DESC

Fast and flexible stellarator equilibrium optimization

Daniel Dudt
With Rory Conlin & Dario Panici
Princeton Plasma Control Group

Advisor: Egemen Kolemen

IPP MHD Topical Group
April 15, 2021
Why is *another* equilibrium code needed?

**Understand the solution space of stellarator equilibria**
- 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?

**Integrate stellarator optimization tools into equilibrium solver**
- 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, adjoint methods, near-axis expansions\(^1\), low-\(\beta\) expansions, etc.

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

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

toroidal coordinates: \( (R, \phi, Z) \)

straight field-line coordinates\(^1\): \( (\rho, \theta, \zeta) \)

\[ \rho = \sqrt{\psi / \psi_a} \]

\[ \theta = \theta + \lambda(\theta, \phi) \]

Computational domain:
\[ (\rho, \theta, \zeta) \]

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

Magnetic field is written in flux coordinates

• Assume nested flux surfaces: \( \mathbf{B} \cdot \nabla \rho = 0 \)

\[
\mathbf{B} = B^\theta \mathbf{e}_\theta + B^\zeta \mathbf{e}_\zeta
\]

• Include the constraint of Gauss’s law\(^1\): \( \nabla \cdot \mathbf{B} = 0 \)

\[
\mathbf{B} = \frac{\psi'}{2\pi \sqrt{g}} \left( (1 - \partial_\zeta \lambda) \mathbf{e}_\theta + (1 + \partial_\theta \lambda) \mathbf{e}_\zeta \right)
\]

\[
\mathbf{B}(\rho, \theta, \zeta) = \mathbf{B}(R(\rho, \theta, \zeta), Z(\rho, \theta, \zeta), \lambda(\rho, \theta, \zeta), \iota(\rho))
\]

• Use Ampere’s Law: \( \nabla \times \mathbf{B} = \mu_0 \mathbf{J} \)

\[
\mathbf{J}(\rho, \theta, \zeta) = \mathbf{J}(R(\rho, \theta, \zeta), Z(\rho, \theta, \zeta), \lambda(\rho, \theta, \zeta), \iota(\rho))
\]

Equilibrium is obtained when the force balance residuals vanish

\[ F \equiv J \times B - \nabla \rho = 0 \]

• Substitute in \( B \) and \( J \):

\[ F = F_\rho \nabla \rho + F_\beta \beta \]

\[ F_\rho = \sqrt{g} (B^\zeta J^\theta - B^\theta J^\zeta) - \rho' \]

\[ F_\beta = \sqrt{g} B^\zeta J^\rho \]

\[ \beta = \nabla \theta - i \nabla \zeta \]

• Convert to scalar equations (weighted with volume elements):

\[ f_\rho = F_\rho \|\nabla \rho\|_2 \Delta V \]

\[ f_\beta = F_\beta \|\beta\|_2 \Delta V \]

Equilibrium force balance is solved as a system of nonlinear equations

\[ f(x, c) \approx 0 \]

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

- **Pseudo-spectral approach**
  - Solution defined by global basis functions
  - Minimizes residuals at collocation nodes
  - Exponential convergence if smooth

\[ x^* = \operatorname{argmin}_x (\|f(x, c)\|^2) \]

- **Flexible choice of solver**
  - Modified Newton methods, least-squares minimization, custom routines, etc.
  - Quadratic convergence near solution

- **Written in Python3**
  - Object-oriented structure is easy to use, extend, and interface with other codes

- **Uses JAX**
  - Developed by Google, same backend as TensorFlow
  - Automatic differentiation for exact derivatives of arbitrary order
  - JIT compilation to CPU & GPU via XLA
Global Fourier-Zernike\textsuperscript{1,2} spectral bases are used for discretization

\[ R(\rho, \theta, \zeta) = \sum R_{lmn} Z_l^m(\rho, \theta) F^n(\zeta) \]

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

\[
\begin{align*}
 f(\rho, \theta) &= \sum_m \rho^m (a_{m,0} + a_{m,2}\rho^2 + \cdots) \cos(m\theta) \\
 &+ \sum_m \rho^m (b_{m,0} + b_{m,2}\rho^2 + \cdots) \sin(m\theta)
\end{align*}
\]

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

DESC shows strong convergence and agreement with VMEC

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.

![W7X force balance error, $\beta=2\%$](chart.png)
DESC’s equilibrium solver speed is comparable to VMEC

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

<table>
<thead>
<tr>
<th>W7-X resolution: M=12, N=12</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESC on CPU*</td>
</tr>
<tr>
<td>VMEC on CPU*</td>
</tr>
<tr>
<td>DESC on GPU</td>
</tr>
</tbody>
</table>

- 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\textsuperscript{1}):  
  - $\sim O(\text{hours} – \text{days})$  
  - Limited accuracy

- Adjoint method (VMEC + ALPOpt\textsuperscript{2}):  
  - $\sim O(\text{minutes} – \text{hours})$  
  - Difficult to implement  
  - Not applicable for all objectives / derivative orders

- Automatic differentiation (DESC)  
  - $\sim O(\text{seconds} – \text{minutes})$  
  - 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)

\textsuperscript{1}Lazerson et. al., \textit{STELLOPT}, (2020).  
\textsuperscript{2}Paul et. al., \textit{J. Plasma Phys.}, (2021).
Perturbations provide approximations to neighboring solutions

• First-order Taylor expansion:

\[
\begin{align*}
    f(x + \Delta x, c + \Delta c) &= f(x, c) + \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial c} \Delta c = 0 \\
    \Delta x &= -\left(\frac{\partial f}{\partial x}\right)^{-1} \left( f(x, c) + \frac{\partial f}{\partial c} \Delta c \right)
\end{align*}
\]

• Derivatives are computed exactly and efficiently with automatic differentiation

• SVD of Jacobian matrix \( \left(\frac{\partial f}{\partial x}\right)^{-1} \) is already computed in the final Newton iteration when solving the equilibrium

• Has been extended to third-order in the code
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
Vacuum tokamak

Finite-β tokamak

Finite-β stellarator

Continuation Example: W7-X

(Second-order perturbations)
Optimization objectives can be incorporated into the perturbations

- Define an optimization cost function $g \equiv g(x, c)$
  - Quasi-symmetry, coil complexity, etc.
- First-order Taylor expansion:
  $$g(x + \Delta x, c + \Delta c) = g(x, c) + \frac{\partial g}{\partial x} \Delta x + \frac{\partial g}{\partial c} \Delta c = 0$$
  $$\left[ \frac{\partial g}{\partial x} \left( \frac{\partial f}{\partial x} \right)^{-1} \frac{\partial f}{\partial c} - \frac{\partial g}{\partial c} \right] \Delta c = g(x, c) - \frac{\partial g}{\partial x} \left( \frac{\partial f}{\partial x} \right)^{-1} f(x, c)$$
- This yields the perturbation $\Delta c$ that most improves the objective
- No additional equilibrium solves required!
Example optimization objective: quasi-symmetry

\[ g(x, c) \equiv \nabla \psi \times \nabla B \cdot \nabla (B \cdot \nabla)^1 \]

- Optimized the boundary shape for quasi-symmetry at the \( \rho = 0.9 \) flux surface

- Second-order method in development

---

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
Future Work

• Ideal MHD stability
• Free-boundary equilibria
• Combined coil/plasma optimization
• Magnetic islands & stochastic regions
Thank You!

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:

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

• Jacobian: $\sqrt{g} = e_\rho \cdot e_\theta \times e_\zeta$
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:
  
  o Control of grid refinement
  
  o Avoidance of rational surfaces
Solving force balance does not enforce $\nabla \cdot J = 0$

Ampere’s Law: $\nabla \times B = \mu_0 J$

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

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

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

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

<table>
<thead>
<tr>
<th>Order of Derivatives</th>
<th>Variables</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$R, Z$</td>
<td>$\nabla \cdot B = 0$</td>
</tr>
<tr>
<td>1</td>
<td>$\partial_i R, \partial_i Z \rightarrow B$</td>
<td>$J \times B = \nabla p$</td>
</tr>
<tr>
<td>3</td>
<td>$\partial_{ijk} R, \partial_{ijk} Z \rightarrow \nabla \cdot J = 0$</td>
<td></td>
</tr>
</tbody>
</table>
Initial vacuum tokamak

$\zeta \cdot NFP/2\pi = 0.000$

$Z$ (m)

$R$ (m)

2nd-order pressure perturbation

$\zeta \cdot NFP/2\pi = 0.000$

$Z$ (m)

$R$ (m)

True solution with pressure

$\zeta \cdot NFP/2\pi = 0.000$

$Z$ (m)

$R$ (m)
Initial tokamak solution

$\zeta \cdot \text{NFP}/2\pi = 0.000$

2nd order boundary perturbation

$\zeta \cdot \text{NFP}/2\pi = 0.000$

$\zeta \cdot \text{NFP}/2\pi = 0.222$

True stellarator solution

$\zeta \cdot \text{NFP}/2\pi = 0.000$

$\zeta \cdot \text{NFP}/2\pi = 0.222$
W7X force balance error, $\beta=0$

Flux surface average $|F|/|\nabla(B^2)|$

- 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, $M=14, N=14$
VMEC DSHAPE Normalized Force Error Convergence

- M = 14
- M = 16
- M = 18

Average Normalized Force Error

Radial Resolution

DESC DSHAPE Normalized Force Error Convergence

Average Normalized Force Error

Radial resolution, L, at M=14
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

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