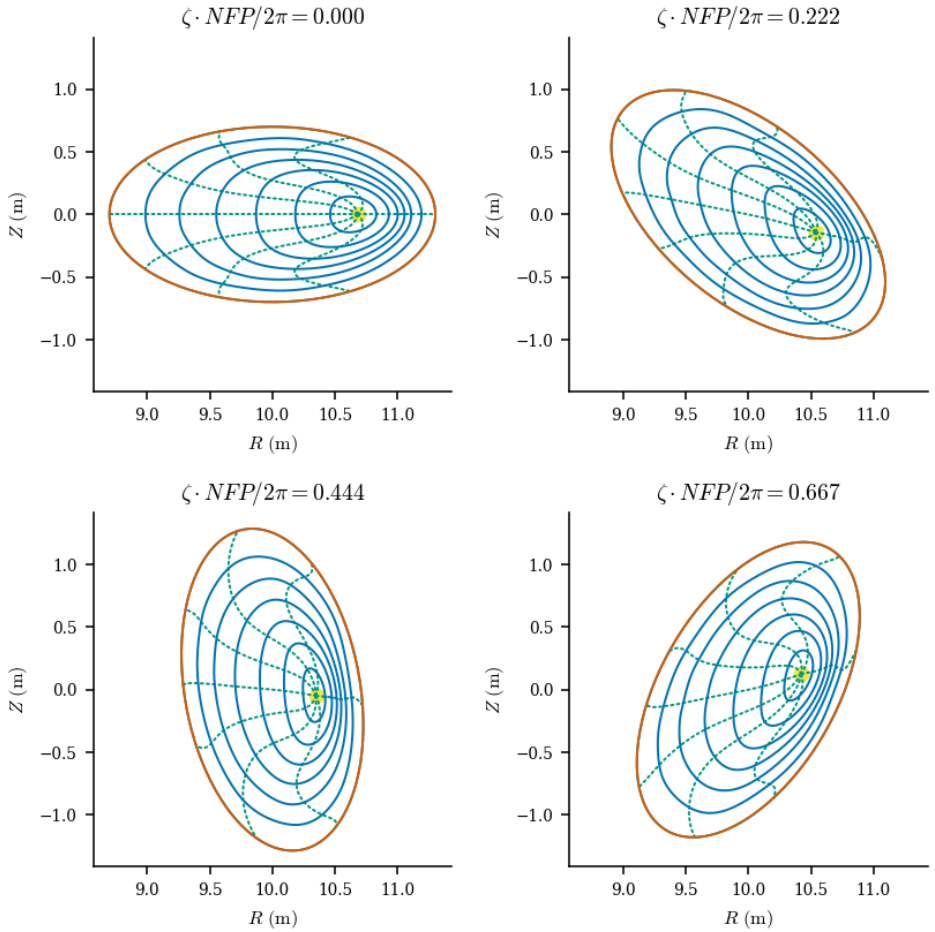
$$\boldsymbol{J}(\rho,\theta,\zeta) = \boldsymbol{J}(R(\rho,\theta,\zeta), Z(\rho,\theta,\zeta), \lambda(\rho,\theta,\zeta), \iota(\rho))$$

Order of Derivatives	Variables	Equations
0	R, Z	
1	$\partial_i R, \partial_i Z \rightarrow \boldsymbol{B}$	$\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$
2	$\partial_{ij} R, \partial_{ij} Z \rightarrow \boldsymbol{J}$	$\boldsymbol{J} \times \boldsymbol{B} = \boldsymbol{\nabla} p$
3	$\partial_{ijk} R, \partial_{ijk} Z$	$\boldsymbol{\nabla} \cdot \boldsymbol{J} = 0$

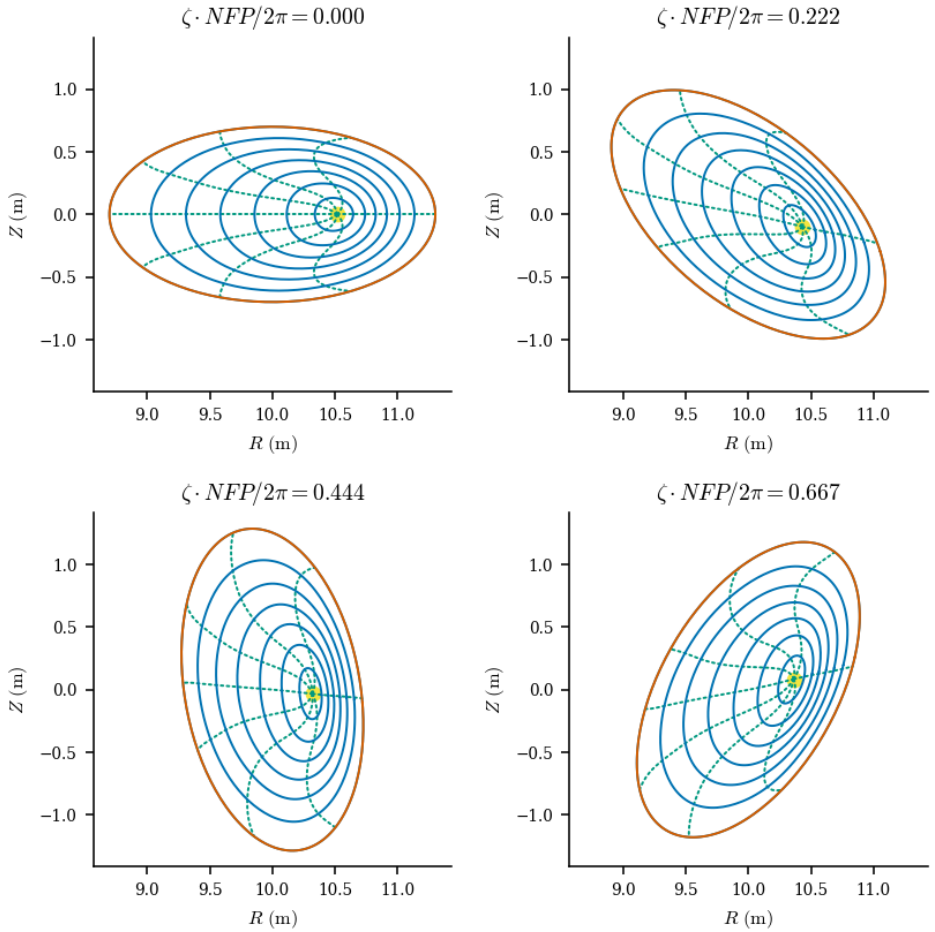
Initial tokamak solution

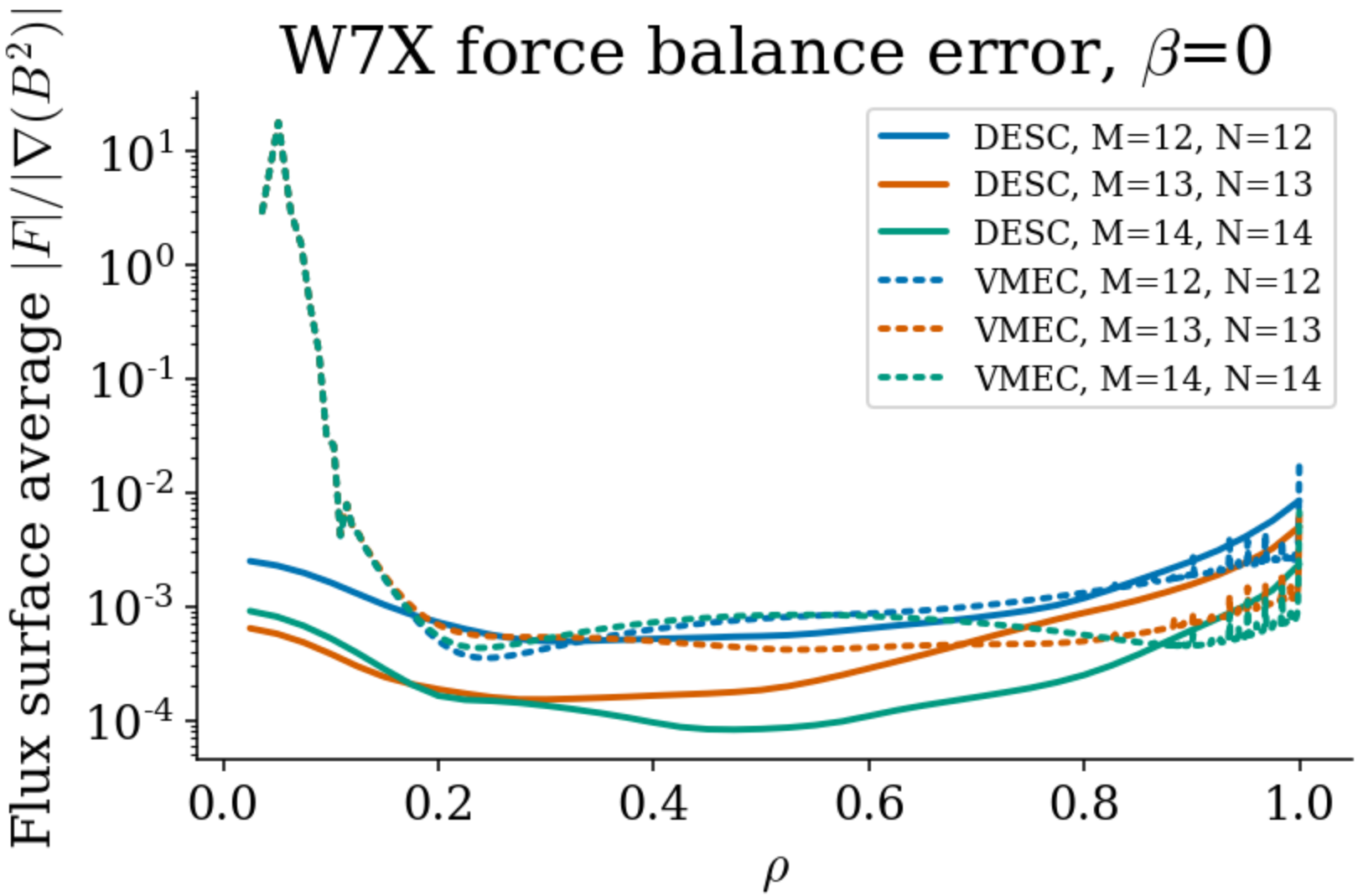


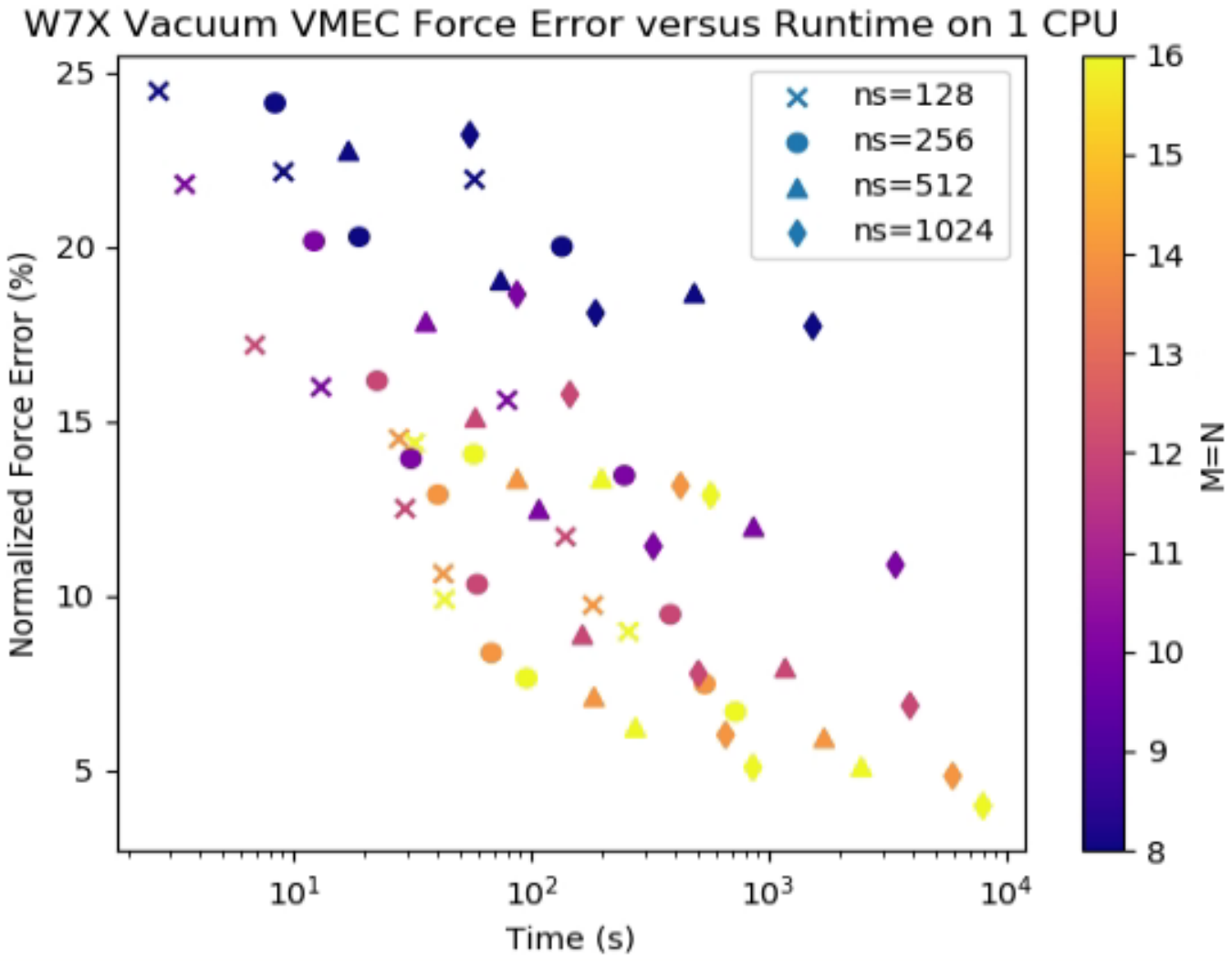
2nd order boundary perturbation

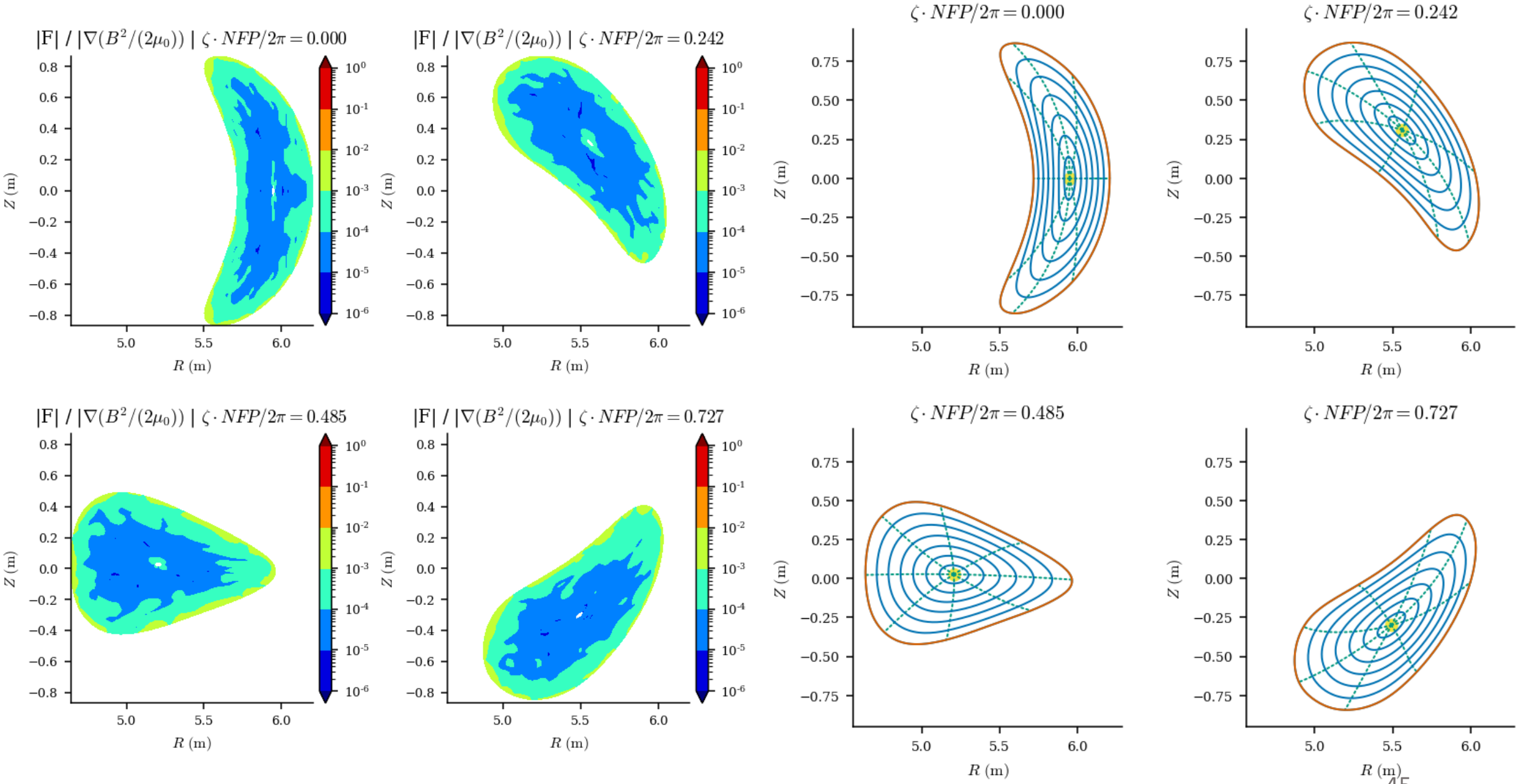


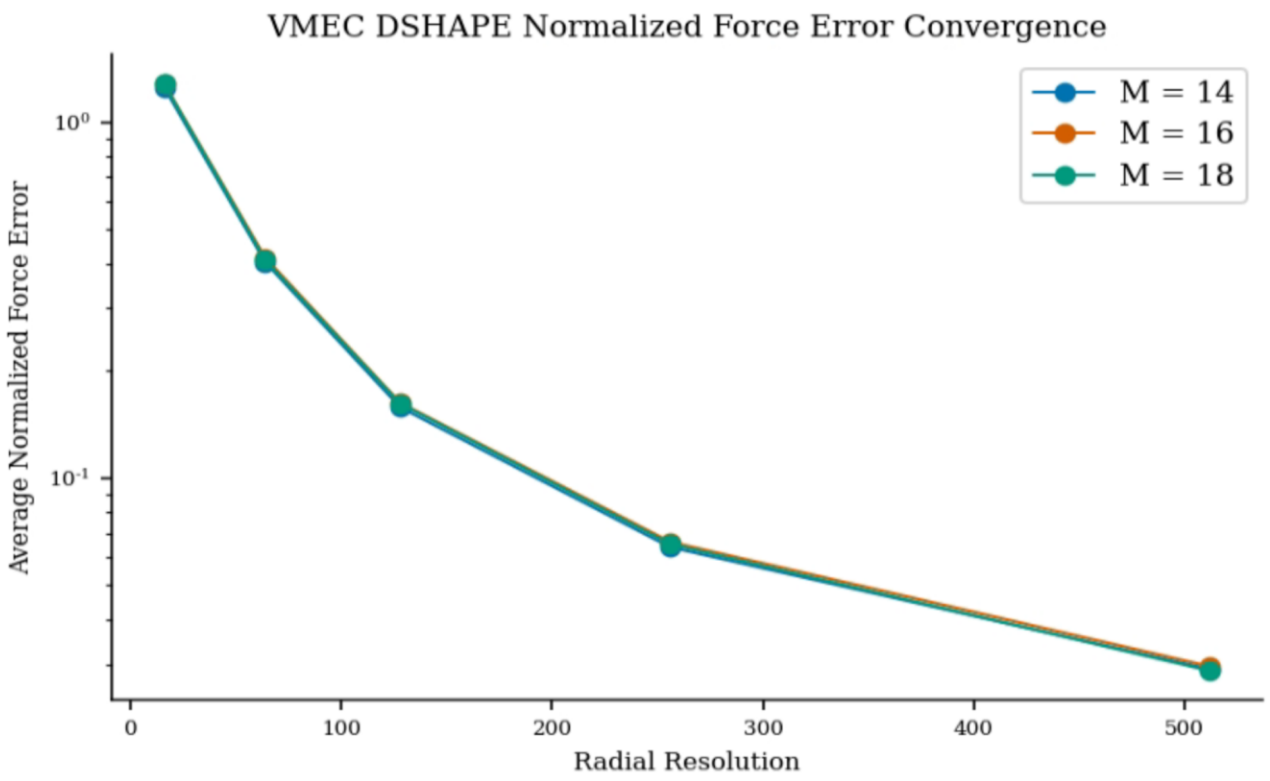
True stellarator solution

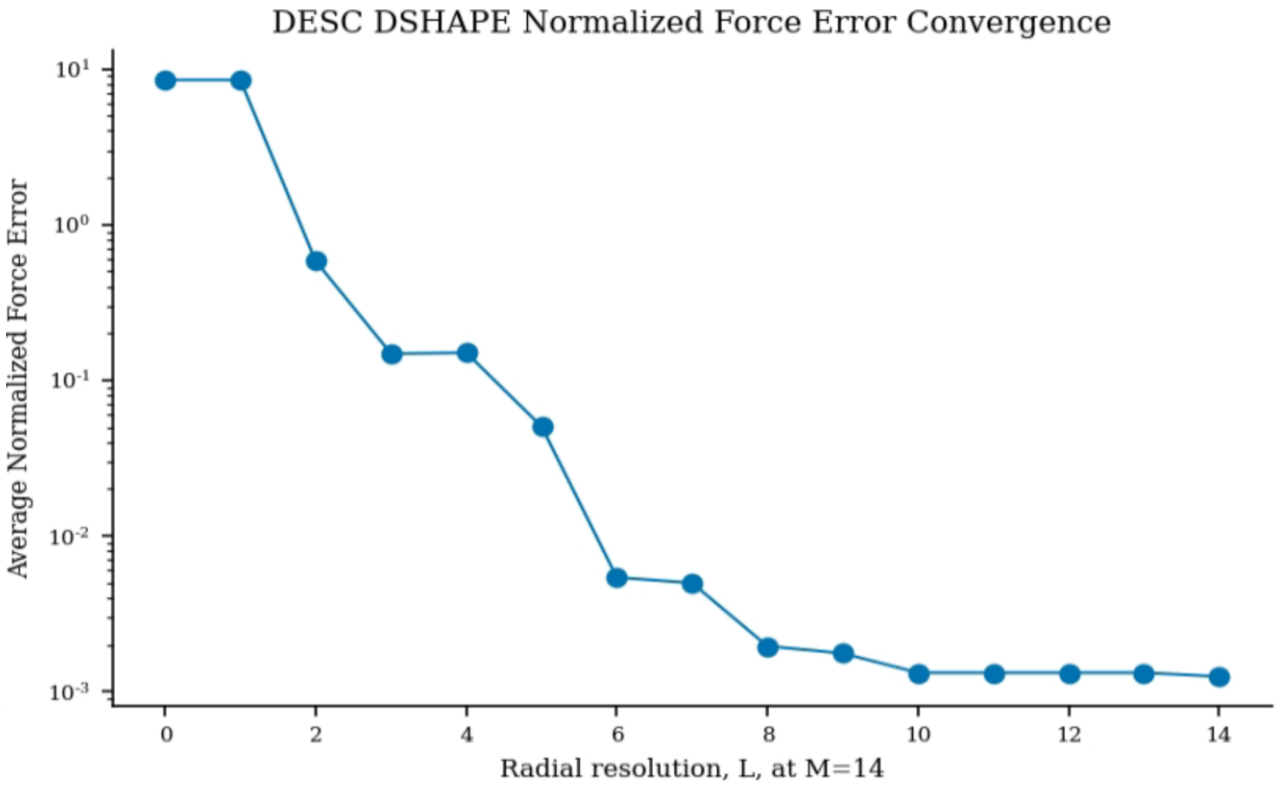


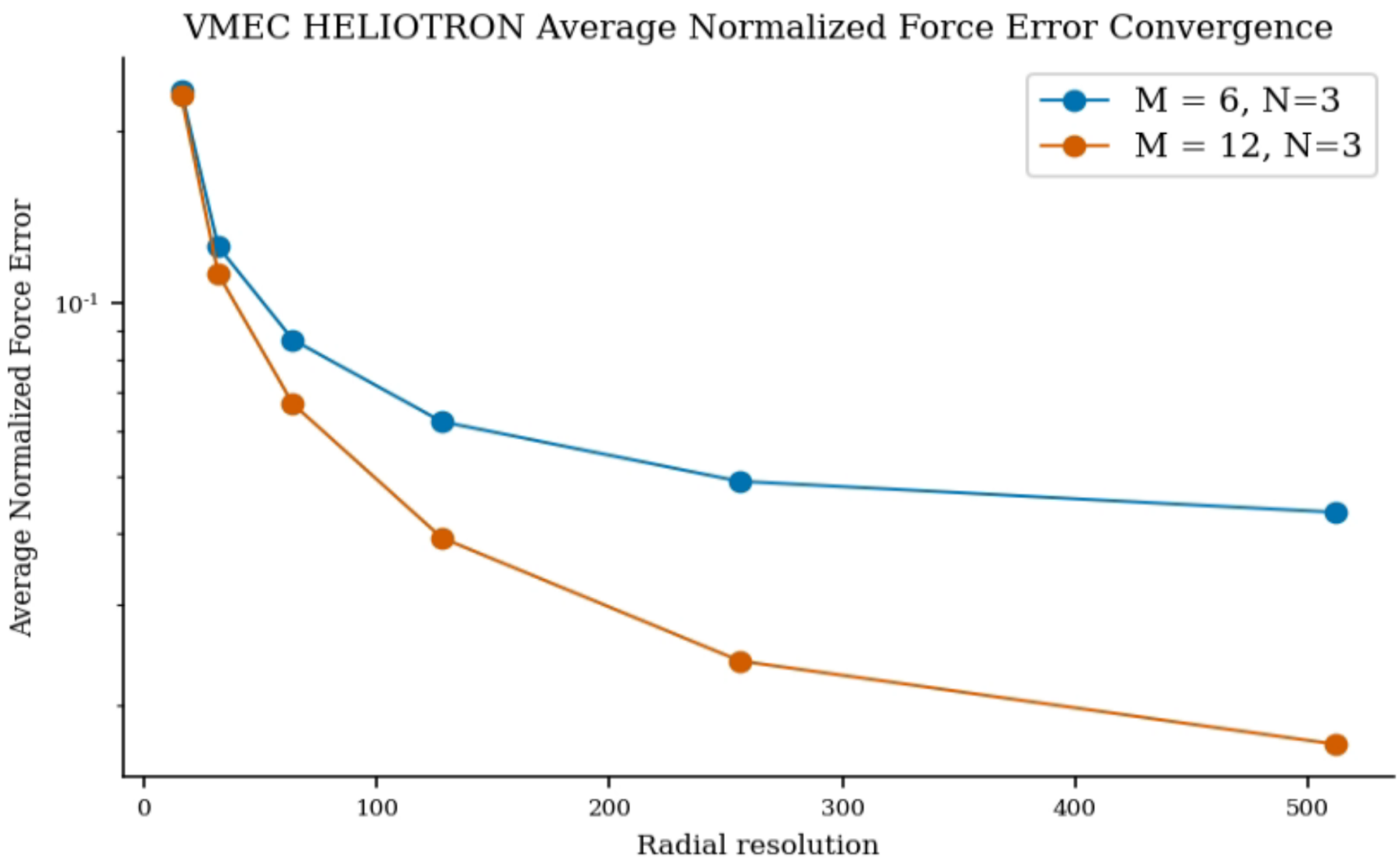

W7X force balance error, β=0
Flux surface average $|F|/|\nabla(B^2)|$
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0.0
0.2
0.4
0.6
0.8
1.0
ρ
DESC, M=12, N=12
DESC, M=13, N=13
DESC, M=14, N=14
VMEC, M=12, N=12
VMEC, M=13, N=13
VMEC, M=14, N=14


W7X Vacuum VMEC Force Error versus Runtime on 1 CPU
Normalized Force Error (%)
Time (s)
ns=128
ns=256
ns=512
ns=1024
M=N
25
20
15
10
5
10^1
10^2
10^3
10^4
16
15
14
13
12
11
10
9
8


|F| / |∇(B²/(2μ₀))| ζ · NFP/2π = 0.000
|F| / |∇(B²/(2μ₀))| ζ · NFP/2π = 0.242
|F| / |∇(B²/(2μ₀))| ζ · NFP/2π = 0.485
|F| / |∇(B²/(2μ₀))| ζ · NFP/2π = 0.727
ζ · NFP/2π = 0.000
ζ · NFP/2π = 0.242
ζ · NFP/2π = 0.485
ζ · NFP/2π = 0.727
Z (m)
R (m)


VMEC DSHAPE Normalized Force Error Convergence
M = 14
M = 16
M = 18
Average Normalized Force Error
10^0
10^{-1}
0
100
200
300
400
500
Radial Resolution


DESC DSHAPE Normalized Force Error Convergence
Average Normalized Force Error
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
0
2
4
6
8
10
12
14
Radial resolution, L, at M=14


VMEC HELIOTRON Average Normalized Force Error Convergence
M = 6, N=3
M = 12, N=3
Average Normalized Force Error
10^{-1}
0
100
200
300
400
500
Radial resolution

Example Inputs

Axisymmetric
"D-shaped" Tokamak

$$R^b = 3.51 - \cos\theta + 0.106\cos 2\theta$$
$$Z^b = 1.47\sin\theta + 0.16\sin 2\theta$$
$$\iota = 1 - 0.67\rho^2$$
$$p = 1.65 \times 10^3 (1 - \rho^2)^2$$
$$\psi_a = 1$$

Non-Axisymmetric
high-beta Heliotron

$$R^b = 10 - \cos\theta - 0.3\cos(\theta - 19\phi)$$
$$Z^b = \sin\theta - 0.3\sin(\theta - 19\phi)$$
$$\iota = 1.5\rho^2 + 0.5$$
$$p = 3.4 \times 10^3 (1 - \rho^2)^2$$
$$\psi_a = 1$$