Stellarator Optimization with DESC

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

November 8, 2021

2021 APS DPP

PPPL

andlinger center for energy+the environment

PRINCETON UNIVERSITY

Executive Summary

1. Overview of equilibrium solver
2. Explanation of optimization method
3. Examples with quasi-symmetry optimization
4. Comparison of computation speed
5. Discussion of future work

- DESC is a flexible, accurate, and fast stellarator optimization code!

DESC Equilibrium Solver Overview

- $\boldsymbol{c}$ = input parameters specifying a unique equilibrium solution
  - last closed flux surface boundary shape
  - pressure and rotational transform profiles
- $\boldsymbol{x}$ = independent variables representing inverse equilibrium solution
  - $R(\rho,\theta,\zeta),\ Z(\rho,\theta,\zeta),\ \lambda(\rho,\theta,\zeta) = \vartheta - \theta$
- $\boldsymbol{f}$ = force balance residuals at discrete set of collocation nodes

$$\boldsymbol{F} \equiv \boldsymbol{J} \times \boldsymbol{B} - \boldsymbol{\nabla} p \quad \longrightarrow \quad f_\rho = \boldsymbol{F} \cdot \boldsymbol{\nabla}\rho \|\boldsymbol{\nabla}\rho\|,\ f_\beta = \boldsymbol{F} \cdot \boldsymbol{\beta} \|\boldsymbol{\beta}\|,\ \boldsymbol{\beta} \equiv B^\zeta \boldsymbol{\nabla}\theta - B^\theta \boldsymbol{\nabla}\zeta$$

- Equilibrium solution: $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{c}) = \begin{bmatrix} f_\rho \\ f_\beta \end{bmatrix} \approx \boldsymbol{0} \qquad \boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{f}(\boldsymbol{x},\boldsymbol{c})\|^2$

Gauss-Newton Trust-Region Method

- Non-linear least squares optimization

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{\Delta}$$

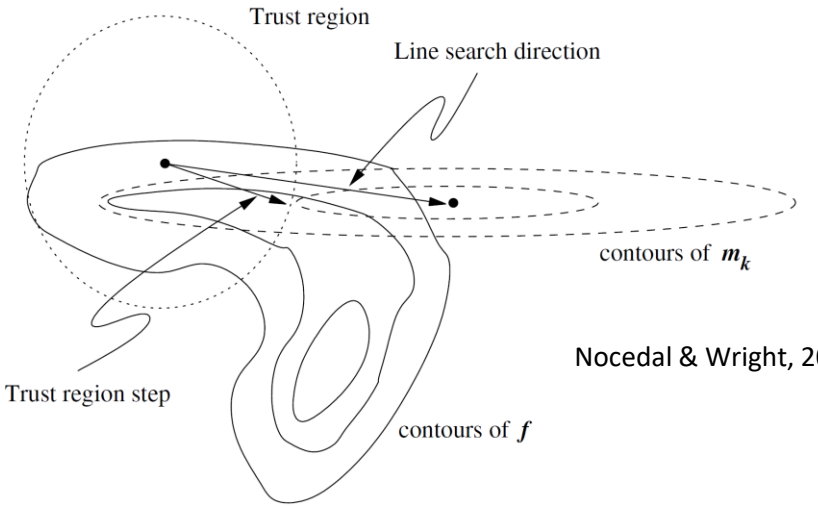
- Solves the linear system $\boldsymbol{J\Delta} = -\boldsymbol{r}$

$$\boldsymbol{\Delta}^* = \arg\min_{\boldsymbol{\Delta}} \|\boldsymbol{J\Delta} + \boldsymbol{r}\|, \ \|\boldsymbol{\Delta}\| \leq \delta$$

- Sub-problem: find Levenberg–Marquardt parameter α such that

$$(\boldsymbol{J}^T\boldsymbol{J} + \alpha\boldsymbol{I})\boldsymbol{\Delta} = -\boldsymbol{J}^T\boldsymbol{r}$$

- Solution requires singular value decomposition (SVD) of $\boldsymbol{J}$
- Can yield super-linear convergence rate



Nocedal & Wright, 2006

$f(\boldsymbol{x})$ = function to minimize
$m(\boldsymbol{\Delta})$ = quadratic model function
δ = trust region radius

DESC Optimization Process

0. Initial equilibrium
1. Find optimal perturbation
   - $\boldsymbol{c}_{i+1} = \boldsymbol{c}_i + \Delta\boldsymbol{c}$
2. Perturb equilibrium to maintain force balance
   - $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta\boldsymbol{x}$
3. Refine equilibrium solution
   - A few Gauss-Newton iterations
4. Repeat steps 1-3 until convergence

$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$ = objective function
$\boldsymbol{c}$ = optimization variables
$\boldsymbol{x}$ = equilibrium solution

$$\Delta\boldsymbol{x} = \varepsilon\boldsymbol{x}_1 + \varepsilon^2\boldsymbol{x}_2 + \cdots$$
$$\Delta\boldsymbol{c} = \varepsilon\boldsymbol{c}_1 + \varepsilon^2\boldsymbol{c}_2 + \cdots$$

Optimization Perturbations

- $\boldsymbol{f}_C$ = constraint equations (e.g., equilibrium force balance)
- $\boldsymbol{f}_O$ = objective functions (e.g., quasi-symmetry, etc.)
- Taylor expand $\boldsymbol{f}_C(\boldsymbol{x}+\Delta\boldsymbol{x}, \boldsymbol{c}+\Delta\boldsymbol{c})$ and $\boldsymbol{f}_O(\boldsymbol{x}+\Delta\boldsymbol{x}, \boldsymbol{c}+\Delta\boldsymbol{c})$
- Optimal perturbation:

$$\left[\frac{\partial \boldsymbol{f}_O}{\partial \boldsymbol{x}}\left(\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{x}}\right)^{-1}\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{c}} - \frac{\partial \boldsymbol{f}_O}{\partial \boldsymbol{c}}\right]\varepsilon\boldsymbol{c}_1 = \boldsymbol{f}_O(\boldsymbol{x},\boldsymbol{c}) - \frac{\partial \boldsymbol{f}_O}{\partial \boldsymbol{x}}\left(\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{x}}\right)^{-1}\boldsymbol{f}_C(x,c)$$

- Equilibrium perturbation:

$$\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{x}}\varepsilon\boldsymbol{x}_1 = -\boldsymbol{f}_C(\boldsymbol{x},\boldsymbol{c}) - \frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{c}}\varepsilon\boldsymbol{c}_1$$

- Least-squares Gauss-Newton trust-region steps: $\boldsymbol{J}\boldsymbol{\Delta} = -\boldsymbol{r}$

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

Advantages of DESC Optimization

- All Jacobian and Hessian matrices computed with automatic differentiation
  - Exact derivates, efficient evaluation
- Easily extended to higher-order approximations:

$$\left[\frac{\partial \boldsymbol{f}_O}{\partial \boldsymbol{x}}\left(\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{x}}\right)^{-1}\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{c}}-\frac{\partial \boldsymbol{f}_O}{\partial \boldsymbol{c}}\right]\varepsilon \boldsymbol{c}_2=\left[\frac{\partial^2 \boldsymbol{f}_O}{\partial \boldsymbol{x}\partial \boldsymbol{c}}-\frac{\partial \boldsymbol{f}_O}{\partial \boldsymbol{x}}\left(\frac{\partial \boldsymbol{f}_C}{\partial \boldsymbol{x}}\right)^{-1}\frac{\partial^2 \boldsymbol{f}_C}{\partial \boldsymbol{x}\partial \boldsymbol{c}}\right]\varepsilon^2 \boldsymbol{x}_1 \boldsymbol{c}_1^T+\cdots$$

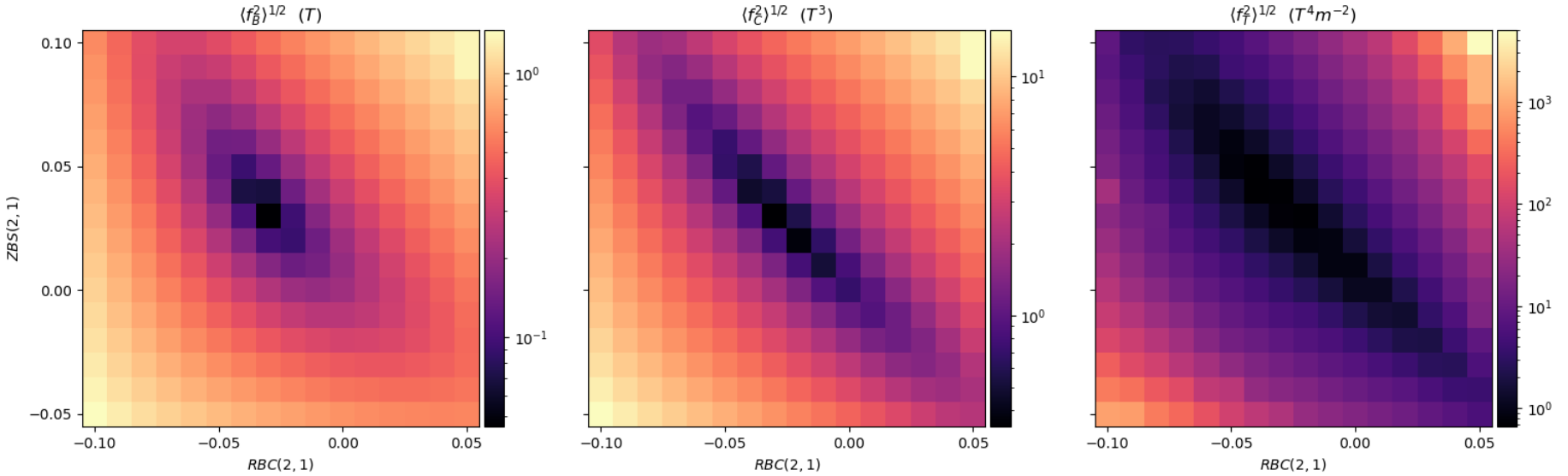
  - LHS matrix is same as 1st order method; SVD already computed!
- RHS computed as Jacobian-vector products
  - Faster evaluation, less memory required
- Only **one** equilibrium solve required per optimization iteration!

Optimization Objective Functions

- DESC is designed to allow arbitrary objective functions, in combination with each other
- Quasi-symmetry
  - Boozer coordinates: $f_B = \sqrt{\sum_{\frac{m}{n} \neq \frac{M}{N}} B_{mn}^2}$
  - Flux function: $f_C = (\boldsymbol{B} \times \boldsymbol{\nabla}\psi) \cdot \boldsymbol{\nabla} B - \left(\frac{MG+NI}{M\iota-N}\right) \boldsymbol{B} \cdot \nabla B$
  - Triple product: $f_T = \boldsymbol{\nabla}\psi \times \boldsymbol{\nabla} B \cdot (\boldsymbol{B} \cdot \boldsymbol{\nabla} B)$
- Others:
  - plasma volume, aspect ratio, target rotational transform, etc.

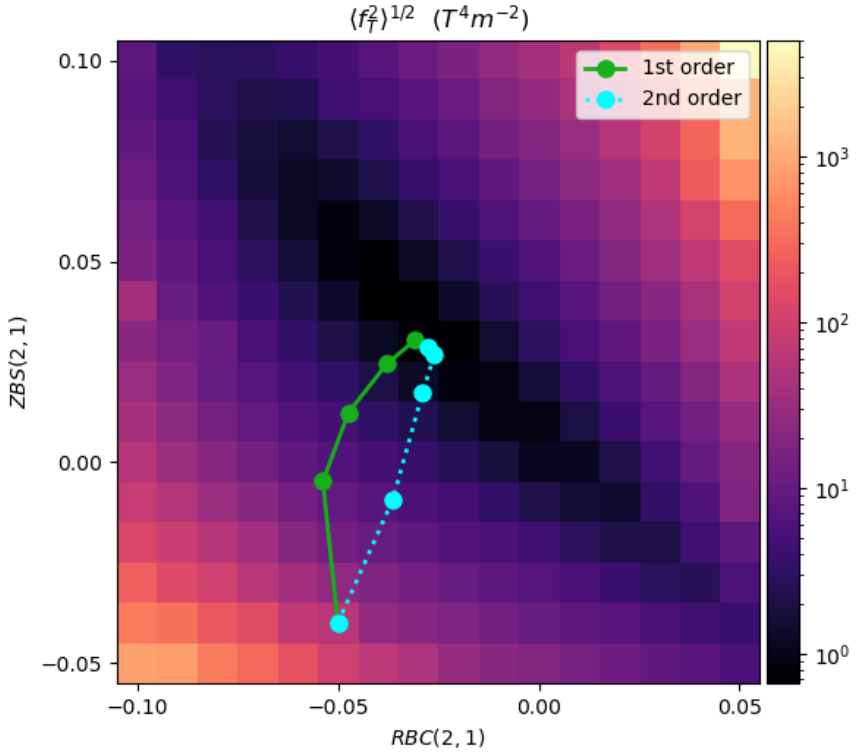
Comparison of Quasi-Symmetry Objectives

- Different QS objectives can have different optimization landscapes
- These are all proxy functions for particle/heat flux



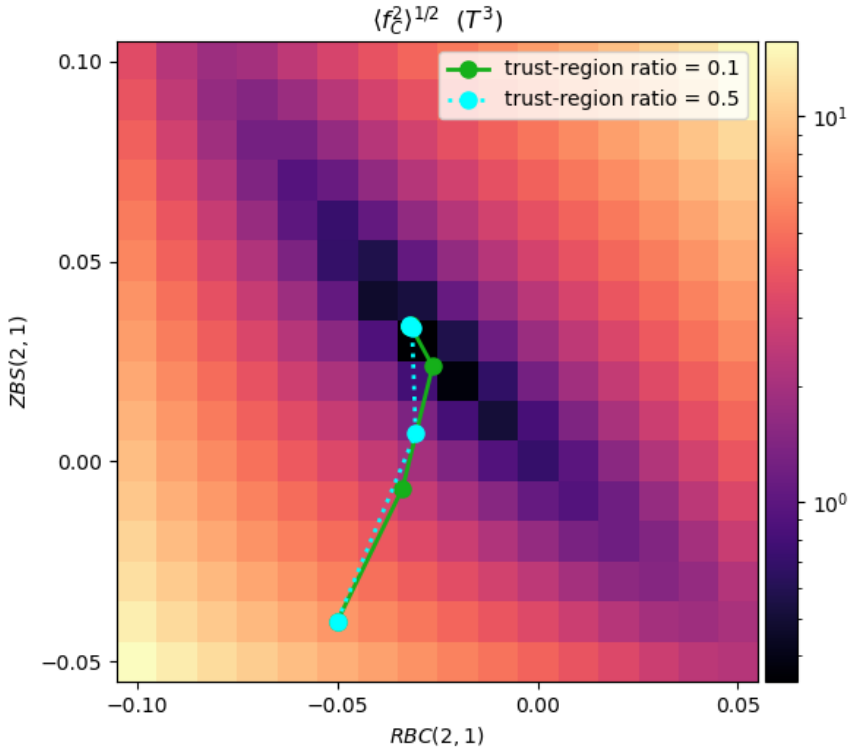
1st vs 2nd order Optimization Methods

- "1st order" is 2nd order in scalar optimization terminology
  - Jacobian of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c})$ gives Hessian of least-squares objective function
- "2nd order" is 3rd order, etc.
- Higher-order methods allow optimizer to take larger steps & more direct path



Trust-Region Radius Sensitivity

- Trust region method depends on choosing an appropriate radius
  - Small radius: more robust
  - Large radius: fewer steps
- Radius can be chosen adaptively based on previous steps
- Default is a small trust region
  - Additional iterations are inexpensive to compute



Speed Comparison

- 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)	$\sim$ 2 hours
STELLOPT (16 CPUs)	$\sim$ 1.5 hours
STELLOPT (32 CPUs)	$\sim$ 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

Future Work

- Improve user interface for running stellarator optimizations
- Improve speed and memory performance
- Refine adaptive trust region radius procedure
- Higher order optimal perturbations
- Add additional objective functions
- Determine proper weighting between objectives
- Your suggestions?

Thank You! Questions?

PP11.00086 - A Comparison of VMEC and DESC 3D Equilibrium Codes

PP11.00087 - Improvements to the DESC code for finding and optimizing stellarator equilibria

Publication: Dudt et. al. *Phys. Plasmas.* 2020.

Repository: `https://github.com/PlasmaControl/DESC`

Python package: `pip install desc-opt`

PPPL

andlinger center for energy+the environment

PRINCETON UNIVERSITY