DESC Suite: Integrated Stellarator Optimization

Egemen Kolemen, Prof. Princeton Univ./PPPL

With Daniel Dudt, Rory Conlin, Dario Panici, Patrick Kim, Kaya Unalmis, Eduardo Rodriguez, Aza Jalalvand

Princeton Plasma Control
control.princeton.edu

PRINCETON UNIVERSITY

Simons NYC Meeting / May 2023

PPPL
PRINCETON PLASMA PHYSICS LABORATORY

What is the ideal way to optimize stellarators?

- **Constraints g(x):**
  - MHD equilibrium
  - *Physicist insight: Analytical calculations (e.g. NEA)*
  - *Engineer insight: e.g. A<5, …*
- **Objectives f(x):**
  - Quasi-symmetry
  - Turbulence
  - …
- *Physicist/engineer insight: relative importance of f(x)*

What is the ideal way to optimize stellarators?

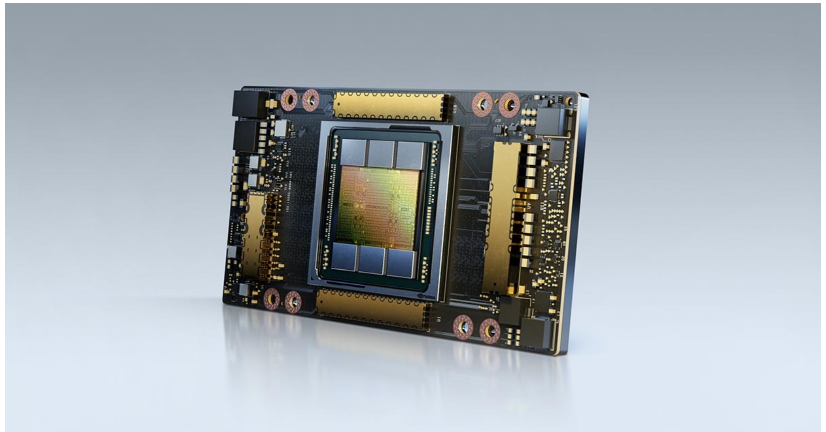
- We don’t exactly know what we want
- We are not looking for one optimum but series of optima in the space defined by the physicist/engineer
- A map $g_{physicist}$ ➔ Optima

Then Call A Fast Code

$$\min_x f(x)$$

$$\text{subject to} \quad \begin{aligned} g_{eq}(x) &= 0 \\ g_{ineq}(x) &\geq 0 \end{aligned}$$

➔



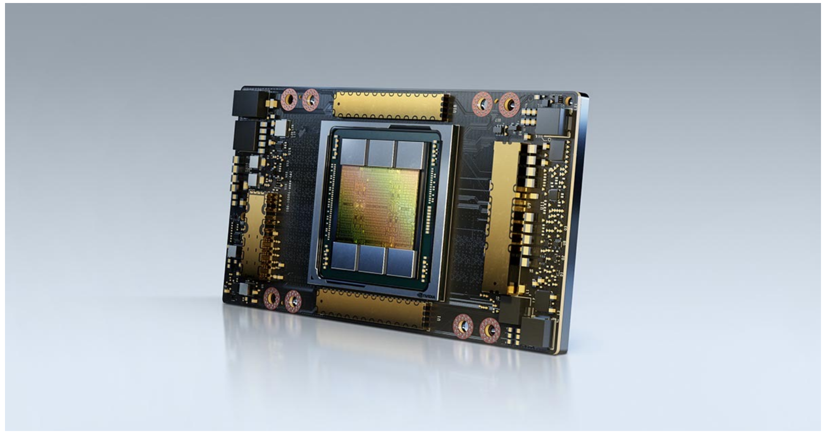
Fast= GPU + Jacobian

Then Call A Fast Code

$$\min_x f(x)$$

$$\text{subject to} \quad g_{eq}(x) = 0, \quad g_{ineq}(x) \geq 0$$

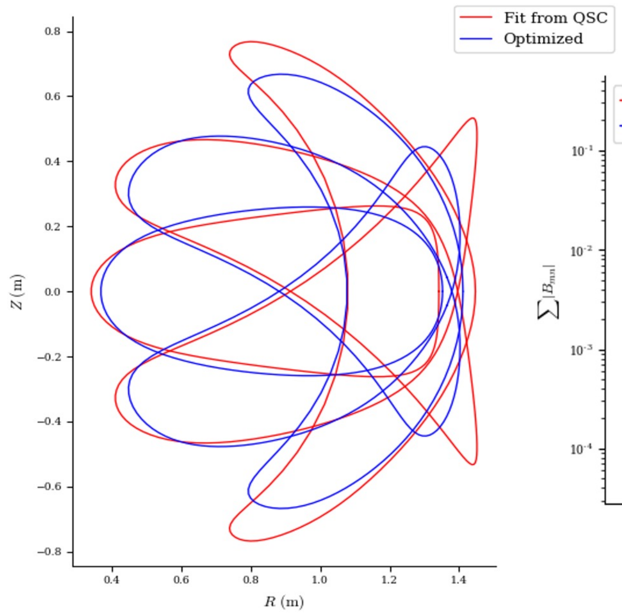
➔

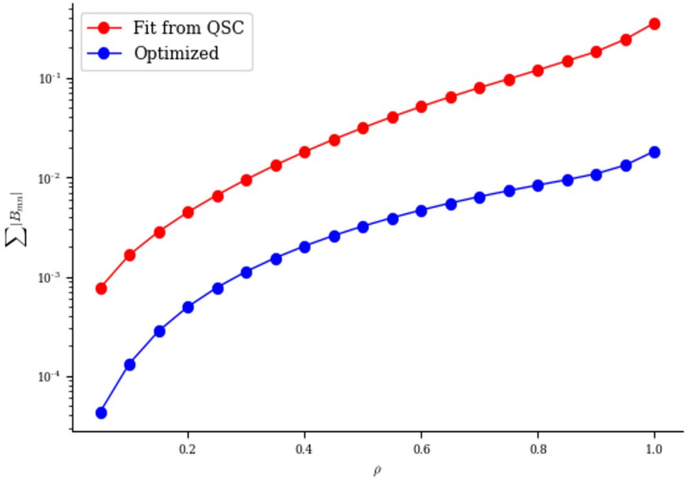


Fast= GPU + Jacobian

Constraints: $g_{physicist}$ =Fix NEA ➔
+ g= MHD Eq.

Optimize remaining volume



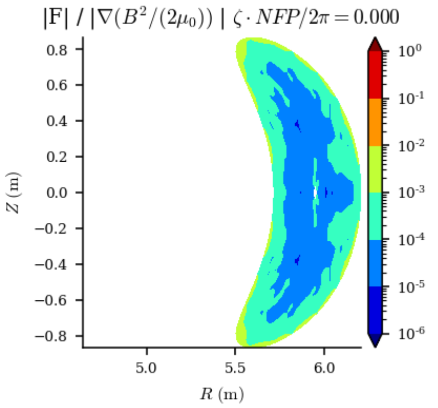


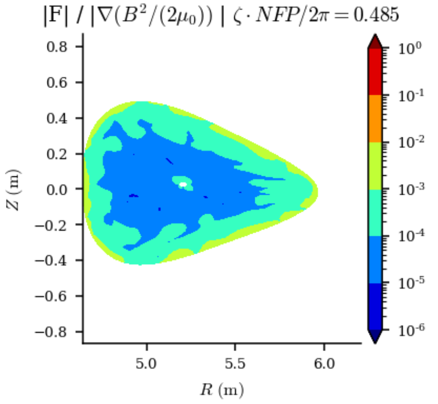
Final Take: Fix the core, do proper constrained optimization

1. Don't specify R, Z surface Fourier! It is 2x the needed # param. on surface (x5 Poincare)
   - Why specify looping/intersecting, over constrained parameters we have no intuition for? And %100 will give non-nested solutions?
2. Specify core with NEA (maybe +-%10 inequality constraint): underconstrained
   - Extra: if you want QI specify the phase space parameterization.
3. Stop the loopy optimization (perturb > project)!
   - Use Augmented Lagrangian or Interior Point methods
   - Force balance will be satisfied not with a loop within a loop but by the optimizer
4. Problem is way simpler! Physicists just need to write their cost function for high level physics (turbulence, radiation,…)

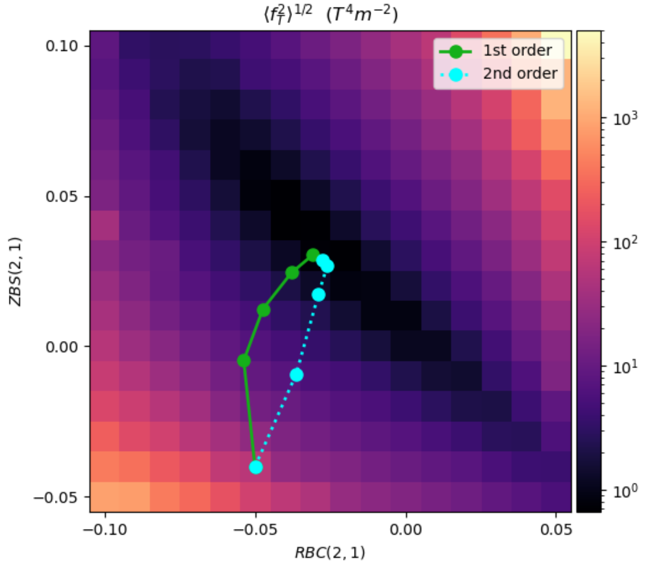
DESC is a new tool for stellarator optimization

Accurate Equilibria

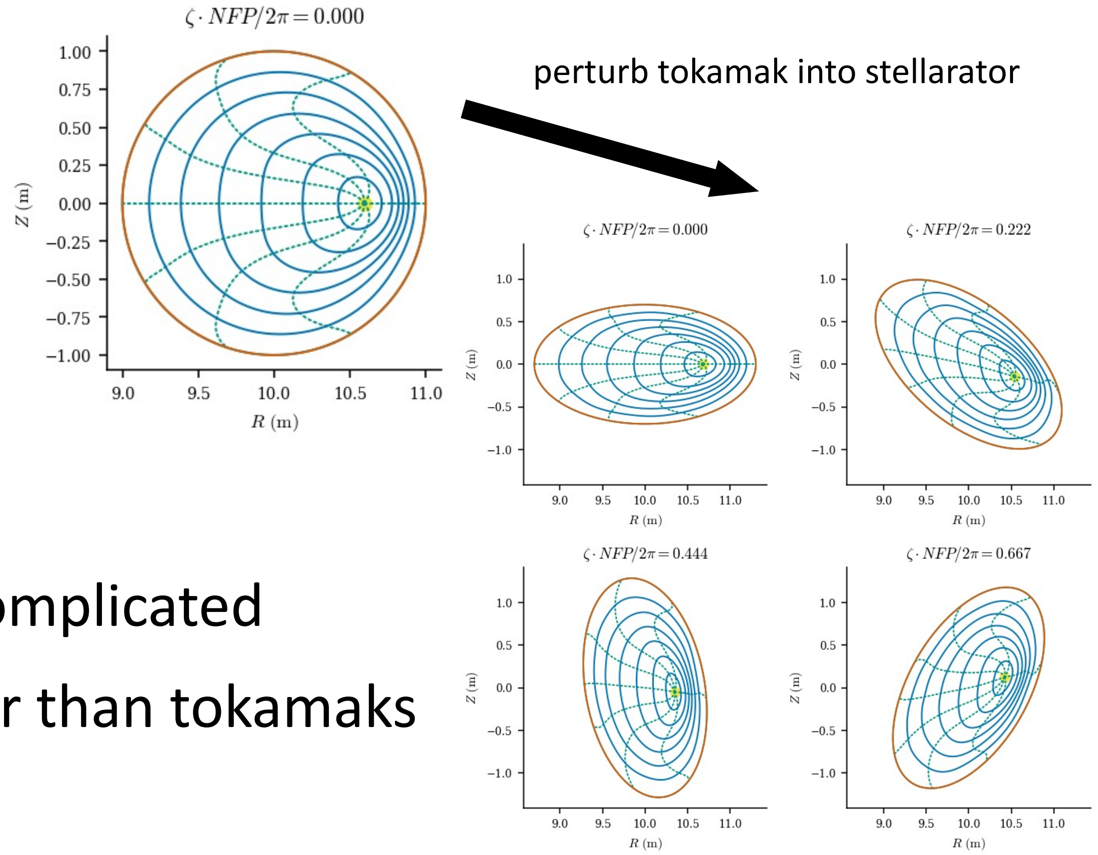




Fast Optimization

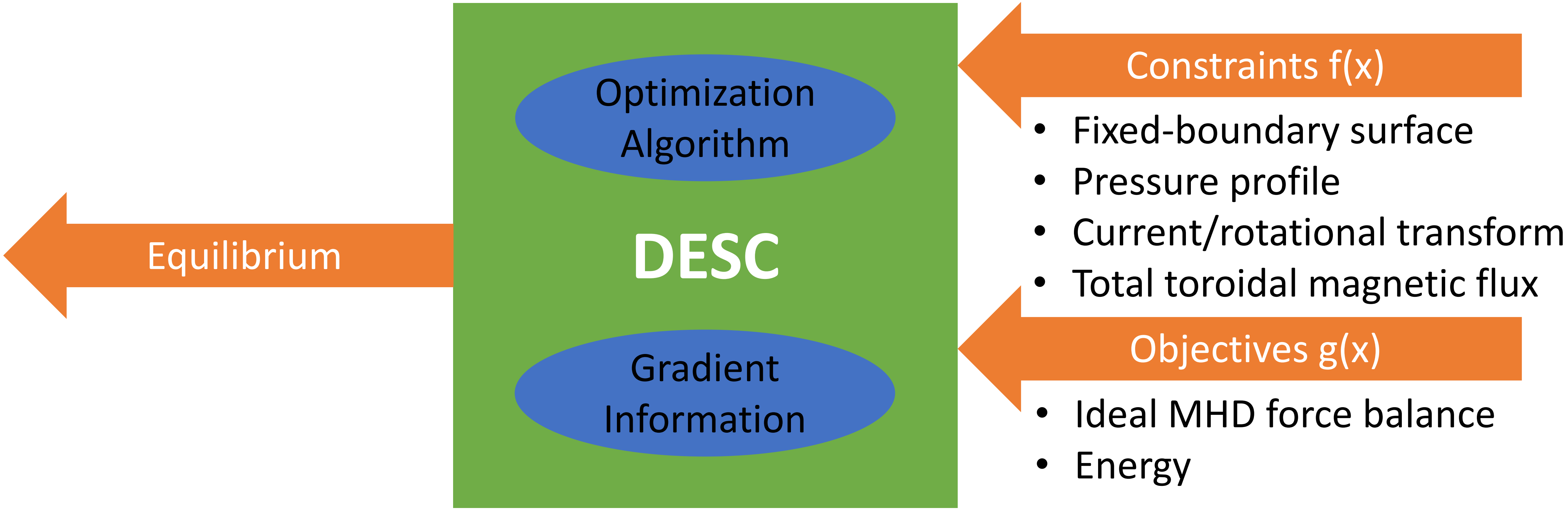


Phase-Space Connections

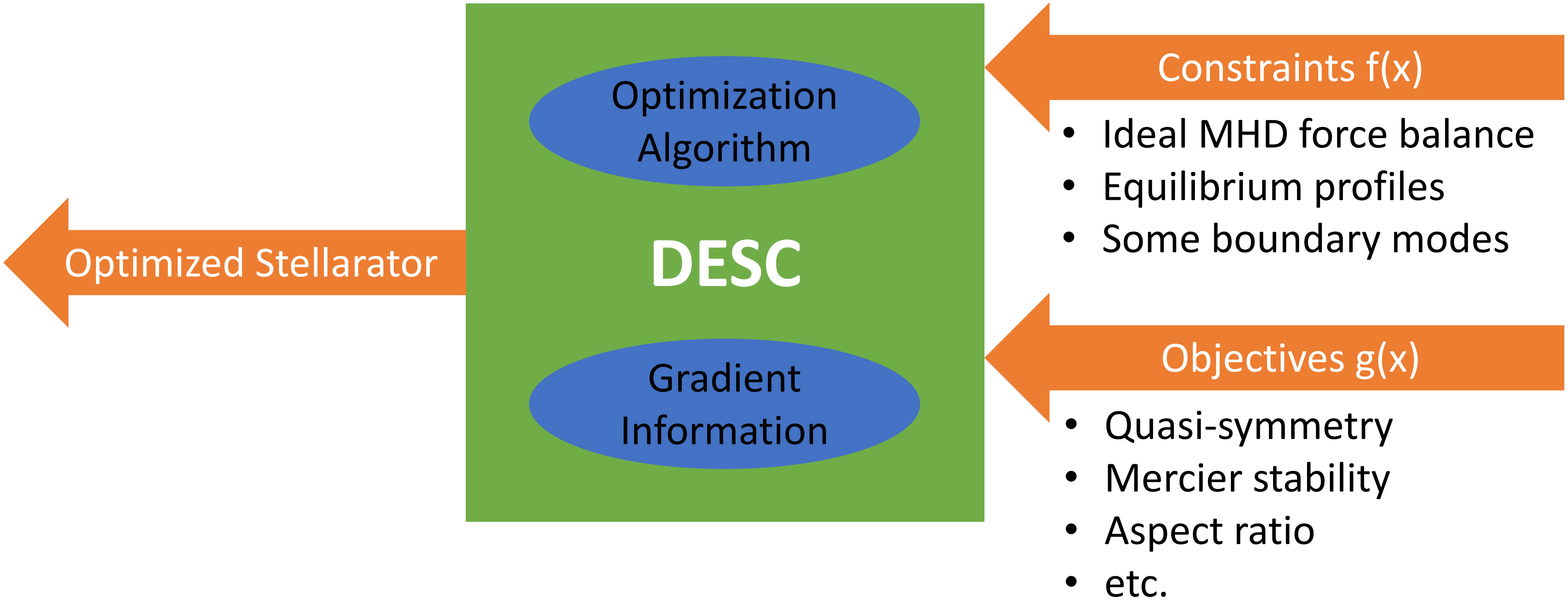


- Stellarator equilibria are complicated
- Design space is much larger than tokamaks

A flexible stellarator optimization suite



A flexible stellarator optimization suite



Why do we need *another* stellarator code?

Equilibrium solvers: VMEC, NEAR, PIES, HINT, SPEC, GVEC, etc.

Optimization codes: STELLOPT, ROSE, WISTELL, SIMSOPT, etc.

1. Better understand the solution space of stellarator equilibria
2. Integrate the equilibrium solver with optimization tools
3. Avoid Jacobian approximations, near-axis expansions, low-β expansions, etc.
4. Use modern numerical methods and scientific computing practices

Developed with the following design principles:

1. **Simple user interface**
   - Open-source Python code
   - Well documented
   - High test coverage
   - Easy to install
2. **Local error quantification**
   - Pseudo-spectral (collocation) methods
3. **Properly resolve the magnetic axis**
   - Global basis functions
   - Zernike polynomials
4. **Exact derivatives of all objectives**
   - Automatic differentiation
5. **Hardware agnostic**
   - Run on CPUs, GPUs, and TPUs
6. **Extendable to new applications**
   - Modular & flexible code structure

Zernike spectral basis inherently satisfies boundary conditions at the magnetic axis

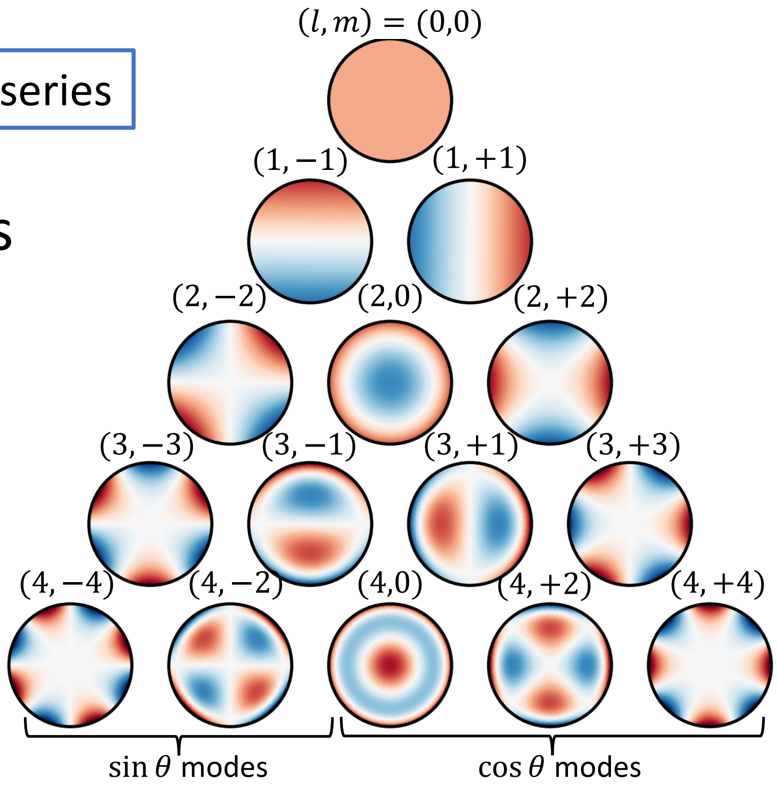
spectral coefficients — Zernike polynomials — Fourier series

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

- Periodic boundary conditions for poloidal & toroidal angles
- Satisfies analyticity conditions at the magnetic axis:

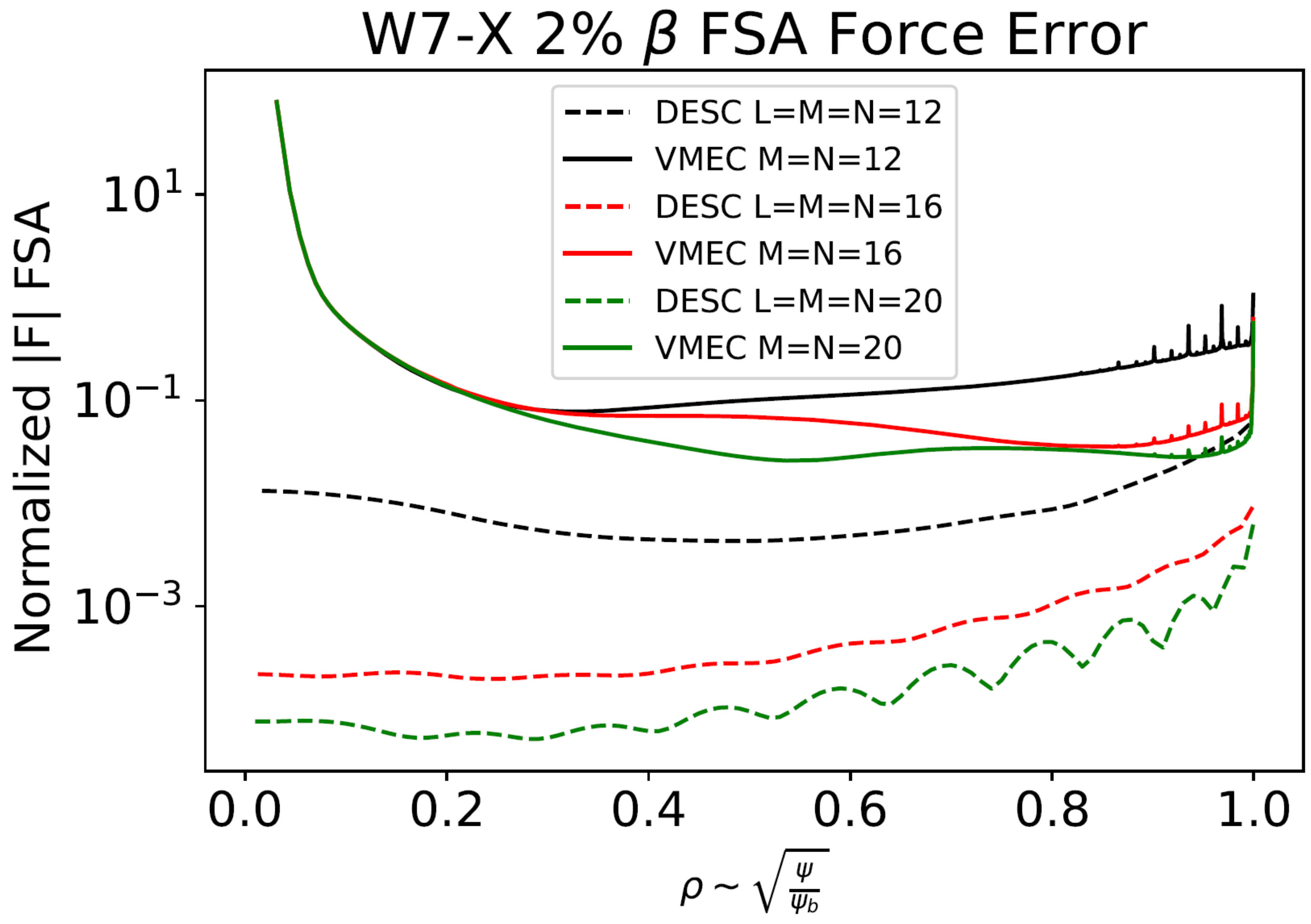
$$f(\rho,\theta) = \sum_m \rho^m\left(a_{m,0} + a_{m,2}\rho^2 + \cdots\right)\cos(m\theta) + \sum_m \rho^m\left(b_{m,0} + b_{m,2}\rho^2 + \cdots\right)\sin(m\theta)$$

- Exponential convergence (if solution exists and is smooth)

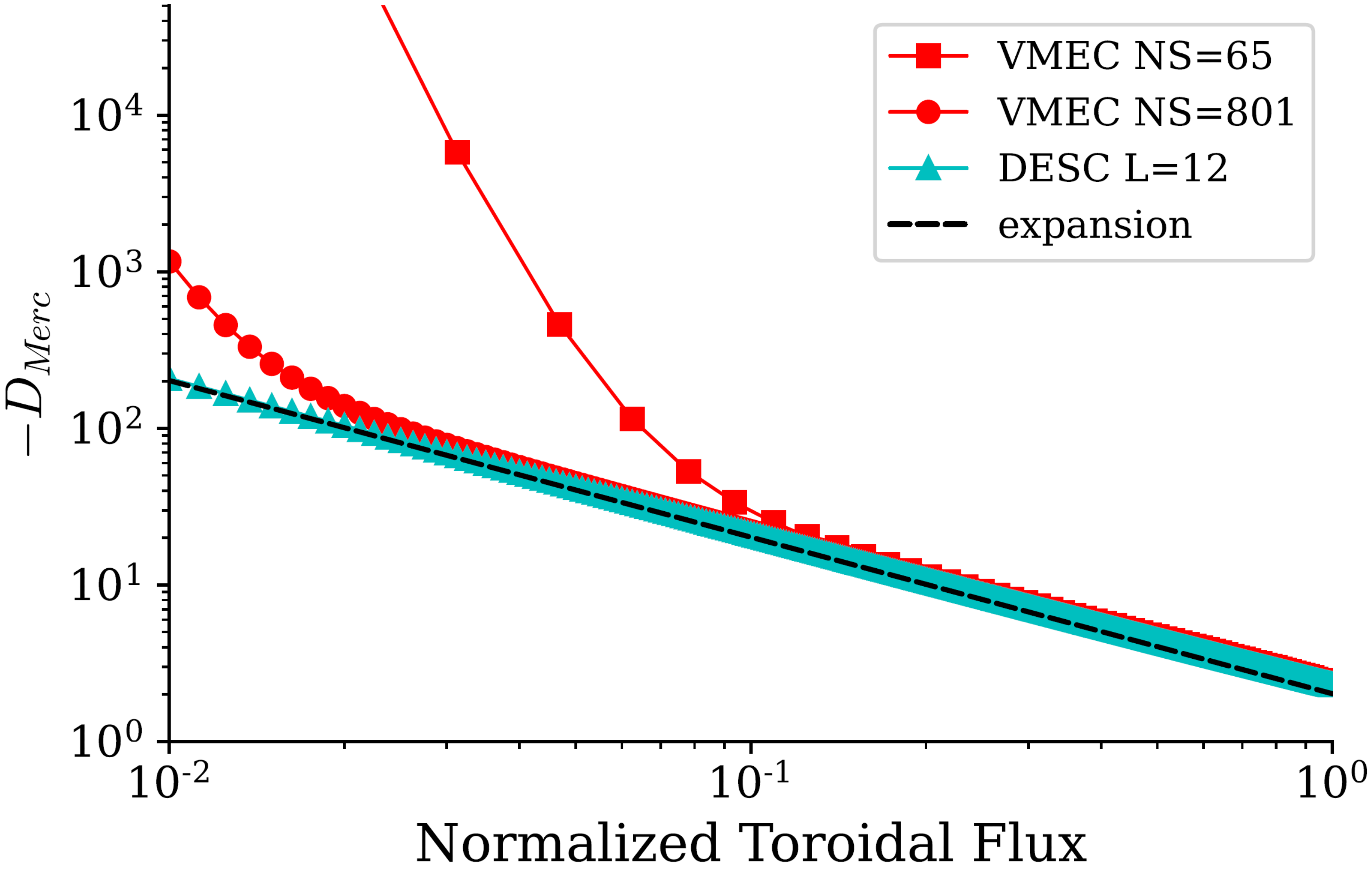


Boyd et al., *J. Comput. Phys.* (2011)
Lewis et al., *J. Math. Phys.* (1990)

Spectral methods yield more accurate equilibrium solutions



Accurately resolving the magnetic axis is important for stability calculations



VMEC requires high radial resolution to resolve axis

DCON3D collaboration

<u>Run times:</u>

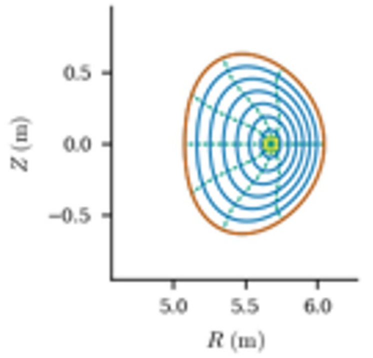
- DESC = 0.2 GPU-hours (NVIDIA A100)
- VMEC = 5.2 CPU-hours (AMD Opteron 6276)

Landreman & Sengupta, *J. Plasma Phys.* (2019)

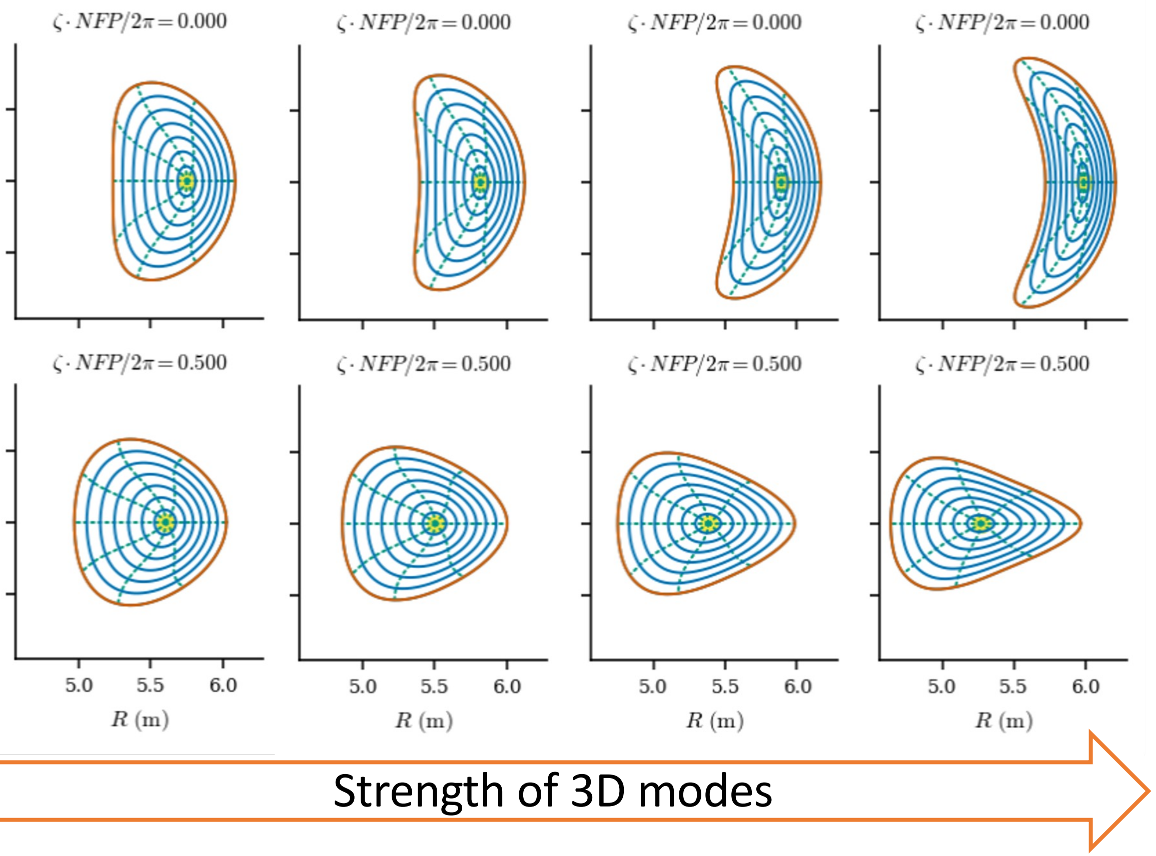
Continuation method example: from tokamak to 3D stellarator boundary

Initial solution

axisymmetric boundary (tokamak)



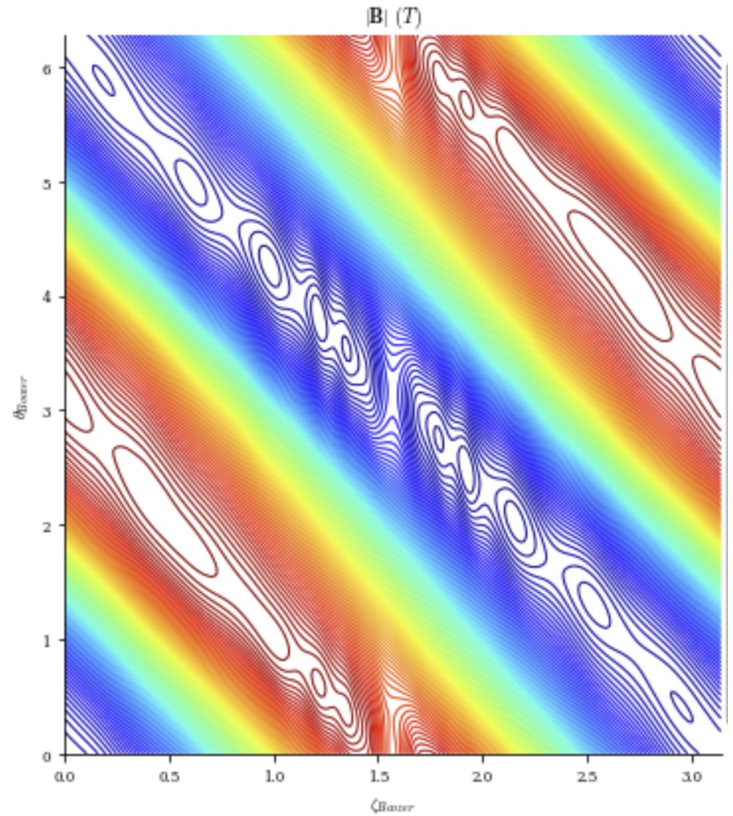
Intermediate solutions

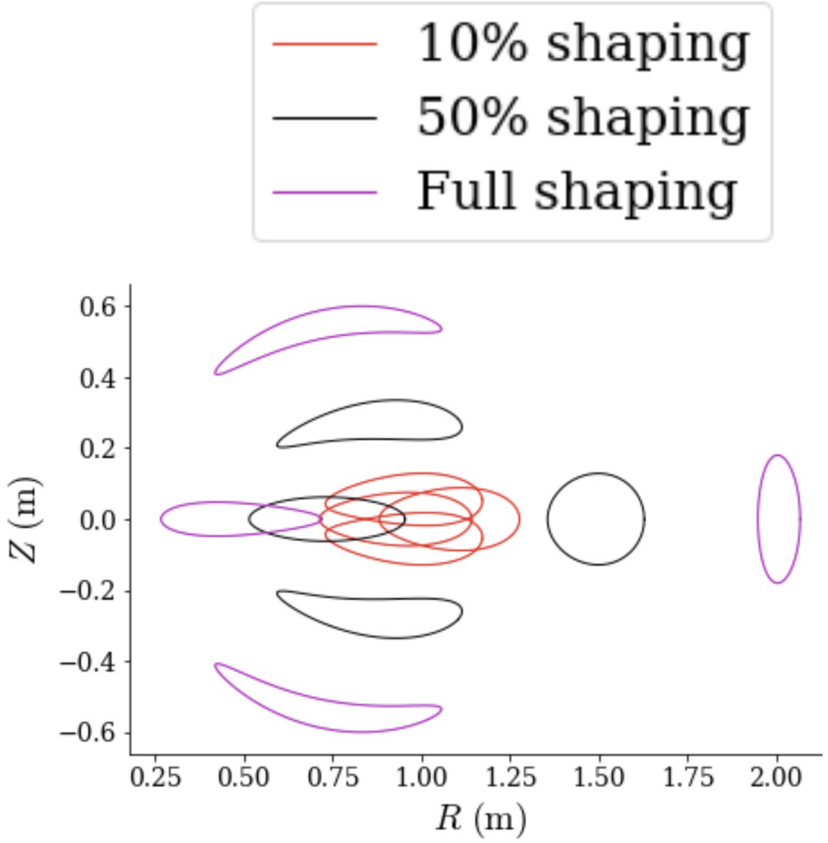


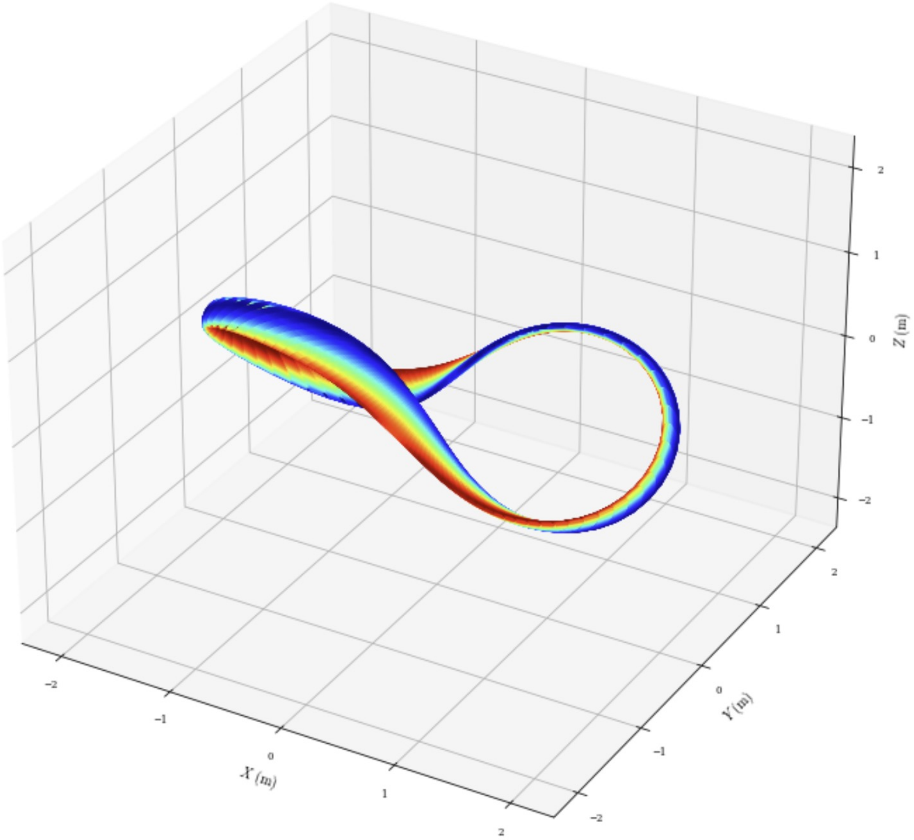
Final solution

strongly shaped stellarator

Solving Highly-Shaped Boundaries in DESC

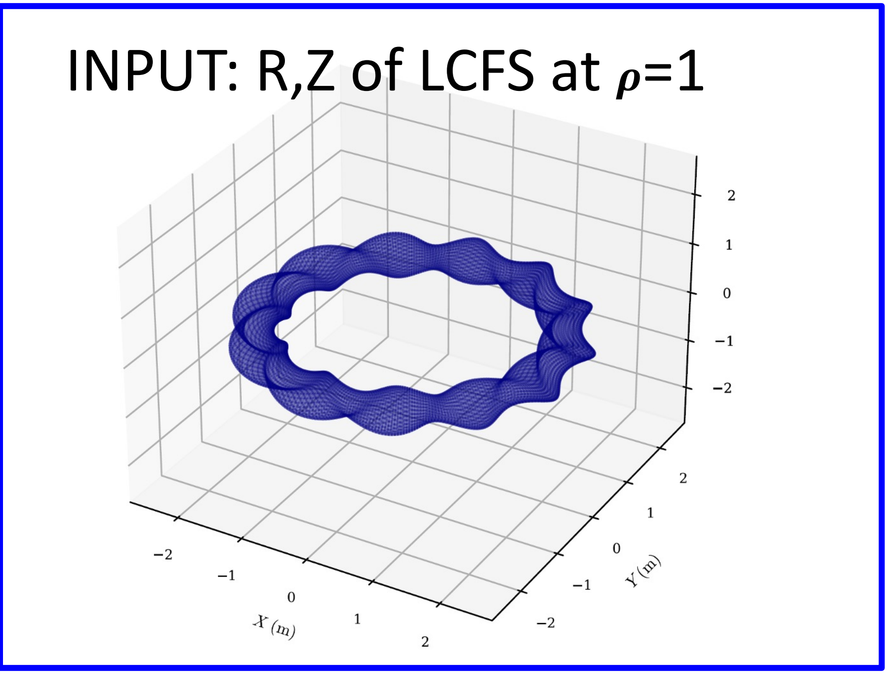






- Equilibrium which SPEC/VMEC have trouble with
- According to Joaquim Loizu

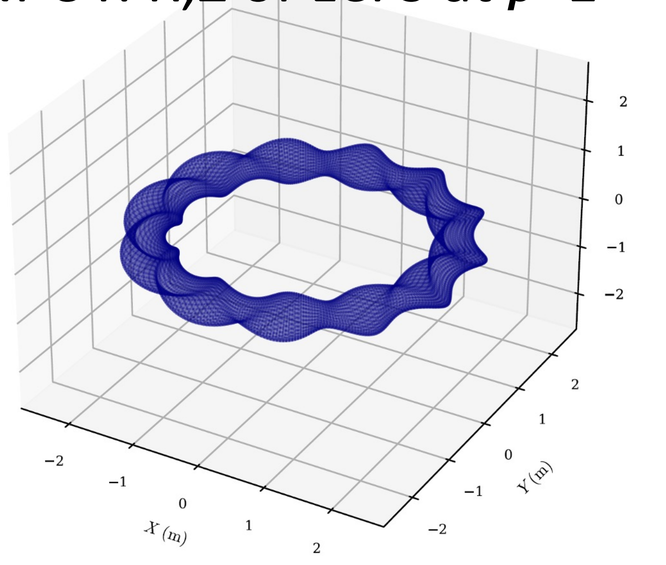
Specifying surface shape is not ideal



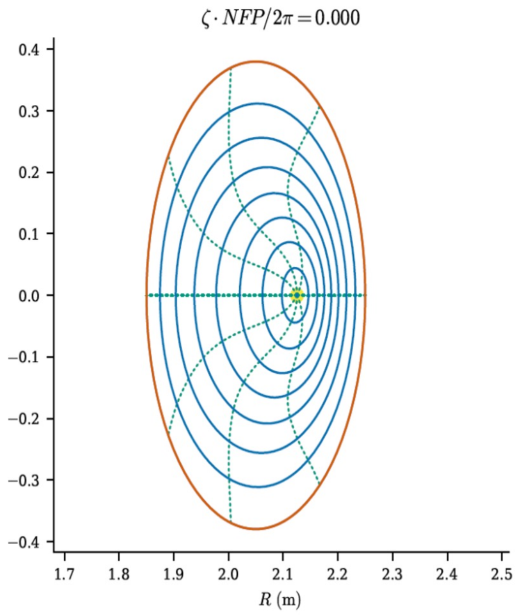
- **Our aim is to optimize (not solve for equil.)**
- **We are not interested in any non-nested solutions**
- **You need n*m parameters to specify a toroidal surface**
- **R, Z Fourier Series need 2*n*m**
- **There are n*m hidden constraints (a pain for optimization)**
- **Loops/intersections occur**
- **There exists ways to represent the problem with lower dimensional setup**

Novel boundary conditions to better parameterize stellarator design space

INPUT: R,Z of LCFS at ρ=1



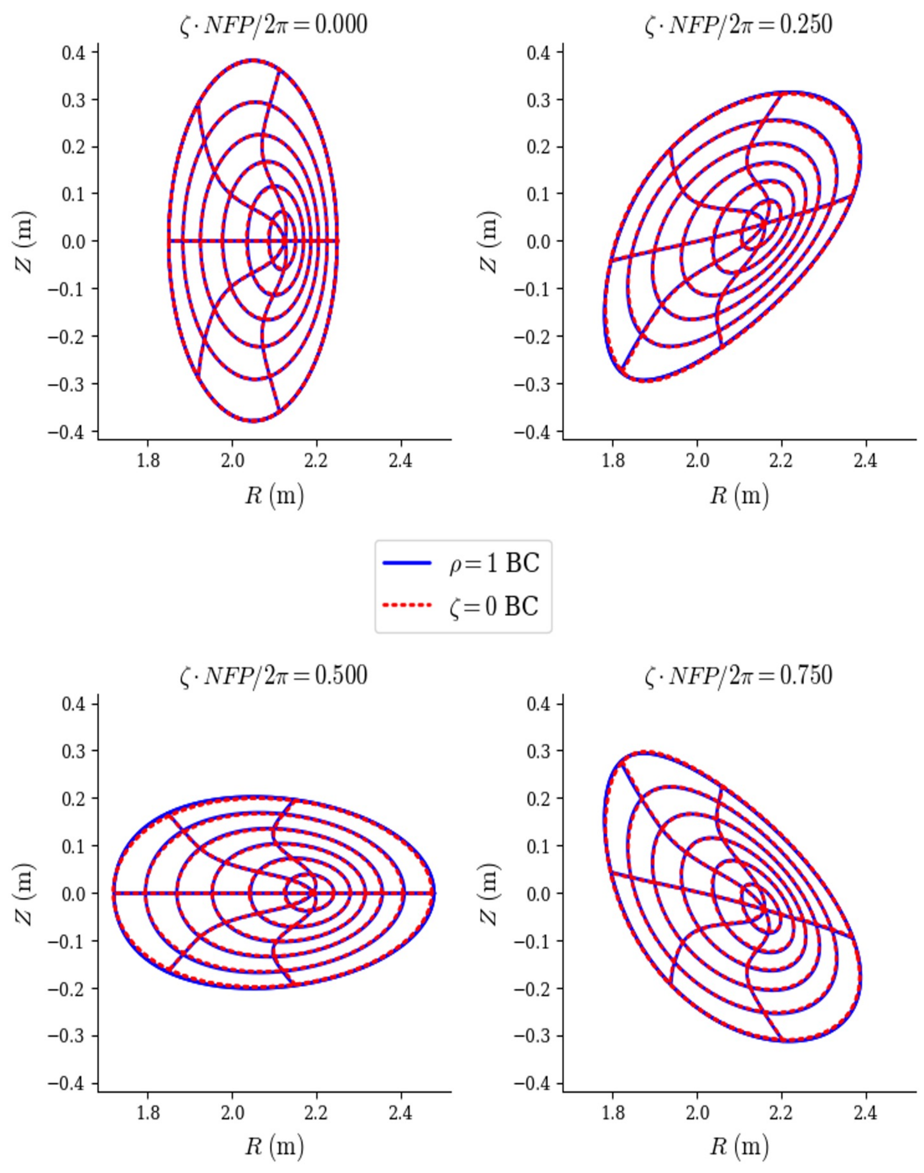
INPUT: R,Z, $\boldsymbol{\lambda}$ of Poincare XS at $\boldsymbol{\zeta}$=0



- **Poincare section requires much fewer number (x5) of input coefficients to describe boundary condition**

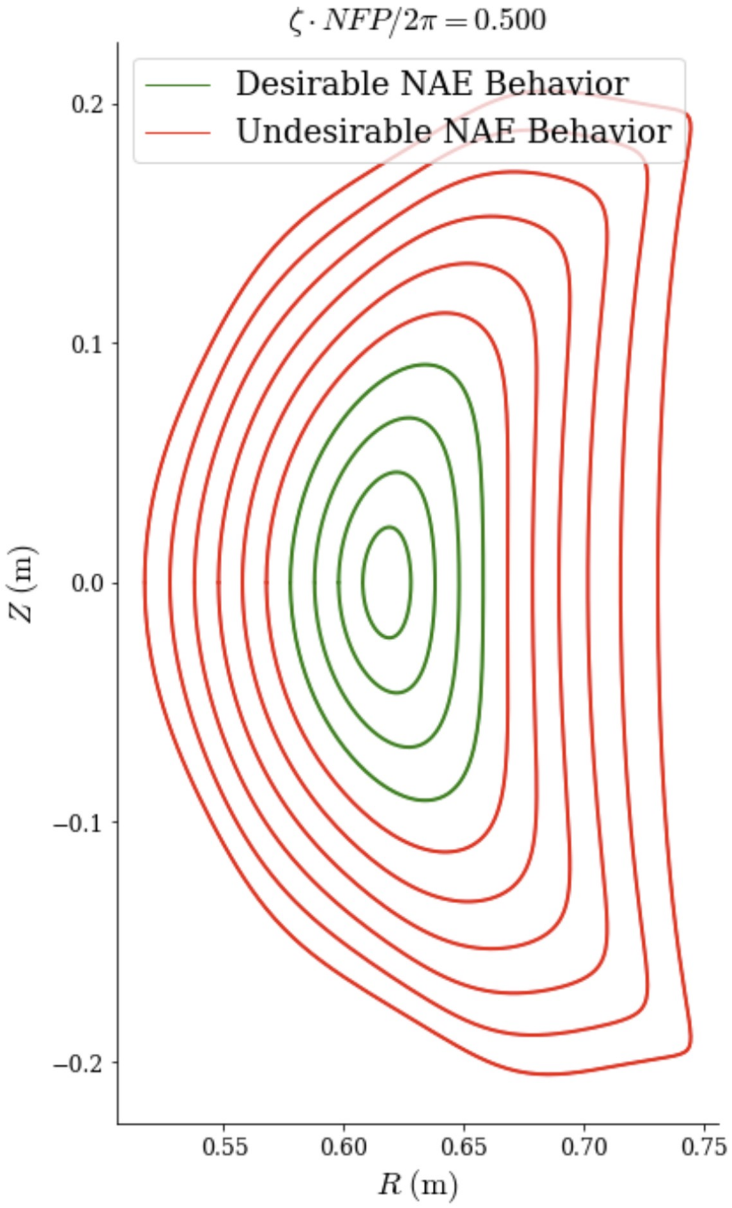
$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$$
$$\boldsymbol{x}(0) = \boldsymbol{x}(2\pi)$$

- **Potentially restrict to only solutions with nested flux surfaces**



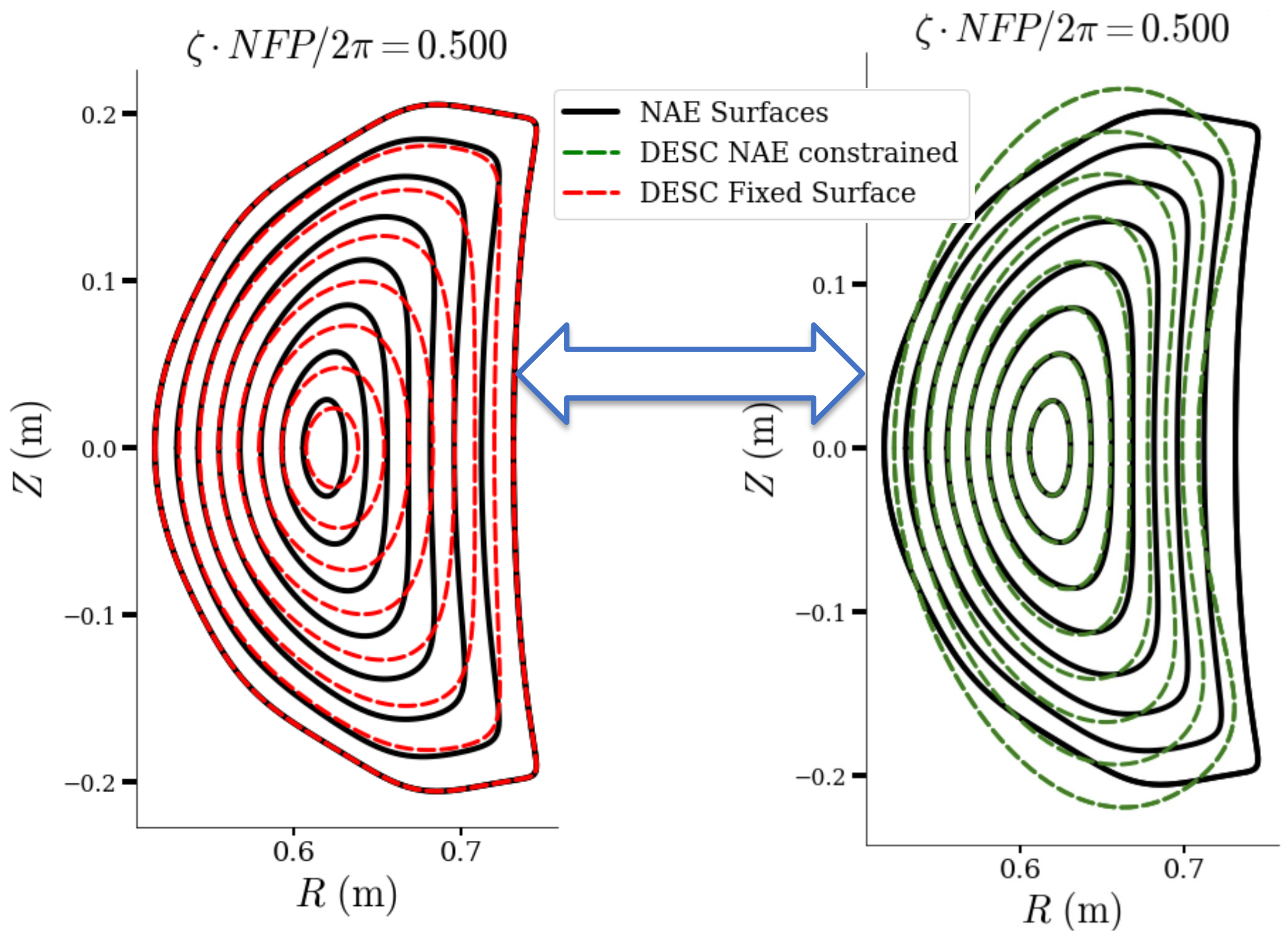
Easy to Fix the Core in DESC

- Idea is to constrain the global equilibrium to have NAE behavior as $\boldsymbol{\rho} \to 0$
  - only use information from NAE where it is most valid
  - Avoid singular behavior present when evaluating at large r
- Map NAE coefficients to Fourier-Zernike modes of DESC to fix $O(\boldsymbol{\rho}^0)$ (axis) and $O(\boldsymbol{\rho}^1)$ behavior



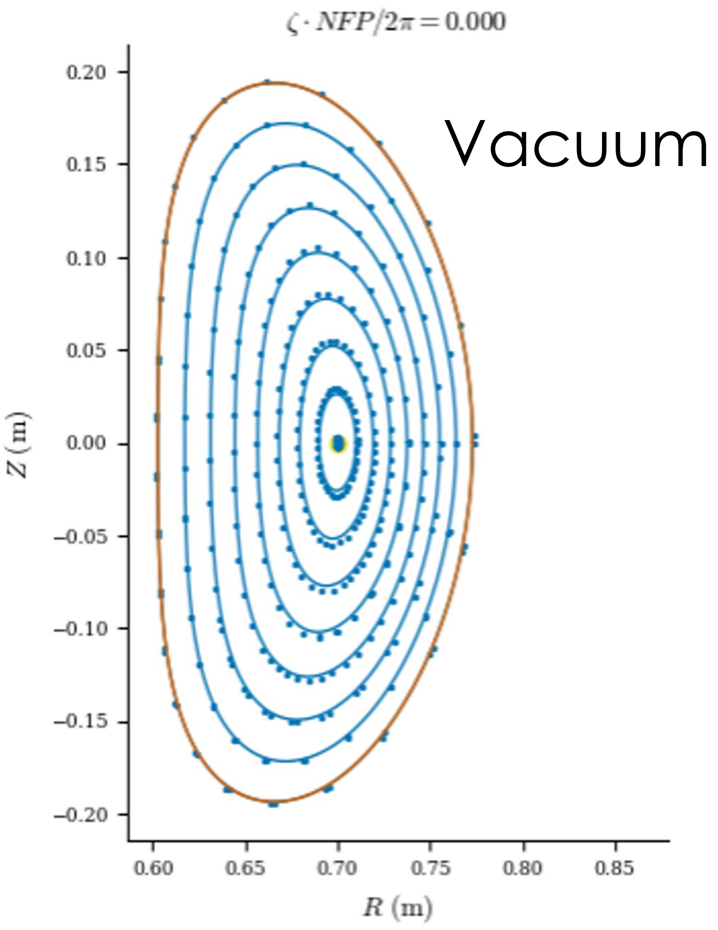
pyQSC equilibrium evaluated at r =0.1

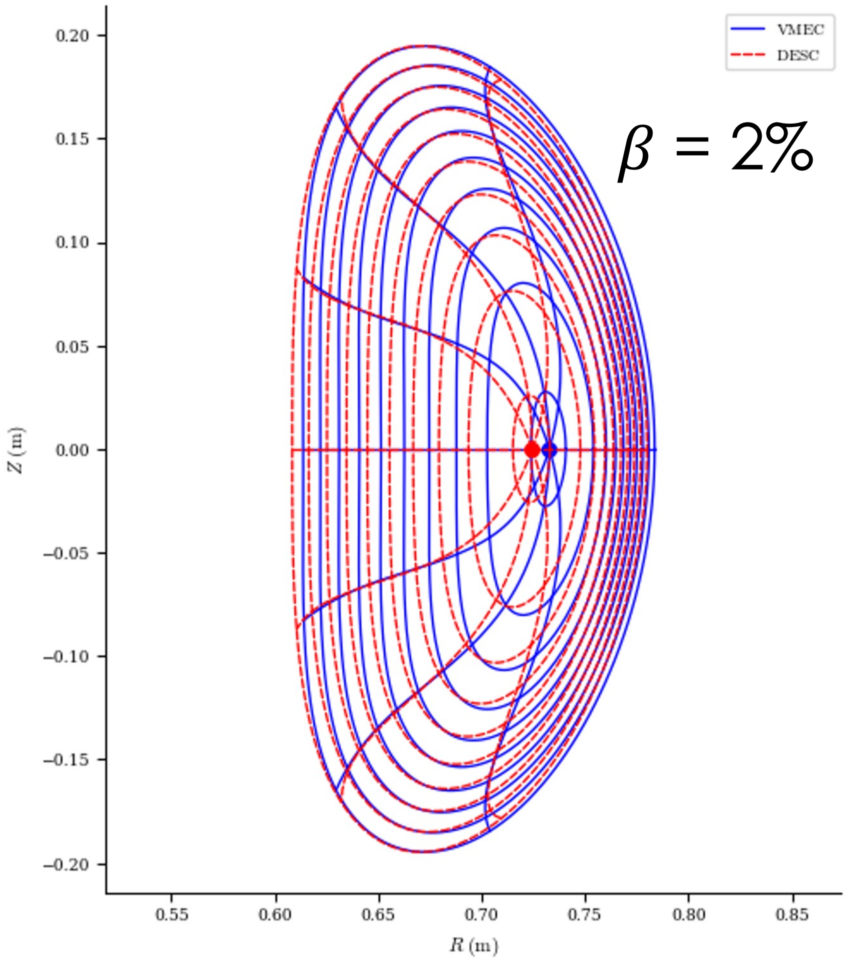
Near-Axis-Expansion Constrained Equilibria in DESC



- **Global equilibria solutions with near-axis behavior constrained to match the NAE to O(ρ)**
- **Enables the connection between global MHD equilibria solutions and the existing insight on optimized stellarators**

Free boundary DESC





- **Agrees with field line tracing for vacuum cases.**
- **Disagrees with VMEC at finite pressure/current**
- **Using re-implementation of NESTOR, benchmarked against original**
- **Also re-implemented high order method from Malhotra (2019)**
  - **Not getting expected level of convergence**
- **Exploring other methods to avoid singular integrals entirely**

Gradient computations are the bottleneck of traditional stellarator optimization

- $g(\boldsymbol{c})$ = cost function to be minimized; $\boldsymbol{c}$ = optimization variables
- Gradient descent optimization:

$$\boldsymbol{c}_{n+1} = \boldsymbol{c}_n - \gamma \nabla g(\boldsymbol{c}_n)$$

Finite Differences:

- Requires $\geq \dim(\boldsymbol{c})$ equilibrium solves
- Inaccurate and sensitive to step size

Adjoint methods:

- Not applicable to all objectives
- Laborious to implement

Efficient computing with the ease of Python

Automatic Differentiation (AD)

- Optimization requires derivative information
- Exact derivatives of arbitrary functions to any order

Just-In-Time (JIT) Compilation

- Comparable speed to C or Fortran compiled languages
- Hardware agnostic (CPU, GPU, TPU)

Requires specific code structure, but easy to implement: **`import jax.numpy as jnp`**

DESC optimization only requires a single **equilibrium solve per iteration**

1. Newton optimization step with equilibrium constraint

$$\boldsymbol{c}_{n+1} = \boldsymbol{c}_n + \Delta\boldsymbol{c}$$

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

Exact Jacobians known from automatic differentiation!

2. Perturb equilibrium solution to reflect new parameters

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta\boldsymbol{x}$$

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

$\boldsymbol{f}$ = equilibrium constraint
$\boldsymbol{g}$ = optimization objective

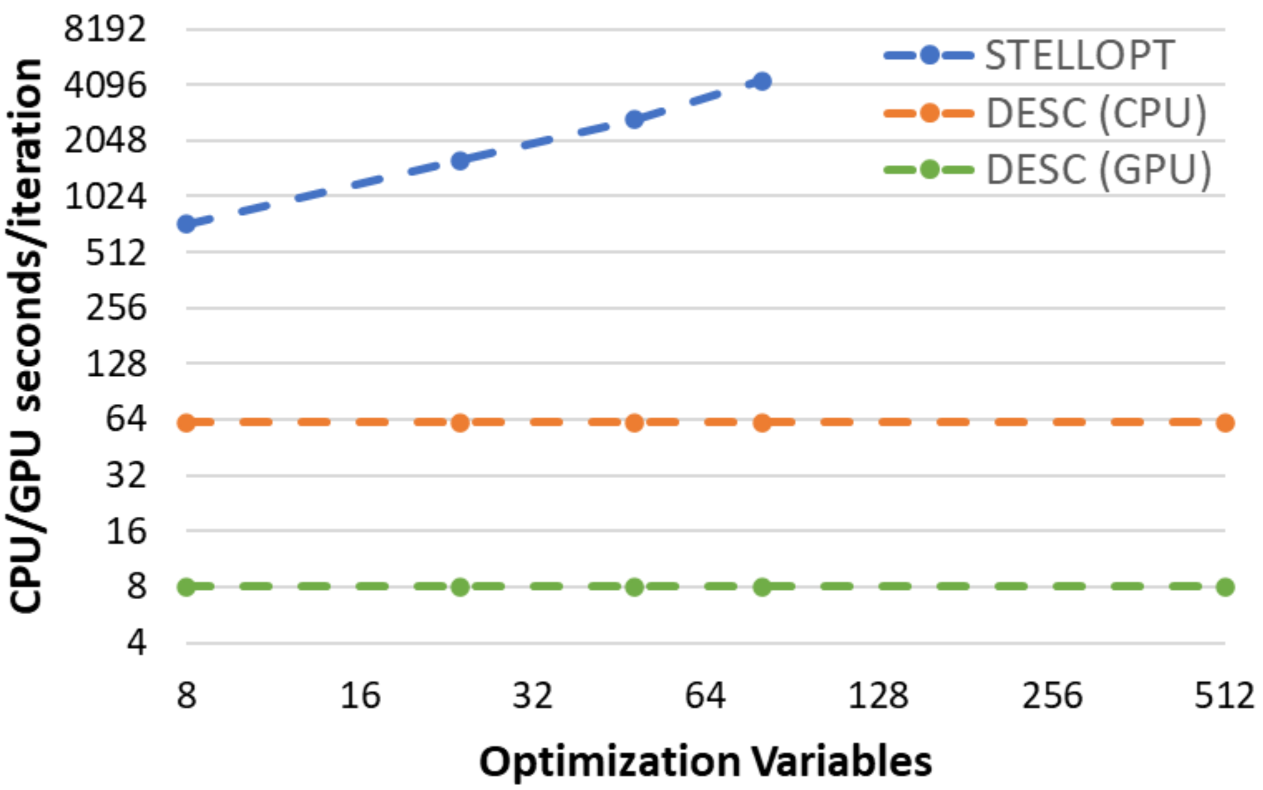
$\boldsymbol{x}$ = equilibrium solution
$\boldsymbol{c}$ = optimization variables

3. Re-solve equilibrium from this close initial guess

$$\boldsymbol{x}_{n+1} = \operatorname{argmin}_{\boldsymbol{x}}(\|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}_{n+1})\|^2)$$

Only 1 “warm-start” equilibrium solve per optimization step!

Fast computations enable exploration of the large stellarator design space



- Finite differences scale unfavorably
- Parallelization can help reduce wall time, but not total resources
- GPU hardware is still improving

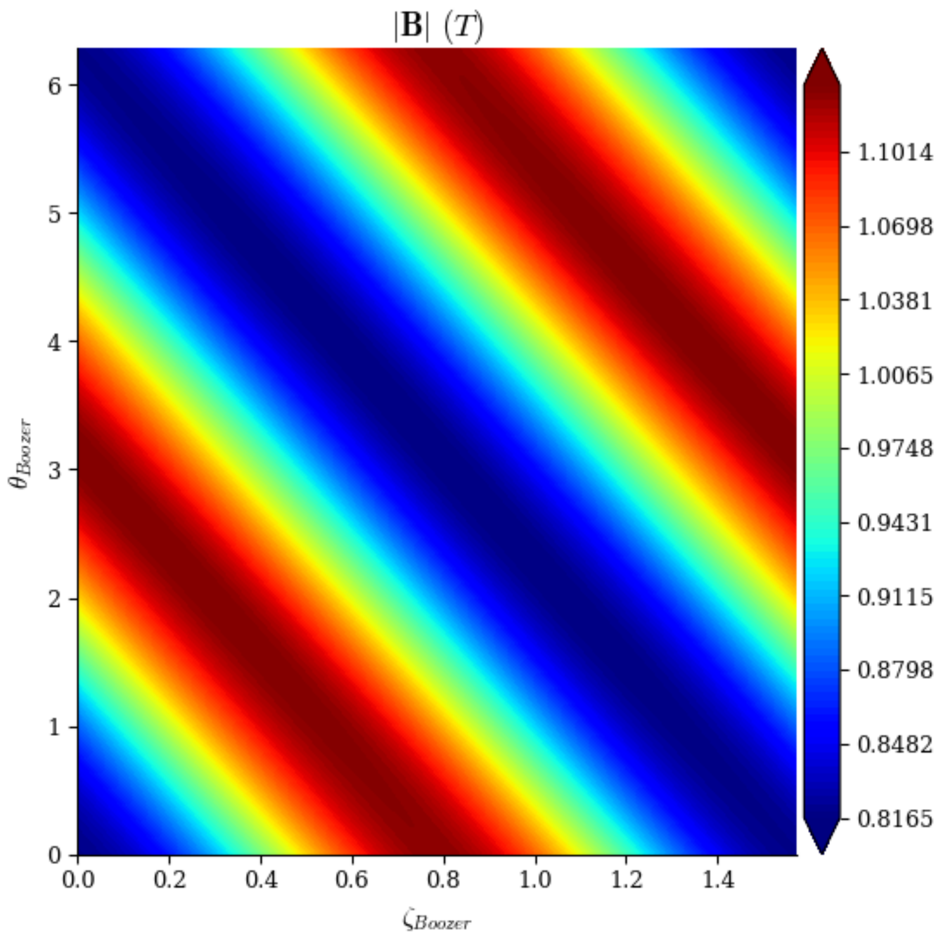
W7-X $\beta = 2\%; L = 24, M = N = 12$

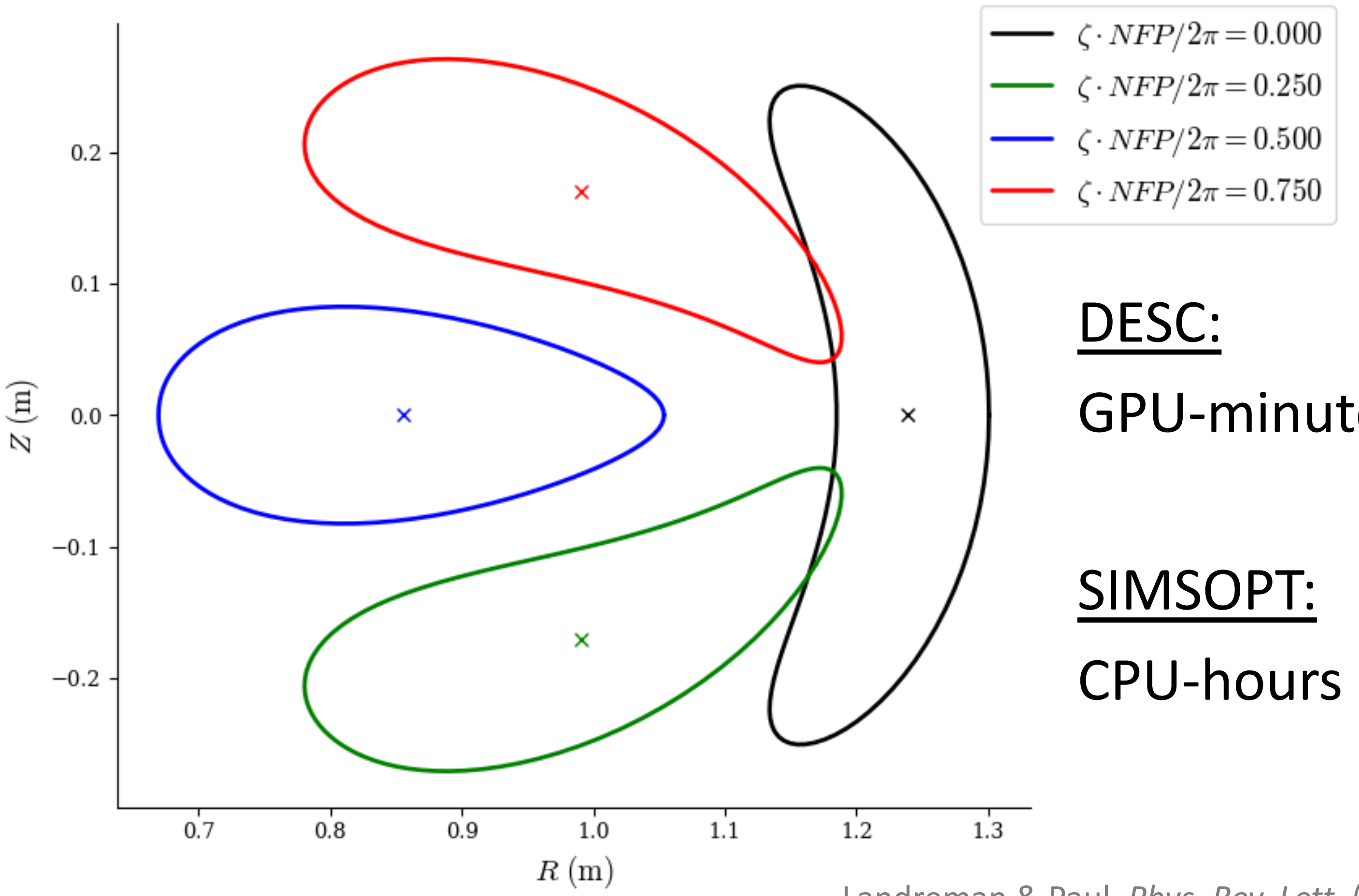
Hardware	Run Time
Intel Cascade Lake CPU	48 min
NVIDIA A100 GPU	20 min

Run optimizations in a few lines of Python code

```
set_device("gpu")  # run on a GPU
eq = desc.io.load("path/to/initial/equilibrium.h5")
grid = LinearGrid(M=eq.M, N=eq.N, NFP=eq.NFP, rho=np.linspace(0.1, 1, 10))  # computation grid
objective = ObjectiveFunction((AspectRatio(target=8),  # target aspect ratio
    QuasisymmetryTwoTerm(helicity=(1, -eq.NFP), grid=grid, weight=2e-1)))  # optimize for QH
# optimize boundary modes with |m|,|n|<=5 (constrain boundary modes with |m|,|n|>5)
R_modes = np.vstack(([0, 0, 0],  # fix major radius
    eq.surface.R_basis.modes[np.max(np.abs(eq.surface.R_basis.modes), 1) > 5, :]))
Z_modes = eq.surface.Z_basis.modes[np.max(np.abs(eq.surface.Z_basis.modes), 1) > 5, :]
constraints = (ForceBalance(), FixBoundaryR(modes=R_modes), FixBoundaryZ(modes=Z_modes),
    FixPressure(), FixCurrent(), FixPsi())  # fix vacuum profiles
optimizer = Optimizer("lsq-exact")  # least-squares optimization algorithm
eq.optimize(objective, constraints, optimizer)  # run optimization
eq.save("path/to/optimal/solution.h5")
```

Can find “precise quasi-symmetry” & more





<u>DESC:</u>

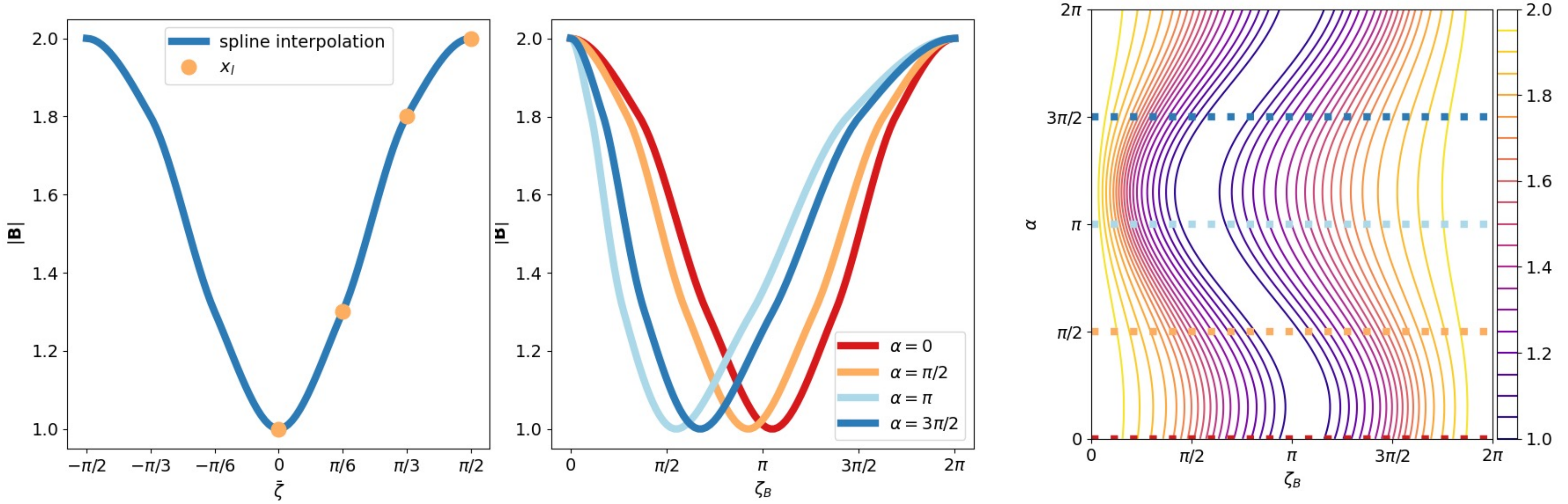
GPU-minutes

<u>SIMSOPT:</u>

CPU-hours

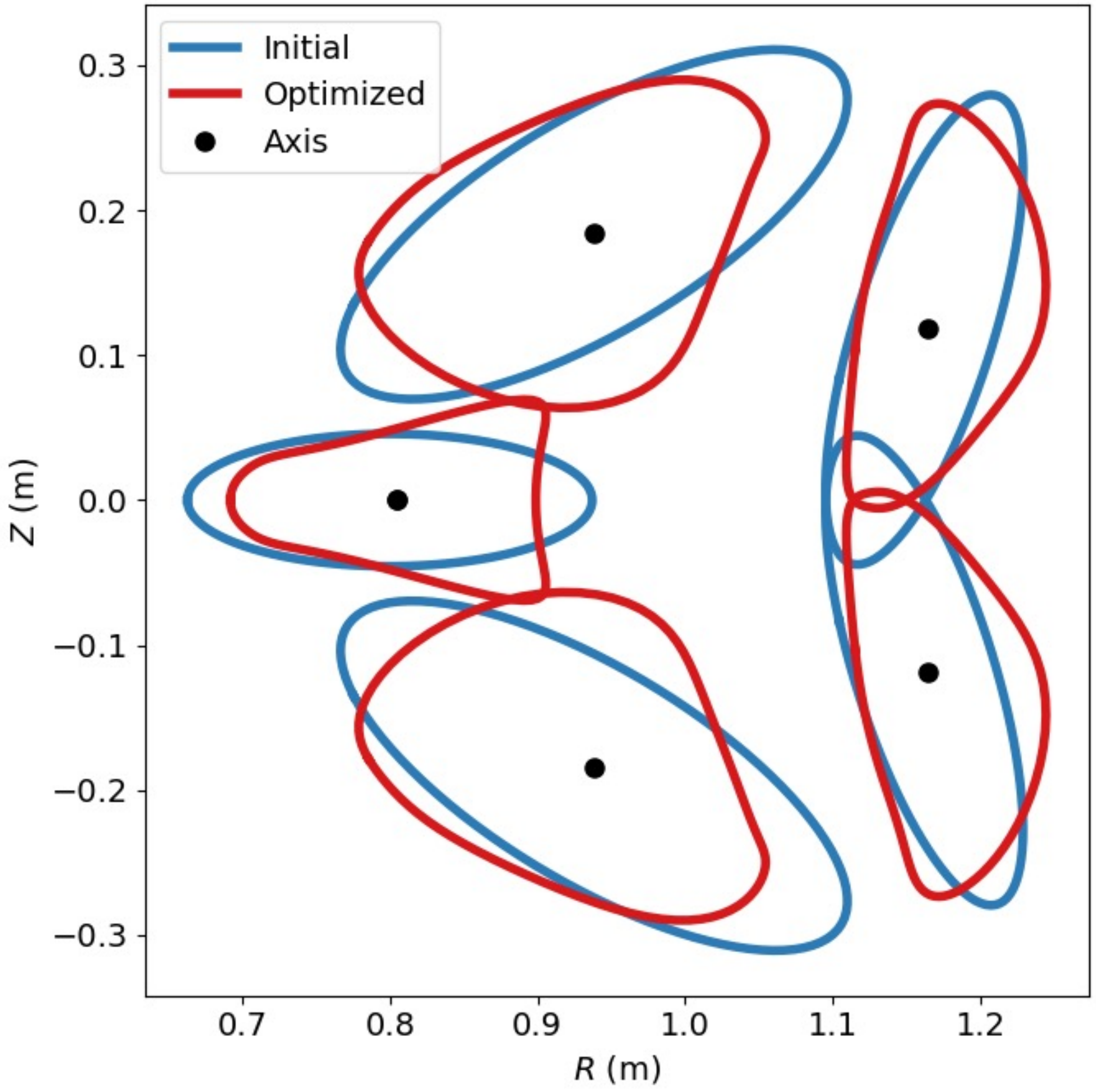
Landreman & Paul, *Phys. Rev. Lett.* (2022)

Full QI Phase Space is defined in DESC



- Specify the magnetic well “shape” with a monotonic spline
- Specify how the well “shifts” on different field lines with a Fourier series
- Generate arbitrary QI magnetic field targets without prior initialization
- **Parameterization enables scans of the QI optimization landscape**

Can Do QI Optimization (with NAE)



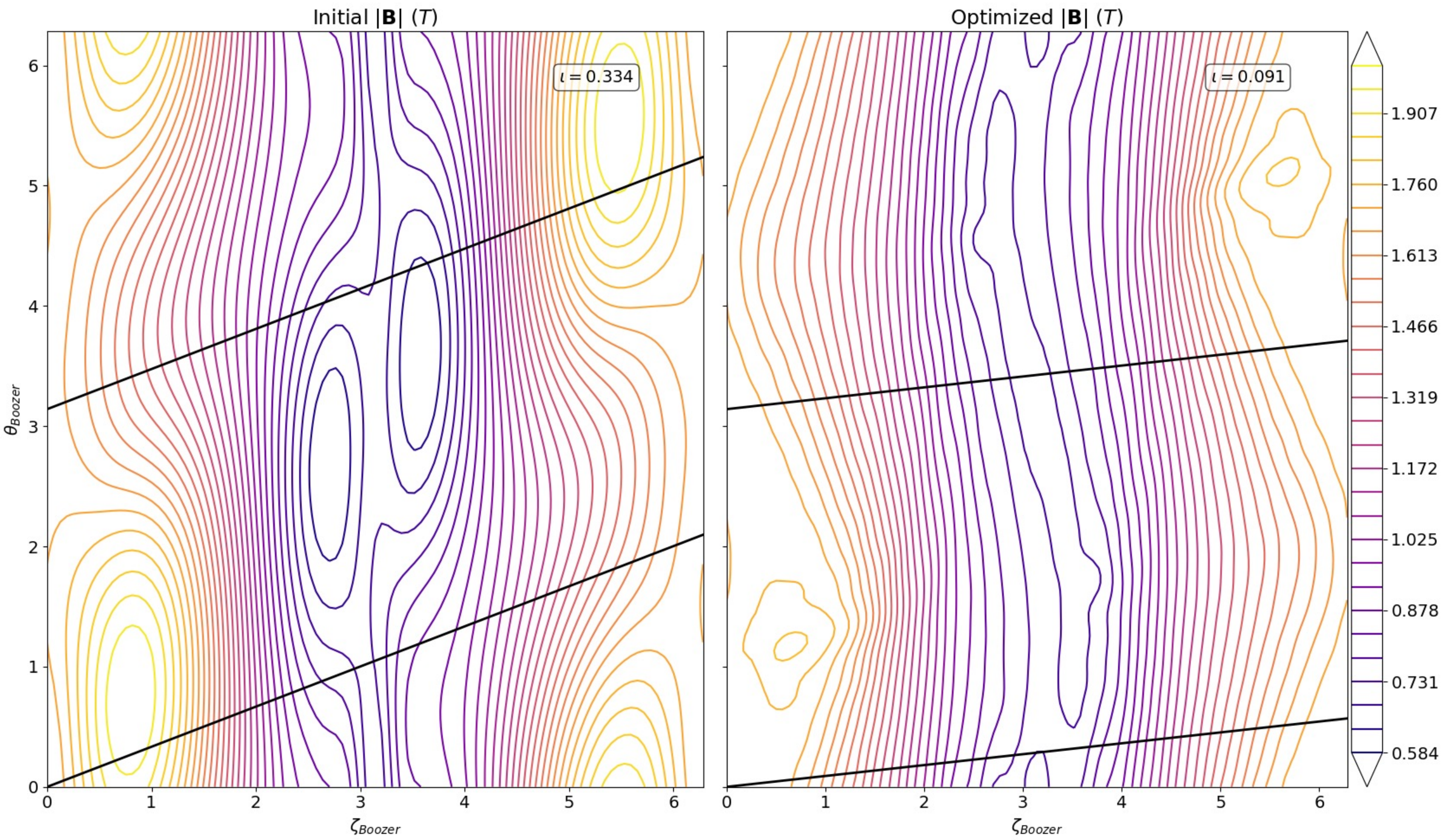
<u>Initial equilibrium:</u>

- Analytic near-axis model
- $O(\rho)$ near-axis behavior constrained

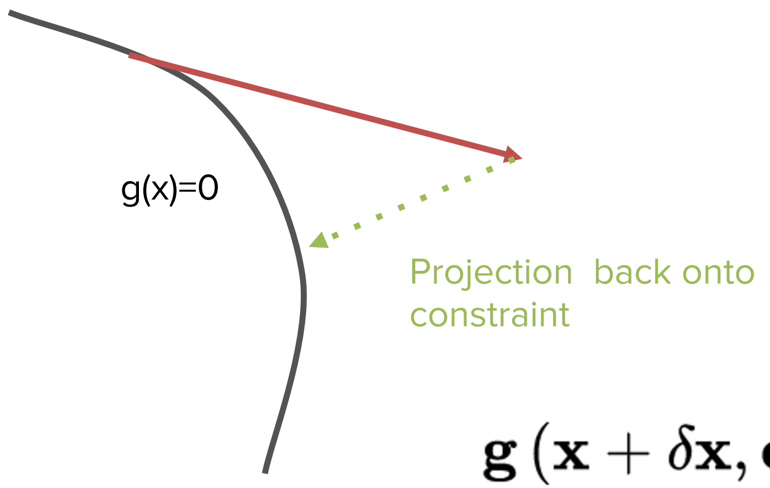
<u>Optimization targets:</u>

- Unconstrained QI on multiple surfaces
- Vacuum force balance: $J^{\rho} = J^{\theta} = J^{\zeta} = 0$

Can Do QI Optimization



Traditional “Loopy” Optimization



$$
\begin{aligned}
\mathbf{g}\left(\mathbf{x}+\delta\mathbf{x},\mathbf{c}+\delta\mathbf{c}\right) &= 0\\
\mathbf{f}\left(\mathbf{x}+\delta\mathbf{x},\mathbf{c}+\delta\mathbf{c}\right) &= 0
\end{aligned}
$$

```
while optimization stopping criteria are not met
    :

    perturb equilibrium solution to improve
    objective

    while equilibrium stopping criteria are not
    met:

        solve equilibrium force balance
```

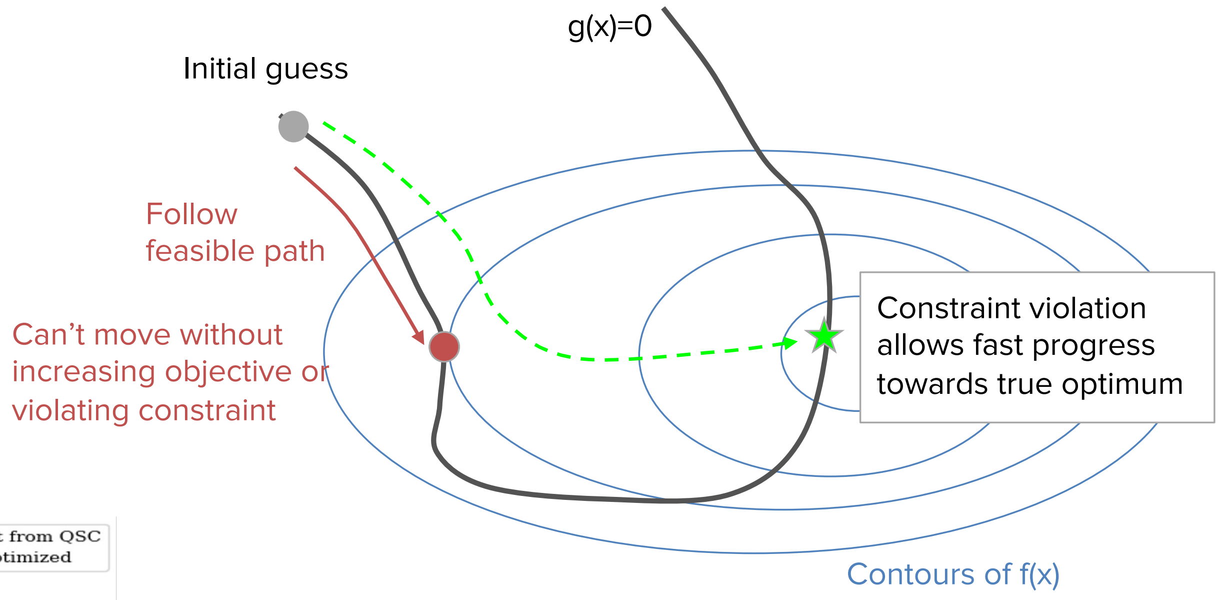
For Equilibrium constraints, standard approach is a “projection” method

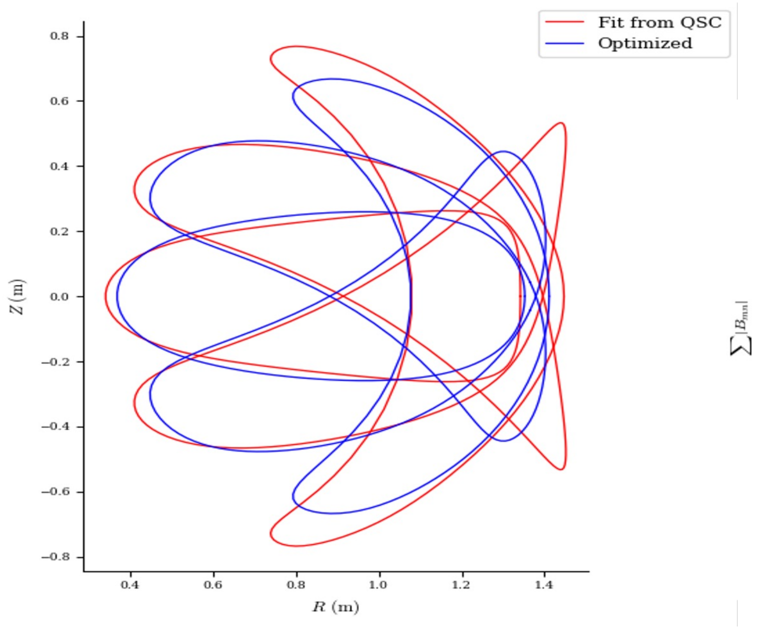
- When trying a new step, resolve equilibrium subproblem before evaluating cost
- Expensive (1+ equilibrium solve at each step)
- Projection can undo progress from optimizer

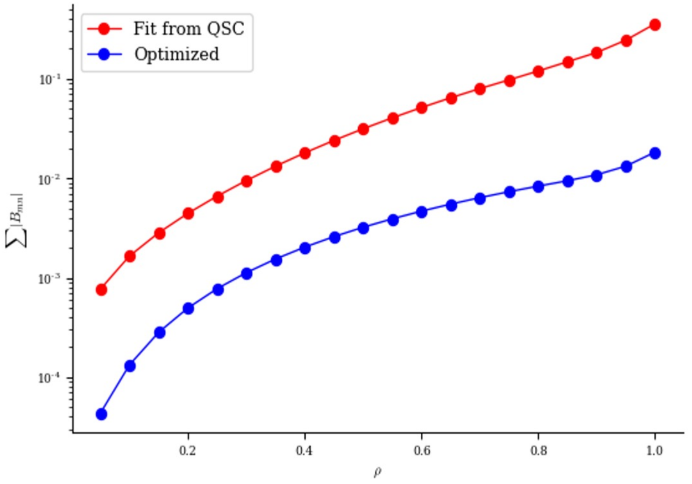
DESC Allow Combined Constraints + Optimization

$$\min_{x} f(x)$$

$$\text{subject to} \quad \begin{aligned} g_{eq}(x) &= 0 \\ g_{ineq}(x) &\geq 0 \end{aligned}$$







Example: Fix NEA + eq. constraint + optimize remaining volume

Current methods : Sum of Squares

Combine equality + inequality constraints

$$\min_x f(x) + w_1[g(x)]^2$$

Choose small weight for inequality constraints to enforce “approximately”

Choose large weight for equality constraints to penalize a lot

Limitations:

- Hard to guess a-priori what weights should be
- Even small weights for “inequality” constraints can overly penalize things we don’t care about

Better methods: Augmented Lagrangian

- Combination of traditional Lagrangian + quadratic penalty

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^T \mathbf{g}(x) + \mu g^2(x)$$

- Doesn't introduce any non-smooth terms
- "Exact" method - doesn't need $\mu \rightarrow$ infinity
- Solve sequence of subproblems for increasing μ, λ
- Provides estimate of true Lagrange multipliers - useful information about trade-offs

- Open source packages available (LANCELOT, NLopt, etc). Also python/JAX version implemented in DESC

Better methods : Interior Point

$$\min_{x,s} f(x) - \mu \sum_i \log(s_i)$$

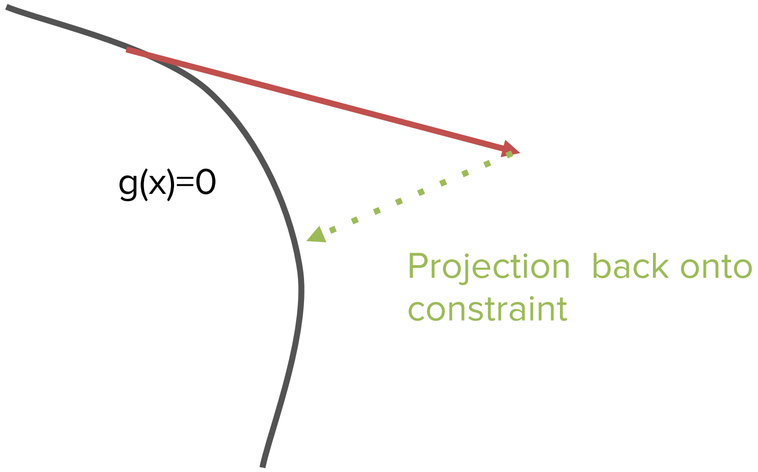
$$\text{subject to} \quad \begin{aligned} g_{eq}(x) &= 0 \\ g_{ineq}(x) - s &= 0 \end{aligned}$$

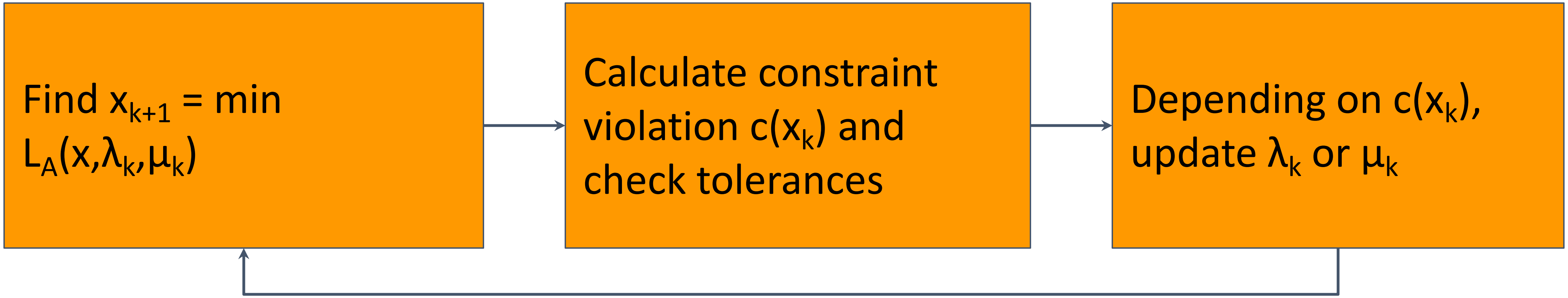
- Introduce log barrier to deal with inequality constraints
- Solve sequence of subproblems for $\mu \to 0$

- High quality open source options (`ipopt`, `scipy`) interfaced with DESC

DESC Allow Combined Constraints + Optimization

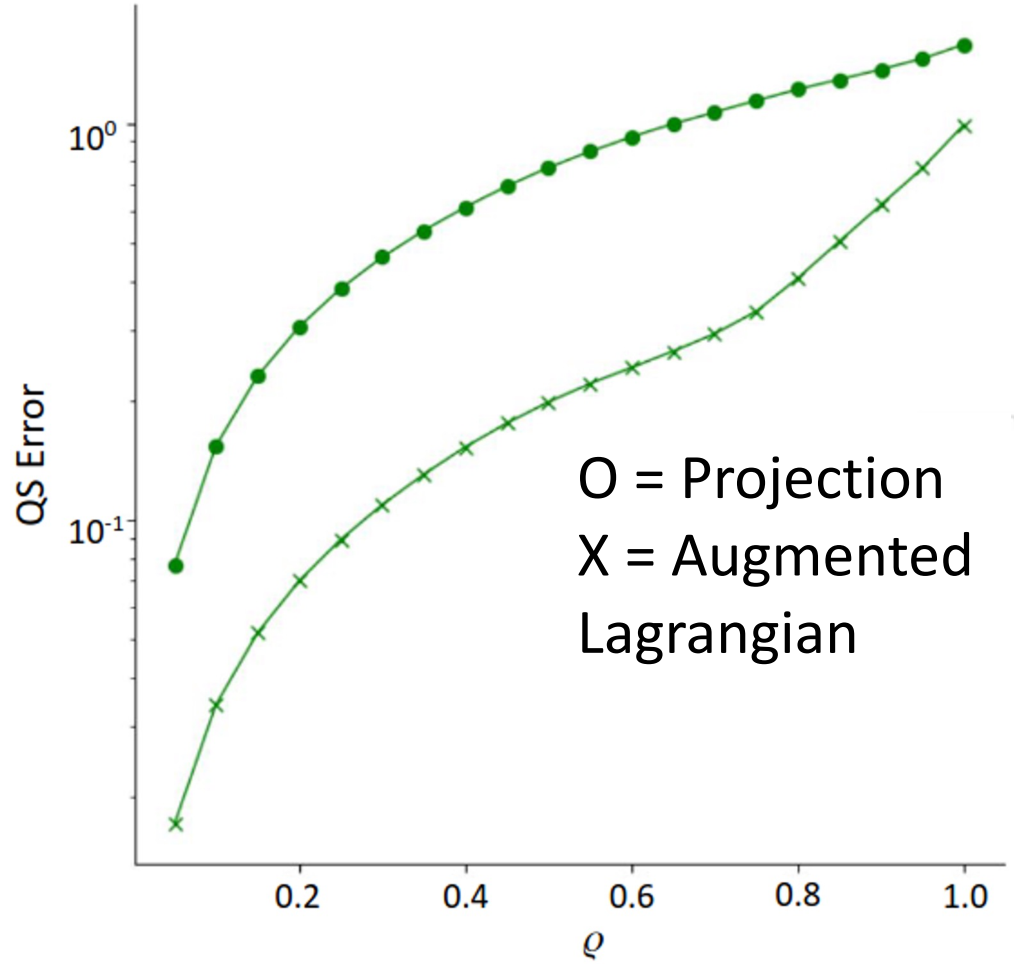
```
while optimization stopping criteria are not met
    :

    perturb equilibrium solution to improve
    objective

    while equilibrium stopping criteria are not
    met:

        solve equilibrium force balance
```



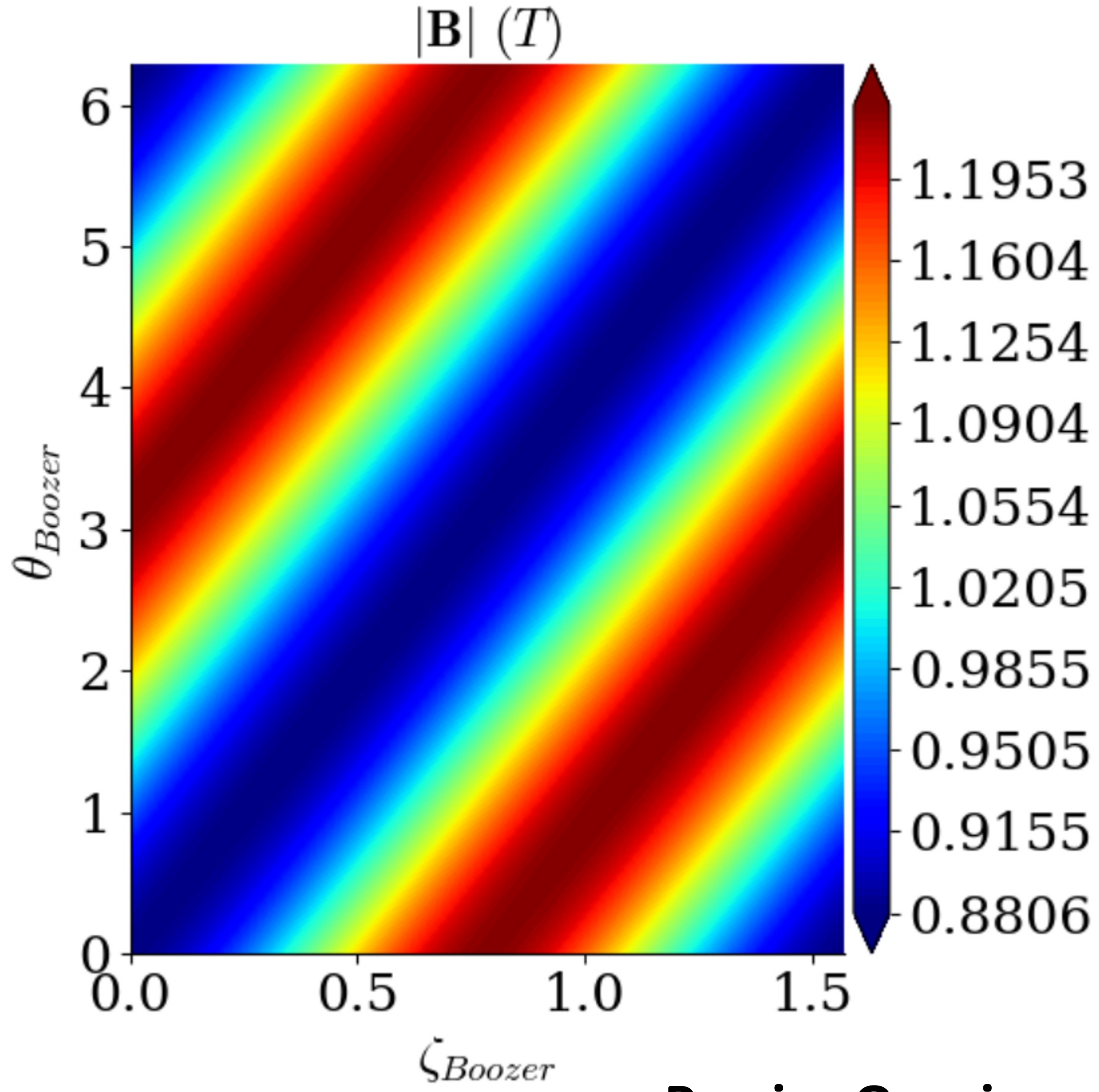


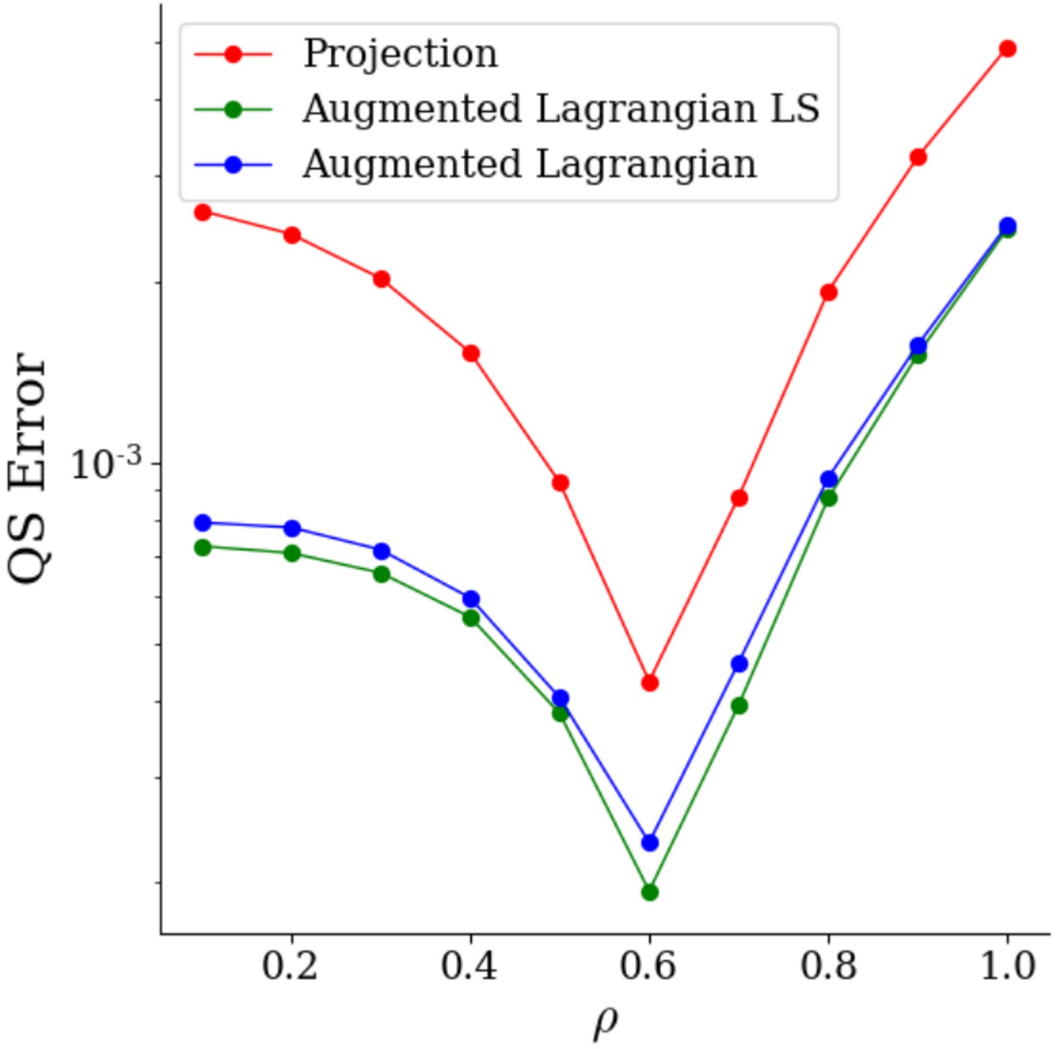
Relaxing constraints during optimization allows for better results

- Projection method resolves from boundary at each step, enforcing force balance
- Causes solution to get stuck in local minima
- Augmented Lagrangian allows solution to temporarily violate equilibrium to improve QS
- Allows it to skip over local minima and achieve better final result



Combined Constraints + Optimization gives better results

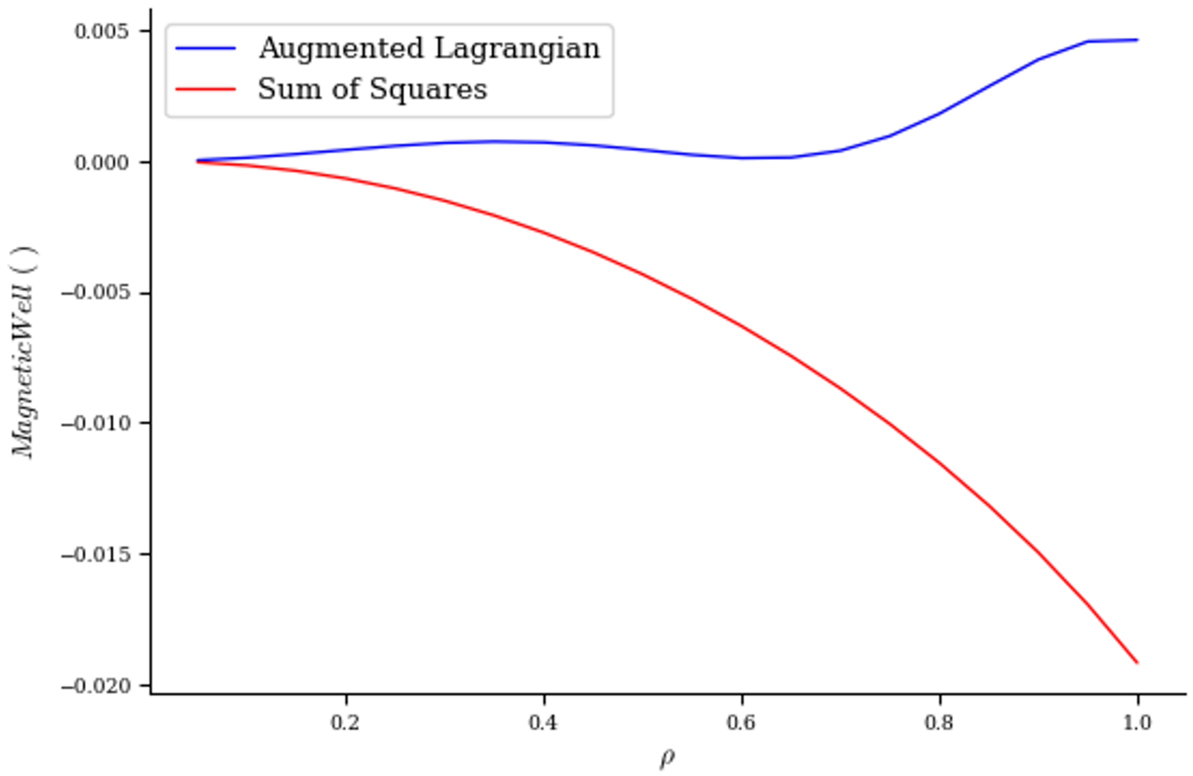




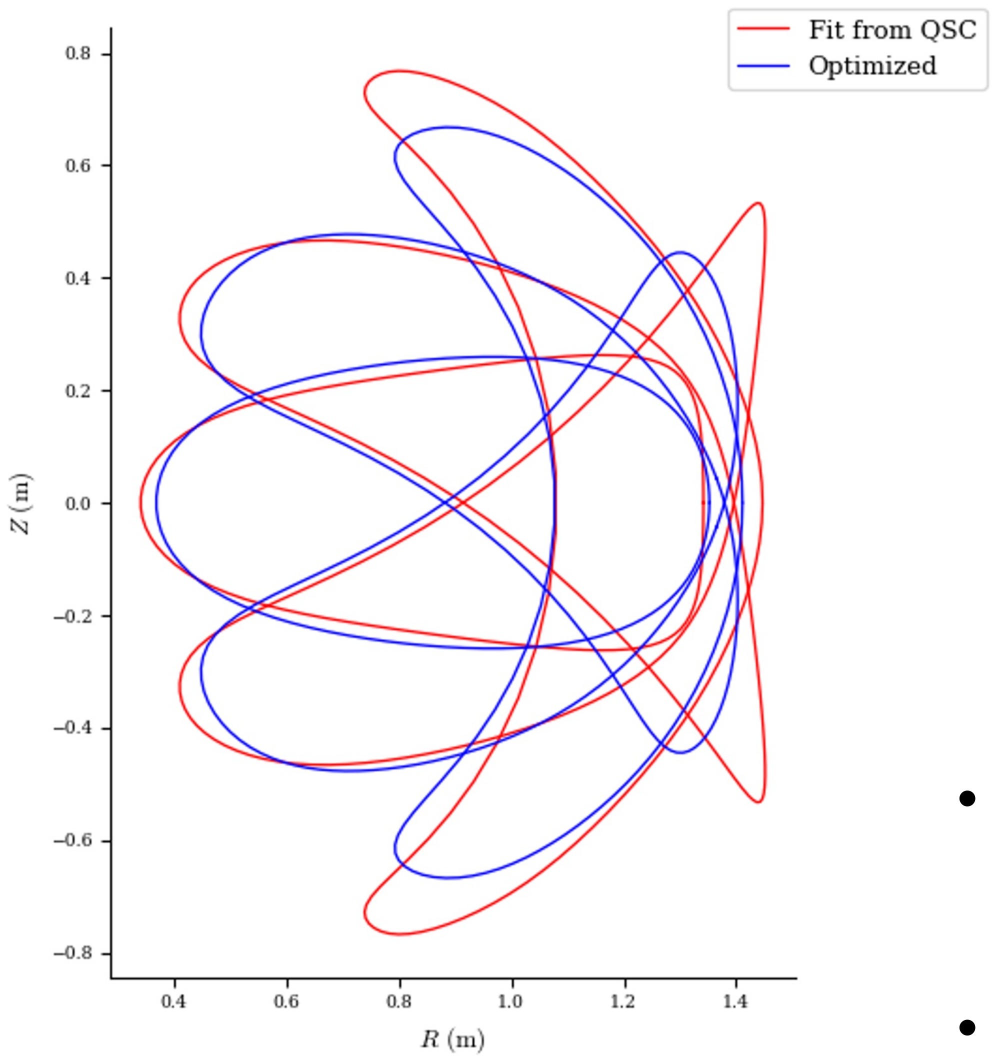
Precise Quasisymmetry Example

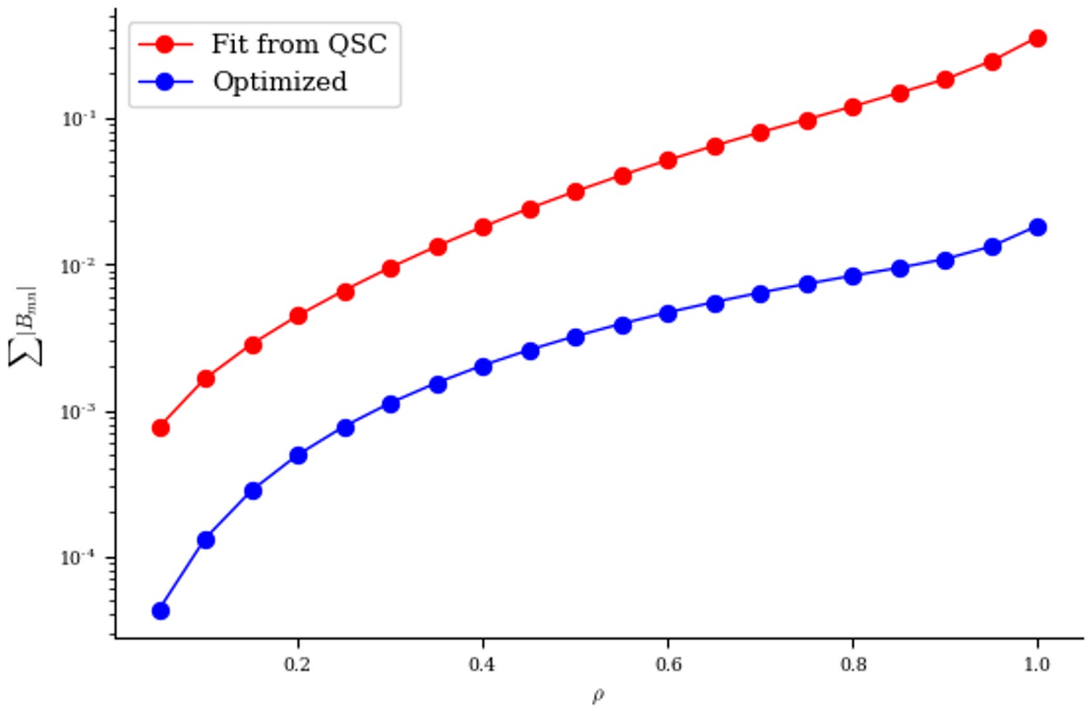
Augmented Lagrangian takes guesswork out of penalty terms

- Simple quadratic penalty fails to give stable equilibrium, even for large values of weight
- Instead applying inequality constraint w/ augmented Langrangian gives magnetic well > 0



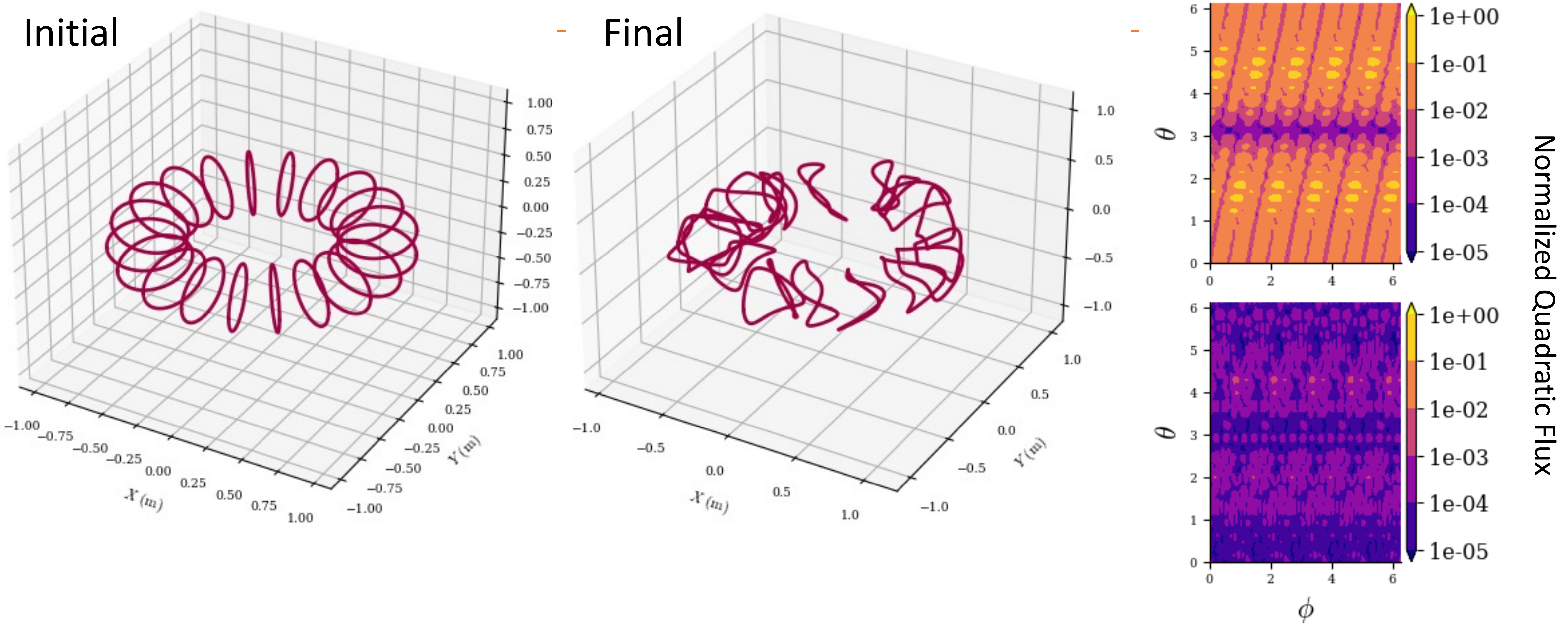
Optimizing with fixed near axis behavior





- Constrained optimizers allow more general constraints than standard approach of optimizing over boundary shape
- Example: Fix near axis behavior from QSC, optimize remaining volume

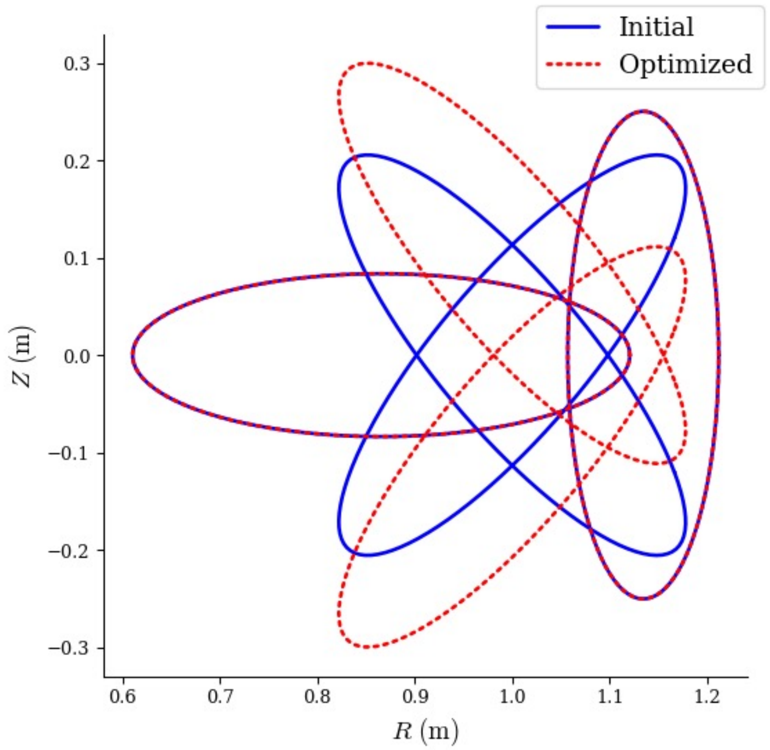
Can perform coil design & optimization

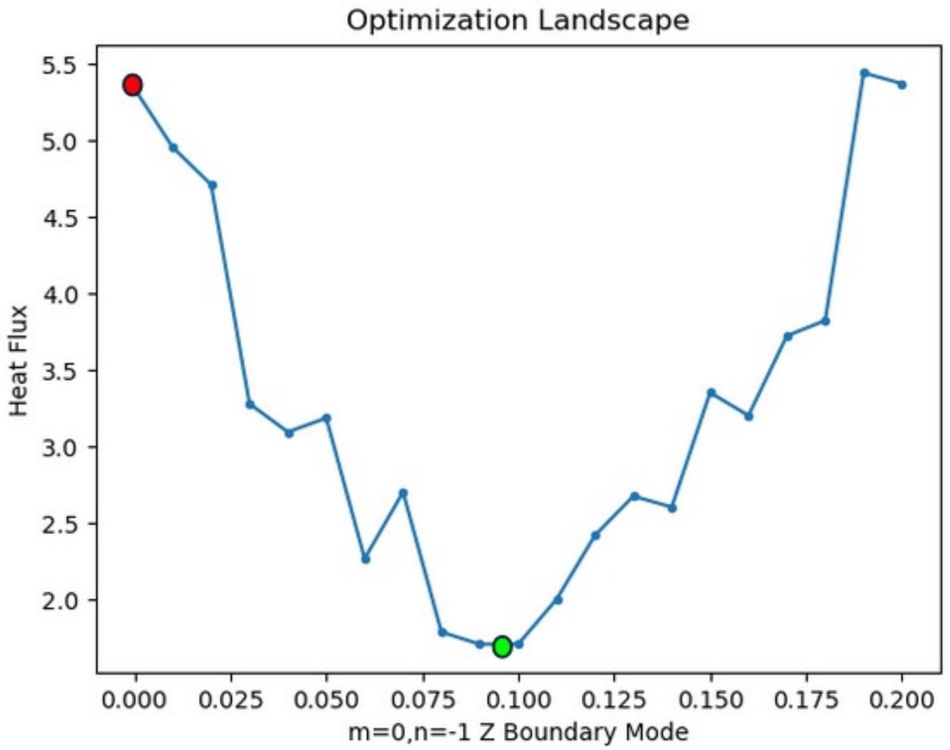


- Fixing length of each coil
- Enforcing minimum coil-coil and coil-plasma distance
- Optimized using SLSQP algorithm from scipy

Can wrap other codes with finite differences

- GX is a fast (minutes) pseudo-spectral gyrokinetic code for stellarators



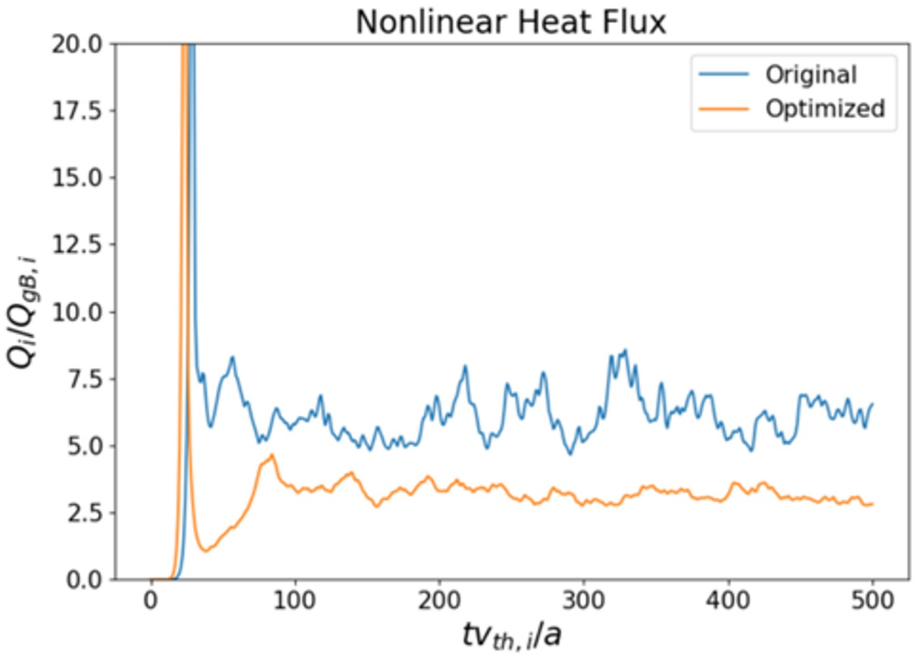


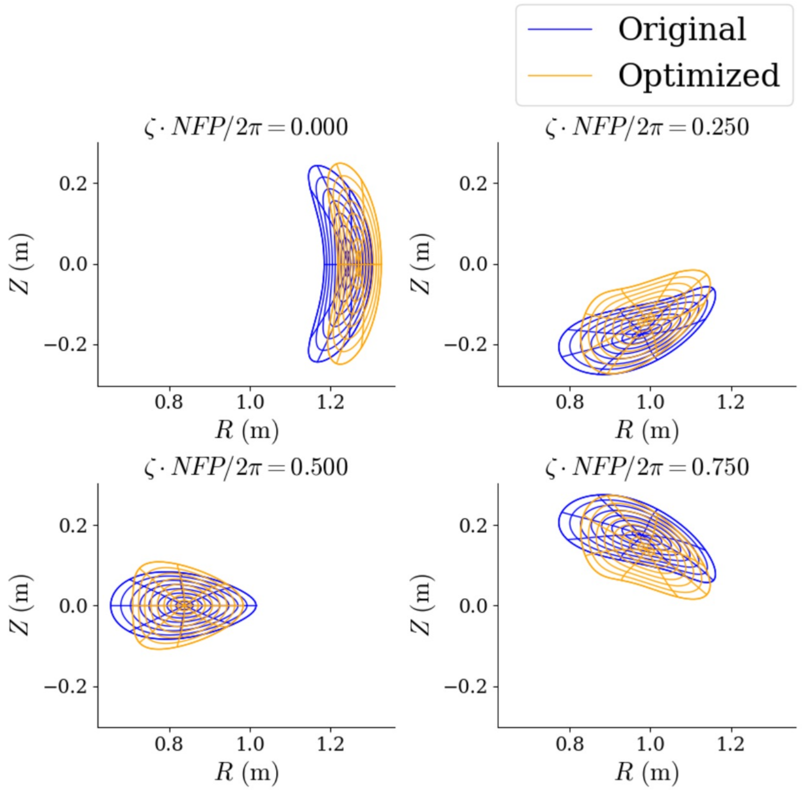
- Also wrapped NEO to optimize for effective ripple ε_{eff}

Mandell et al., *J. Plasma Phys.* (2018)
Gonzalez et al., *J. Plasma Phys.* (2022)
Nemov et al., *Phys. Plasmas* (1999)

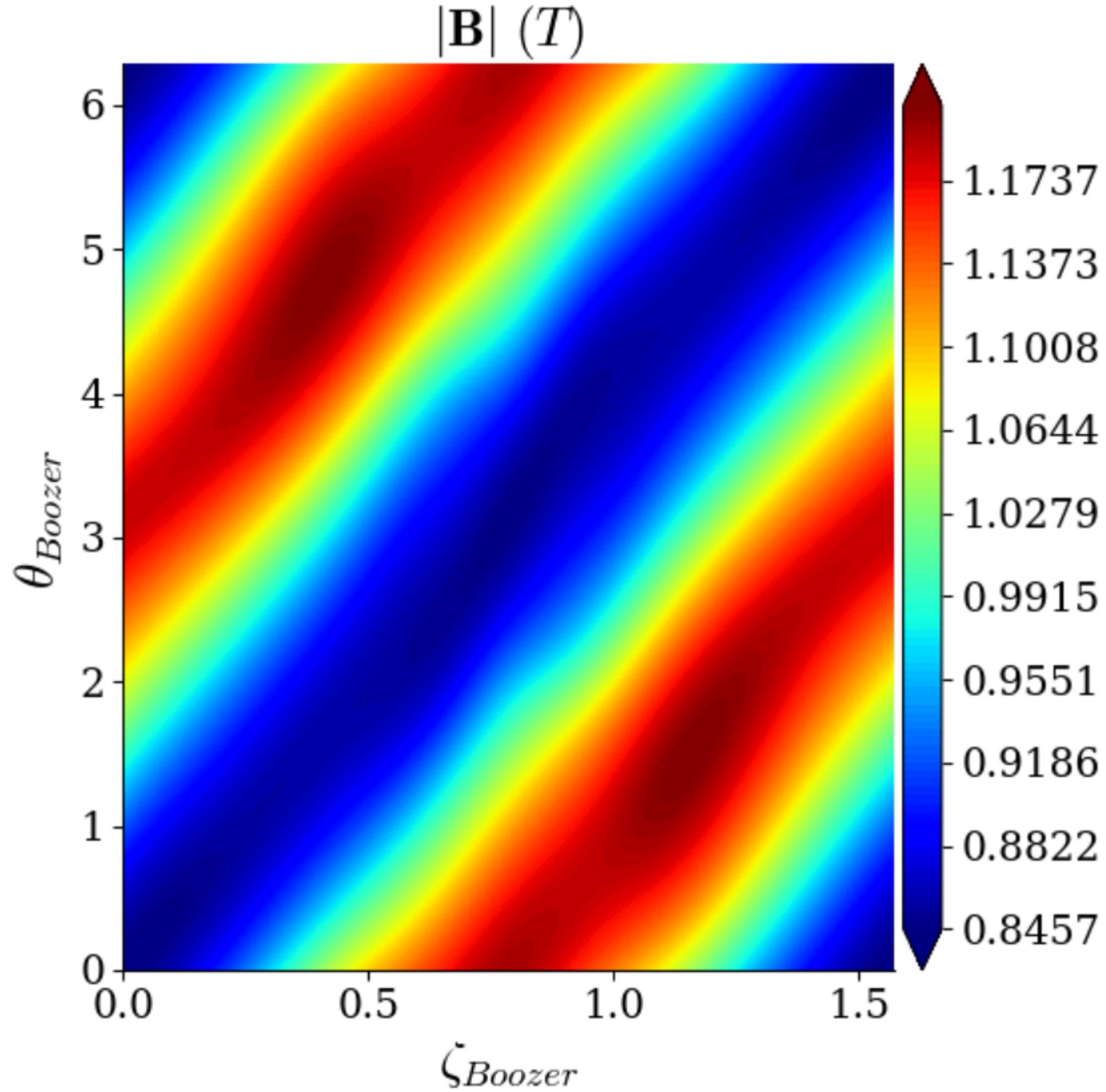
Turbulence + QS Optimization

- Initial equilibrium is a low-resolution version of a precise QH equilibrium.
- Optimizer reduces nonlinear heat flux by about half, while maintaining good quasisymmetry.

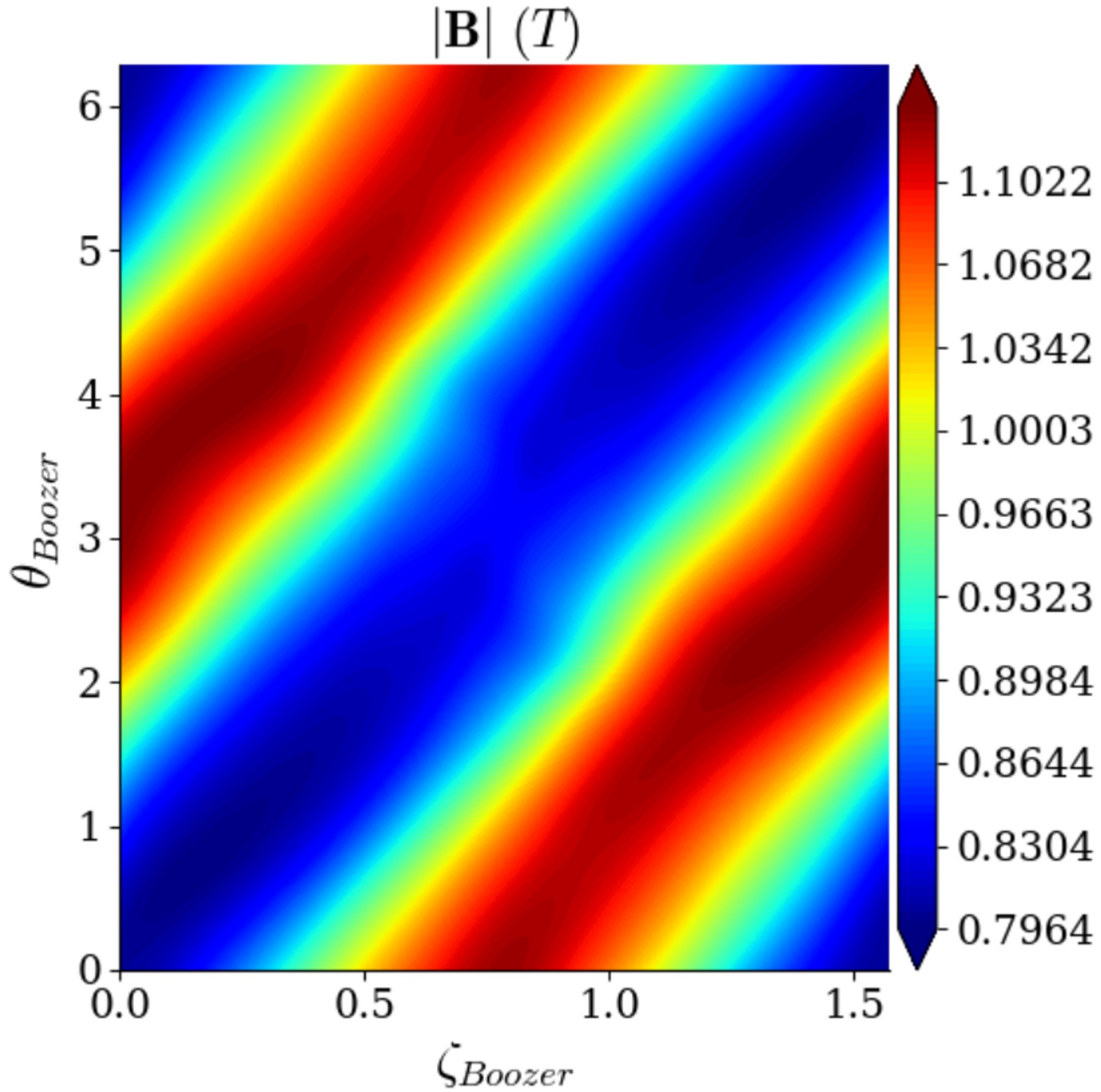




Turbulence + QS Optimization



Original

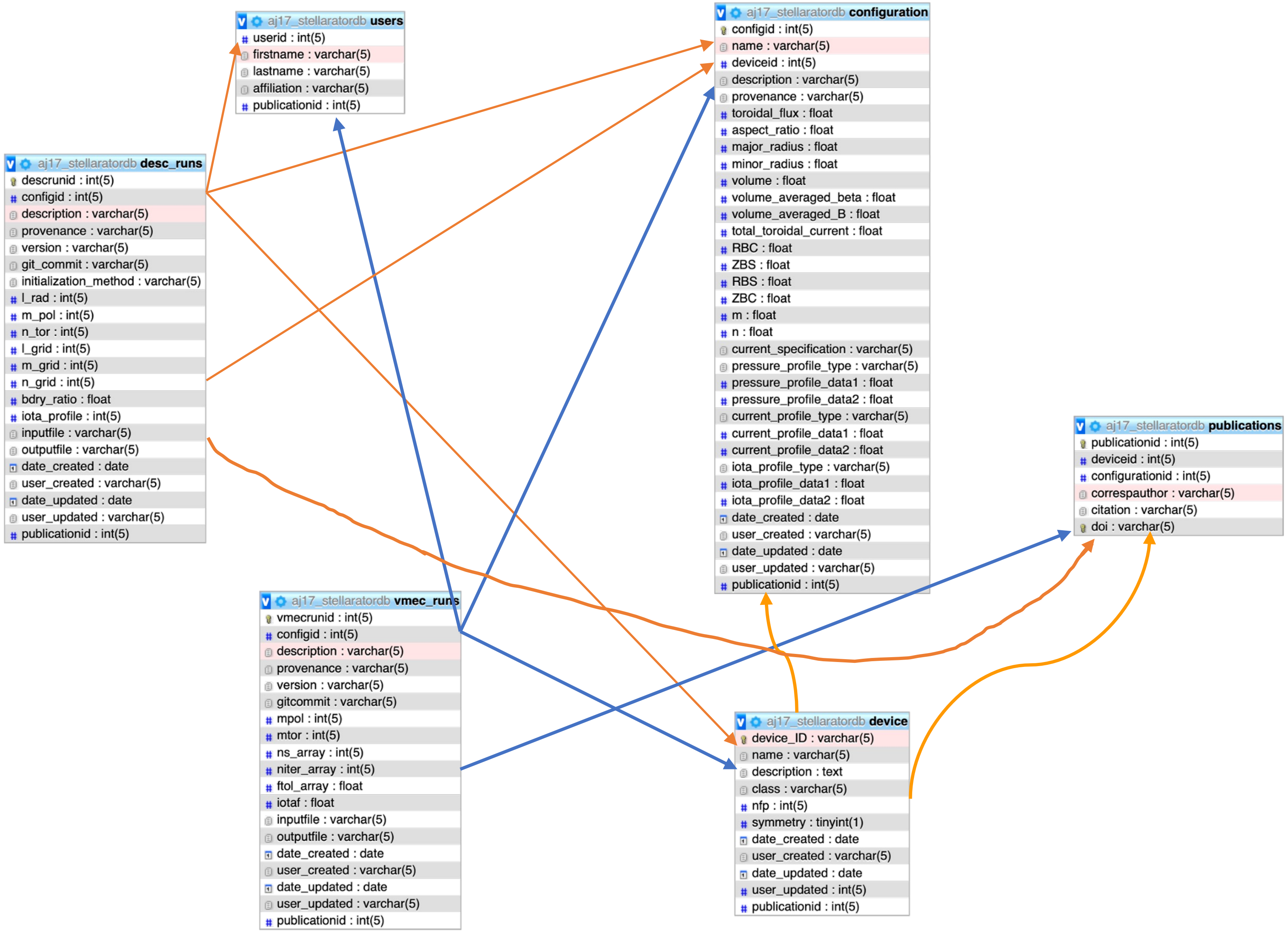


Optimized

Machine Learning for Stellarators

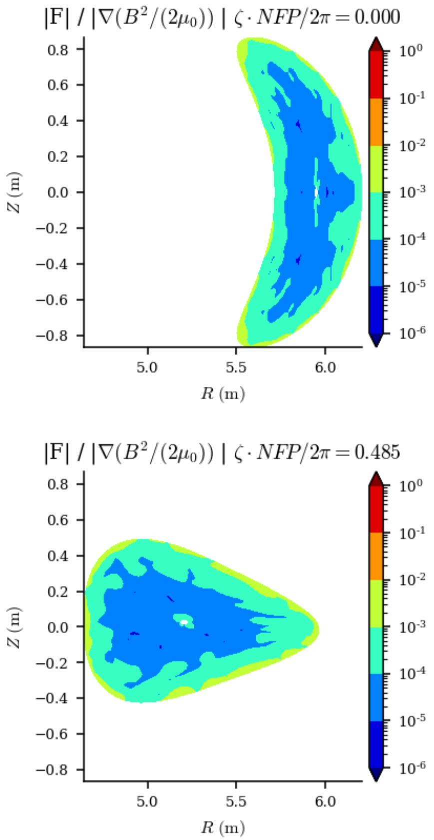
Developing a database structure and storage system for Simons Collaborators (Aza Jalalvand)

Machine Learning for Stellarator Equilibrium and Optimization

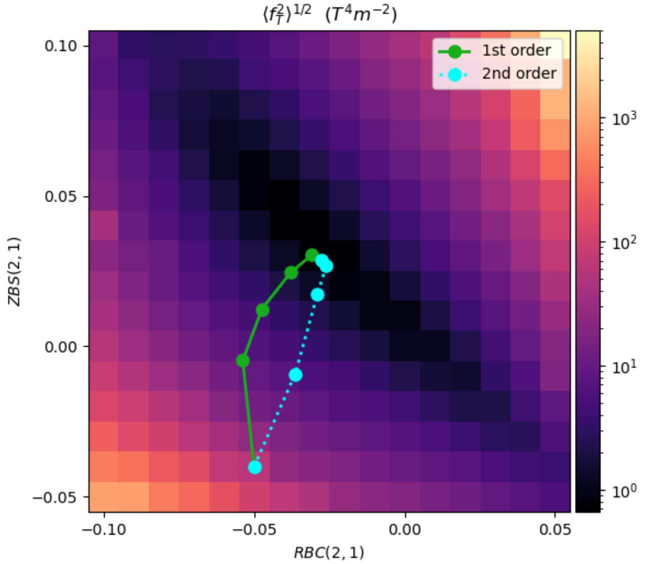


DESC is a new tool for stellarator optimization

Accurate Equilibria



Fast Optimization



Flexible

Better tools = better stellarator reactors!

Final Take: Fix the core, do proper constrained optimization

1. Don't specify R, Z surface Fourier! It is 2x the needed # param. on surface (x5 Poincare)
    - Why specify looping/intersecting, over constrained parameters we have no intuition for? And %100 will give non-nested solutions?
2. Specify core with NEA (maybe +-%10 inequality constraint): underconstrained
    - Extra: if you want QI specify the phase space parameterization.
3. Stop the loopy optimization (perturb > project)!
    - Use Augmented Lagrangian or Interior Point methods
    - Force balance will be satisfied not with a loop within a loop but by the optimizer
4. Problem is way simpler! Physicists just need to write their cost function for high level physics (turbulence, radiation,…)

Ideas/Collaborations

- Prove Poincare section input gives unique equilibrium
- What is the minimum parameter set that define ***nested flux*** phase space?
  - Search within this phase space
- Novel ideas (BEI free) for solving Free Surface Equilibrium

- Codes based on particle integration: We can do fast GPU integration and autodiff for lightning end-end optimization. Rogerio is onboard! Anyone else?

- Take your code to optimization school day: Let's get f(x) g(x) out of the loop!

- New Stellarator SOL code development! Any suggestions?

Additional Resources

Princeton Plasma Control
control.princeton.edu

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

The Princeton Plasma Control group is recruiting graduate students and post-docs!

Contact Egemen Kolemen: ekolemen@pppl.gov