J. Plasma Phys. (2023), *vol.* 89, 955890305
Published by Cambridge University Press

doi:10.1017/S0022377823000399

The DESC stellarator code suite. Part 2. Perturbation and continuation methods

Rory Conlin[1,†], Daniel W. Dudt[1], Dario Panici[1] and Egemen Kolemen[1,†]

[1]Princeton University, Princeton, NJ 08544, USA

(Received 31 October 2022; revised 4 April 2023; accepted 5 April 2023)

A new perturbation and continuation method is presented for computing and analysing stellarator equilibria. The method is formally derived from a series expansion about the equilibrium condition $\boldsymbol{F} \equiv \boldsymbol{J} \times \boldsymbol{B} - \nabla p = 0$, and an efficient algorithm for computing solutions to second- and third-order perturbations is developed. The method has been implemented in the DESC stellarator equilibrium code, using automatic differentiation to compute the required derivatives. Examples are shown demonstrating its use for computing complicated equilibria, perturbing a tokamak into a stellarator and performing parameter scans in pressure, rotational transform and boundary shape in a fraction of the time required for a full solution.

Key words: fusion plasma, plasma devices, plasma simulation

1. Introduction

In the search for controlled nuclear fusion, three-dimensional (3-D) magnetic confinement devices such as stellarators have been shown to have several advantages over two-dimensional (2-D) magnetic geometries such as tokamaks, such as lower risk of disruption (Helander *et al.* 2012) and current-free steady-state operation (Helander 2014). However, achieving good performance in a stellarator often requires significant optimization of the plasma equilibrium. An additional advantage is that because of the generally lower plasma current and consequently fewer instabilities, more of the design and optimization can be done computationally, without requiring the building and testing of full-scale devices (Boozer 2015). Despite this, designing and optimizing 3-D magnetic equilibria that have good properties is still a computationally intensive task, for which a number of codes and software packages have been developed (Hirshman & Whitson 1983*a*; Spong *et al.* 1998; Drevlak *et al.* 2019; Dudt & Kolemen 2020; Lazerson *et al.* 2020; Landreman *et al.* 2021; Hudson *et al.* 2012).

Perturbation methods have been used heavily in tokamak plasma physics, primarily to analyse the stability of axisymmetric magnetohydrodynamic (MHD) equilibria by searching for a perturbation that minimizes the energy of the plasma, as in Bernstein's

† Email addresses for correspondence: wconlin@princeton.edu; ekolemen@princeton.edu

energy principle (Bernstein *et al.* 1958). This has been extended to a wide range of of codes for analysing fusion devices under small perturbations to a MHD equilibrium, such as the GPEC suite of codes (Park, Boozer & Glasser 2007; Park *et al.* 2009; Glasser 2016; Glasser, Kolemen & Glasser 2018) for analysing tokamak configurations under 3-D perturbations. A common feature of perturbation methods is using local approximation methods (commonly Taylor series) to examine solutions nearby to some equilibrium in an infinitesimal limit. In the present work, we extend this to include finite-magnitude perturbations, and extend the approximation to second and higher order to achieve increased accuracy for finite step sizes. While a perturbation usually refers to a single step in parameter space, a sequence of perturbations can be combined into a continuation method to further explore the phase space, where a single perturbation step is used to approximate a nearby solution, and Taylor approximation is recomputed at the new point before performing another finite-size perturbation and so on.

Continuation methods have received less attention in the fusion community, though they have seen extensive use in other fields (e.g. Richter & DeCarlo 1983; Howell 2009). For the purposes of the present work, continuation methods can be used to solve parameterized equations of the form $F(x, \eta) = 0$, where we identify x as the solution vector and η as a continuation parameter. When η is varied we find a family of solutions connected in parameter space, as well as possible branches and bifurcations of this family, indicated by points where the Jacobian of F is singular. Starting from the initial value (x_0, η_0) where F is zero, we continuously vary η while simultaneously varying x such that the equation $F(x, \eta) = 0$ is satisfied. Various methods exist for finding the solution curve (the locus of points (x, η) where $F(x, \eta) = 0$ is satisfied) such as piecewise linear (simplex) continuation and psuedo-arclength methods (Allgower & Georg 1990). In an abstract sense, the standard multigrid method for solving partial differential equations can be though of as a discrete continuation method, where the continuation parameter η governs the level of numerical resolution (i.e. grid spacing, number of basis functions, etc.).

In general, existing codes for computing stellarator equilibria (Hirshman & Whitson 1983*a*; Hudson *et al.* 2012) only find discrete equilibria, and solving for a new equilibrium requires running the code from scratch (possibly with a different starting guess). Several questions that may then be asked are:

(i) Can these equilibria be computed more efficiently?
(ii) Given a single equilibrium solution, can we find other similar solutions?
(iii) What does the full phase space of 3-D MHD equilibria look like?

This is Part 2 of a three-part series of papers on the DESC stellarator optimization code suite. Part 1 details the DESC equilibrium solver in comparison with the VMEC (Hirshman & Whitson 1983*b*) 3-D MHD equilibrium code. Computing 3-D MHD equilibria is the preliminary step in stellarator analysis, so computing these equilibria quickly and accurately is important for further studies. In this paper, we describe a new continuation method for computing complicated stellarator equilibria using perturbations that attempts to resolve these questions. In § 2 we describe the DESC code and why it is the ideal code for implementation of the methods described in this paper, while in § 3 we describe the mathematical background to the perturbation method. In § 4 we demonstrate how to use these perturbations in a continuation method for computing stellarator equilibria. In § 5 we demonstrate other applications of the perturbation method for computing and analysing stellarator equilibria. Part 3 (Dudt *et al.* 2022) presents DESC's unique stellarator optimization capabilities made possible by the efficient equilibrium solver and the perturbation method, resulting in orders-of-magnitude speed-up in optimization. These advantages are shown in the context of quasi-symmetry optimization, where results are compared with conventional tools (Spong *et al.* 1998). Three different quasi-symmetry objective formulations are also shown, with the relative advantages of each compared, highlighting the flexibility of DESC as an optimization code.

2. The DESC code

DESC is a recently developed (Dudt & Kolemen 2020) pseudospectral code for computing 3-D MHD equilibria. DESC computes 3-D MHD equilibria by solving the force balance equations $\boldsymbol{J} \times \boldsymbol{B} - \nabla p = 0$ as opposed to the more common variational method which minimizes the MHD energy $\int_V (B^2/2\mu_0 + p)\, \mathrm{d}V$. The independent variables in the equation are the positions of the flux surfaces (R, Z) as well as the stream function λ, defined as the difference between the boundary poloidal angle θ and the straight-field-line poloidal angle $\vartheta = \theta + \lambda$. These quantities are discretized in a Fourier–Zernike basis, using a Fourier series in the toroidal direction and Zernike polynomials in the poloidal/radial directions. After discretization, the equilibrium equation is expressed as a set of nonlinear algebraic equations $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0$, where $\boldsymbol{x}$ is a vector containing the spectral coefficients of R, Z and λ, while $\boldsymbol{c}$ contains fixed parameters that define the equilibrium problem, such as the pressure and rotational transform profiles and the fixed boundary shape.

Since its original publication (Dudt & Kolemen 2020), DESC has undergone a major upgrade that involved porting it from MATLAB to Python in order to take advantage of the JAX library (Bradbury *et al.* 2018) for automatic differentiation and just-in-time compilation. Initially developed for machine learning applications, JAX provides a NumPy-like (Van Der Walt, Colbert & Varoquaux 2011; Harris *et al.* 2020) application programming interface for common mathematical operations and allows arbitrary functions to be differentiated using forward- or reverse-mode automatic differentiation. This allows the calculation of exact derivatives of the objective function automatically, rather than having to code them by hand which is time-consuming and error prone, or using finite differences which are computationally expensive and can be inaccurate. It is also much more flexible as new objective functions can be added and optimized by defining only the forward pass, which is of great use in stellarator optimization where new objectives may be added in the future. JAX also allows for just-in-time compiling of code to both central and graphics processing units which significantly speeds up calculation, approaching speeds of traditional compiled languages, avoiding one of the primary limitations of Python for scientific computing. Additionally, given that the vast majority of new supercomputers heavily leverage graphics processing units, and this trend is likely to continue, using JAX allows DESC to take full advantage of all the compute capability available, rather than being limited to central-processing-unit-only parallelization like many legacy codes. This allows a ‘best of both worlds’ approach where the code is easy to use, maintain, adapt and upgrade, while still being fast enough for production applications.

Several aspects also make DESC the ideal code for implementing the perturbation method outlined in § 3:

(i) Using a pseudospectral discretization significantly reduces the number of independent variables, resulting in smaller Jacobian matrices.
(ii) Using JAX allows fast and accurate computation of the required derivatives and Jacobian–vector products.

(iii) Formulating the problem as a system of nonlinear equations and solving them in a least-squares sense also effectively gives an extra order of derivative for free.

This last point can be seen by considering the least-squares problem

$$\min_{x} y(\boldsymbol{x}) \equiv \frac{1}{2}\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{f}(\boldsymbol{x}), \tag{2.1}$$

where $\boldsymbol{f}$ is a vector valued function and y is the sum of squares of the residuals of $\boldsymbol{f}$. The gradient of y is given by

$$\frac{\mathrm{d}y}{\mathrm{d}\boldsymbol{x}} = \boldsymbol{f}^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}, \tag{2.2}$$

and its Hessian (matrix of second partial derivatives) is given by

$$\frac{\mathrm{d}^2y}{\mathrm{d}\boldsymbol{x}^2} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}} + \boldsymbol{f}^{\mathrm{T}}\frac{\mathrm{d}^2\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}^2}. \tag{2.3}$$

Given that $\boldsymbol{f}$ is the function we are trying to minimize in the least-squares sense, we can generally assume that the second term in the Hessian is negligible compared with the first, the so-called 'small residual approximation' (Nocedal & Wright 2006). This gives an approximate Hessian $\mathrm{d}^2y/\mathrm{d}\boldsymbol{x}^2 \sim (\mathrm{d}\boldsymbol{f}/\mathrm{d}\boldsymbol{x}^{\mathrm{T}})(\mathrm{d}\boldsymbol{f}/\mathrm{d}\boldsymbol{x})$, meaning that using only first-derivative information about $\boldsymbol{f}$ gives us both first- and second-derivative information about y. In addition, the approximate Hessian that this gives is always positive semi-definite, leading to a convex subproblem which is easy to solve.

3. Perturbations

A general fixed-boundary equilibrium problem can be described by a set of parameters $\boldsymbol{c} = \{R_b, Z_b, p, \iota, \Psi\}$, where R_b, Z_b are the R, Z coordinates of the boundary surface, p is the pressure profile, ι is the rotational transform and Ψ is the total toroidal flux through the torus. In many spectral equilibrium codes (Hirshman & Whitson 1983*a*; Dudt & Kolemen 2020), the independent variables that define the equilibrium can be given by $\boldsymbol{x} = [R_{lmn}, Z_{lmn}, \lambda_{lmn}]$, where R_{lmn}, Z_{lmn} and λ_{lmn} are spectral coefficients of the flux surface positions and the poloidal stream function (indexed by l in the radial direction, m in the poloidal direction and n in the toroidal direction). The condition of MHD equilibrium can then be written as a (possibly vector-valued) nonlinear algebraic equation involving the fixed parameters and independent variables, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0$. The function f is the discretized form of the general MHD force balance $\boldsymbol{J} \times \boldsymbol{B} - \nabla p = 0$ (Dudt & Kolemen 2020), or a condition on the gradient of the energy functional $W = \int_V (B^2/2\mu_0 + p)\,\mathrm{d}V$ (Hirshman & Whitson 1983*a*).

Given a set of parameters $\boldsymbol{c}$ and a solution vector $\boldsymbol{x}$ which satisfy $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0$, we wish to find how the equilibrium would change if the parameters $\boldsymbol{c}$ are perturbed to $\boldsymbol{c} + \Delta\boldsymbol{c}$. For instance, we may have a solution for a vacuum equilibrium and want to see how adding finite pressure changes it, or we may start with a 2-D tokamak solution and add a 3-D perturbation to the boundary shape to form a stellarator.

We assume that the new equilibrium is given by $\boldsymbol{x} + \Delta\boldsymbol{x}$, and expand $\boldsymbol{f}$ in a Taylor series[1]
:

$$\boldsymbol{f}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}\Delta\boldsymbol{c} + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\Delta\boldsymbol{x}\Delta\boldsymbol{x} + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{c}^2}\Delta\boldsymbol{c}\Delta\boldsymbol{c} + \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{c}}\Delta\boldsymbol{x}\Delta\boldsymbol{c} + O(\Delta x^3). \tag{3.1}$$

We wish to solve this equation for $\Delta\boldsymbol{x}$ such that $\boldsymbol{f}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) = 0$. At first order this is a straightforward algebraic equation, but at higher orders it becomes a tensor polynomial equation which can be difficult or impossible to solve efficiently (for example, if $\boldsymbol{f}$ is a vector-valued function, just storing the second-derivative tensor in memory could require upwards of 100 GB). Instead of seeking a direct solution, we can try a perturbative approach, were we introduce an arbitrary small parameter ϵ and further expand $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{c}$ in a perturbation series in powers of ϵ (we assume that $\Delta\boldsymbol{c}$ is known *a priori* and so only a first-order term is required):

$$\Delta\boldsymbol{x} = \epsilon\boldsymbol{x}_1 + \epsilon^2\boldsymbol{x}_2 + \cdots, \tag{3.2}$$

$$\Delta\boldsymbol{c} = \epsilon\boldsymbol{c}_1. \tag{3.3}$$

Plugging this into (3.1) (and setting $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{f}(\boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{c} + \Delta\boldsymbol{c}) = 0$) we get

$$0 = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\epsilon\boldsymbol{x}_1 + \epsilon^2\boldsymbol{x}_2) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}\epsilon\boldsymbol{c}_1 + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}(\epsilon\boldsymbol{x}_1 + \epsilon^2\boldsymbol{x}_2)(\epsilon\boldsymbol{x}_1 + \epsilon^2\boldsymbol{x}_2) + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{c}^2}\epsilon\boldsymbol{c}_1\epsilon\boldsymbol{c}_1 + \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{c}}(\epsilon\boldsymbol{x}_1 + \epsilon^2\boldsymbol{x}_2)\epsilon\boldsymbol{c}_1 + O(\Delta x^3). \tag{3.4}$$

We can then collect powers of ϵ and set each order of ϵ to zero in turn. The first-order equation gives

$$0 = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\epsilon\boldsymbol{x}_1 + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}\epsilon\boldsymbol{c}_1, \tag{3.5}$$

$$x_1 = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1}\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{c}}c_1\right). \tag{3.6}$$

The second-order term gives

$$0 = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\epsilon^2\boldsymbol{x}_2 + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\epsilon^2\boldsymbol{x}_1\boldsymbol{x}_1 + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{c}^2}\epsilon^2\boldsymbol{c}_1\boldsymbol{c}_1 + \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{c}}\epsilon^2\boldsymbol{x}_1\boldsymbol{c}_1, \tag{3.7}$$

$$\boldsymbol{x}_2 = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^{-1}\left(\frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}^2}\boldsymbol{x}_1\boldsymbol{x}_1 + \frac{1}{2}\frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{c}^2}\boldsymbol{c}_1\boldsymbol{c}_1 + \frac{\partial^2 \boldsymbol{f}}{\partial \boldsymbol{x}\partial \boldsymbol{c}}\boldsymbol{x}_1\boldsymbol{c}_1\right). \tag{3.8}$$

In general, the second-derivative terms will be large, dense, rank 3 tensors (recall that can be a vector-valued function), which may be extremely expensive to compute and may

[1]Some care must be taken here, as we are implicitly assuming that the function f is at least C^2 continuous. The DESC code assumes the existence of nested flux surfaces so that there is an analytic mapping between real space and magnetic coordinates. In cases where the assumption of nested surfaces is violated, this mapping may not exist. However, we note that the force balance equations are still analytic functions of the dependent variables (it can be shown that they form a high-order polynomial), though their physical meaning will be somewhat unclear in regions where nested surfaces do not exist.

not even fit in memory for high-resolution cases. However, it is important to note that the full second derivatives are never needed (or indeed full first derivatives apart from the Jacobian matrix $\partial \boldsymbol{f}/\partial \boldsymbol{x}$). All that is needed are directional derivatives or Jacobian vector products. In the DESC code (§ 2) these are calculated using forward mode automatic differentiation at a cost a few times greater than the cost of a single evaluation of $\boldsymbol{f}$. These terms could also be approximated using finite differences at a similar cost. This means that at each order the most expensive operation is solving a linear system of the form

$$\boldsymbol{J}\boldsymbol{x}_i = \boldsymbol{b}, \tag{3.9}$$

where $\boldsymbol{J} \equiv \partial \boldsymbol{f}/\partial \boldsymbol{x}$ is the Jacobian matrix and is the same at each order, so only needs to be decomposed once, and i is the order of the term in the perturbation series. Because of these computation and memory savings, the method has been extended to third order in the DESC code (see § 2), though in most situations first- or second-order perturbations give acceptable results (see figure 2).

It is instructive to note that the first-order term in the perturbation series for $\Delta \boldsymbol{x}$ is effectively the Newton step, while the second-order term is commonly referred to as the Halley step in optimization literature (Gander 1985; Gundersen & Steihaug 2010). While Newton's method converges quadratically (Nocedal & Wright 2006) when started sufficiently close to a solution, if started far away it may diverge and so in practice the method must be globalized using either a line search or a trust region framework. Although we assume that we begin near or at an equilibrium so that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0$, depending on the size of the perturbation the solution landscape may change such that this is no longer the case, and we find in practice in several cases the unconstrained Newton step causes the solution to diverge. There is also an additional constraint inherent in the approach taken in that the second-order term should be smaller than the first-order term by a factor $\epsilon \ll 1$.

This suggests that a natural way to enforce these conditions and ensure reasonable convergence properties is to use a trust region method. When expanding the objective function in a Taylor series we are effectively approximating it by a linear or quadratic function. Instead of seeking the global minimizer to these model functions, the trust region approach instead recognizes that the Taylor approximation is only valid in some small neighbourhood and restricts the step size accordingly. Mathematically, instead of finding the exact solution $\boldsymbol{x}_i^*$ to the linear system

$$\boldsymbol{J}\boldsymbol{x}_i^* = \boldsymbol{b}, \tag{3.10}$$

we instead seek a solution to the following optimization problem:

$$\min_{\boldsymbol{x}_i} \|\boldsymbol{J}\boldsymbol{x}_i - \boldsymbol{b}\|^2 \quad \text{s.t. } \|\boldsymbol{x}_i\| \leq r, \tag{3.11}$$

where r is the radius of the trust region. This subproblem has two possible solutions (Nocedal & Wright 2006): either the true solution $\boldsymbol{x}_i^*$ lies within the trust region or else that there is a scalar $\alpha > 0$ such that

$$(\boldsymbol{J} + \alpha \boldsymbol{I})\boldsymbol{x}_i = \boldsymbol{b}, \quad \|\boldsymbol{x}_i\| = r. \tag{3.12}$$

The constrained value for $\boldsymbol{x}_i$ can be efficiently found using a one-dimensional root-finding method in α. In the DESC code we use a bracketed Newton method. The cost of solving this subproblem is roughly the cost of one singular value decomposition of the Jacobian $\boldsymbol{J}$, or two to three Cholesky factorizations of the approximate Hessian matrix $\boldsymbol{B} \equiv \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}$.

Selection of the trust region radius r is traditionally done adaptively within an optimization loop, by comparing the actual reduction in the residual at each step with that predicted by the model function. In a perturbation, only a single 'step' is taken, and we have found empirically that setting the trust region at a small fraction of $\|\boldsymbol{x}\|$ generally gives good results, typically $r = 0.1\|\boldsymbol{x}\|$. For second- and higher-order perturbations, such a large trust region can lead to inaccurate results. This can be understood by considering the expansion made in (3.2), where it was implicitly assumed that the second-order correction was a factor of ϵ smaller than the first-order term. Using a large trust region for the first-order term ($r = 0.1\|\boldsymbol{x}\|$) and a smaller trust region for the higher-order terms correctly enforces this assumed scaling. In practice we obtained good performance using $r_i = 0.1\|\boldsymbol{x}_{i-1}\|$, so the second-order term is an order of magnitude less than the first-order term and so on for higher orders.

4. Continuation method

Many MHD equilibrium codes (Hirshman & Whitson 1983*a*; Hudson *et al.* 2012) use a multigrid approach, where an initial guess is specified on a coarse grid, and the error is minimized on that grid before being interpolated to a finer grid and re-solved, continuing until the finest resolution level has been reached. This is done both to speed computation by doing more calculations on coarser grids, and also to make it more robust to poor initial guesses and avoid additional saddle points that can appear in high-dimensional spaces (Dauphin *et al.* 2014). DESC offers this as well, allowing the resolution to be varied in the radial, poloidal and toroidal directions independently.

In addition to the standard multigrid approach, DESC has also implemented a new continuation method using the perturbation techniques described in § 3. In general, continuation methods seek to find how the solution to an equation varies as parameters are changed. Using the notation of § 3 we can view $\boldsymbol{x}$ to be an implicit function of $\boldsymbol{c}$, related by the constraint equation $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0$, and try to find the map that relates $\boldsymbol{x}$ and $\boldsymbol{c}$ along lines of $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0$.

In this work, we note two properties that make continuation methods especially relevant:

(i) Two-dimensional (axisymmetric) equilibria are much easier to compute than 3-D equilibria, due to the guaranteed existence of flux surfaces in axisymmetric equilibria, and the significant reduction in the number of variables needed to represent the solution.

(ii) Vacuum (vanishing beta) equilibria are easier to compute than finite-pressure equilibria, due to the lack of Shafranov shift.

We can take advantage of these properties to compute complicated stellarator equilibria by first solving an 'easy' problem, and then introduce a continuation parameter that transforms our initial solution into the solution to a 'hard' problem.

To do this, we introduce two scalar continuation parameters: η_b, a multiplier on all of the non-axisymmetric boundary modes, and η_p, a scaling factor for the pressure profile. Setting $\eta_b = 0$ would give a 2-D axisymmetric boundary that is 'close' to the desired 3-D equilibrium, while setting $\eta_p = 0$ would give the zero pressure equilibrium with the desired boundary shape. By varying these two parameters, we find a 'family' of solutions that are connected continuously.

To vary the parameters, we apply a sequence of perturbation steps as outlined in § 3. With each perturbation we take a small step in the desired direction in parameter space (e.g. varying $\boldsymbol{c} = \{\eta_b, \eta_p\}$). Depending on the step size $\Delta\boldsymbol{c}$ it may be necessary to refine the solution with a small number (2–5) of Newton iterations of (2.1) to ensure a good

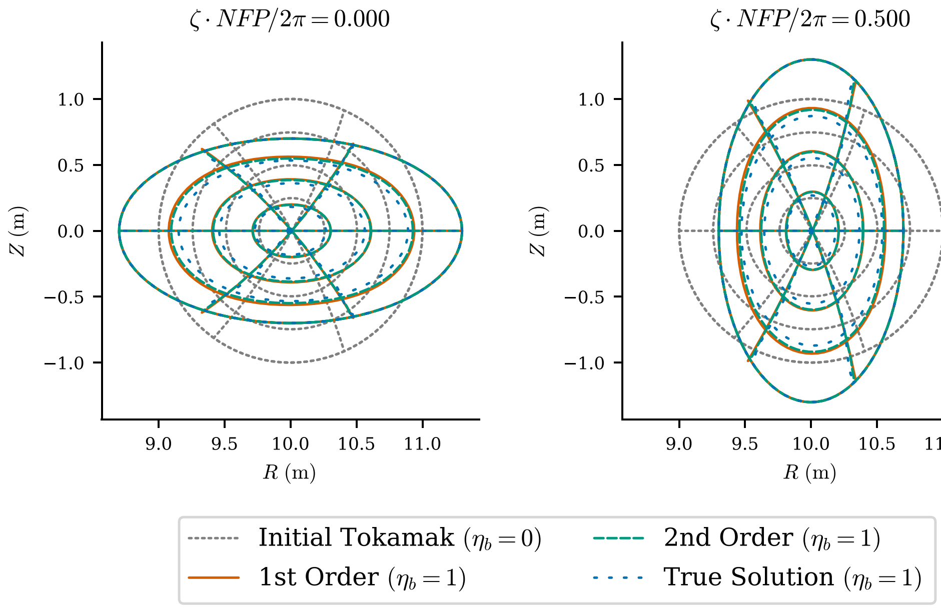

FIGURE 1. Comparison of perturbed tokamak equilibrium versus true solution for a heliotron-like stellarator. The initial tokamak (grey) is axisymmetric. By applying a perturbation (first order in red, second order in green) to the non-axisymmetric boundary modes, we obtain an approximation to the true 3-D solution (blue), obtained by solving from the start with the full 3-D boundary shape. The first-order perturbation captures the majority of the differences between the initial and true solutions. The second-order effects bring the perturbed solution even closer to the true one.

equilibrium is reached. We then re-linearize about the new position and perform the next step, until the final desired parameters are reached.

A standard method for solving continuation problems, known as ‘natural parameter continuation’, would be analogous to a zeroth-order perturbation followed by several Newton iterations of (2.1), while a first-order perturbation would be similar to Gauss–Newton continuation. The higher-order perturbations discussed above do not seem to have been explored in the more general continuation method literature, but can be seen as a version of Halley’s method applied to the combined system $\{\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{c}) = 0, \boldsymbol{c} - \boldsymbol{c}_{\text{desired}} = 0\}$.

In most traditional continuation methods, the step size in the continuation parameter is determined adaptively based on local error estimates and the ratio of predicted to achieved error reduction. In practice we have found that this is often not necessary, and fixing the step sizes *a priori* provides sufficiently accurate results. In most cases going from a zero-pressure equilibrium to a moderate beta of $\sim 3\,\%$ can be done in a single step, and boundary perturbations in anywhere from one to four steps, depending on the desired accuracy and complexity of the boundary. This does require the user to specify the desired perturbation steps explicitly, though a future upgrade to the code is planned to allow adaptive perturbations where the user need only supply an initial guess and the final desired parameters.

As a first example (figure 1), we solve for a zero-pressure heliotron by first solving for a simple circular tokamak, then applying a 3-D perturbation to the boundary. After the perturbation, a small number of regular Newton iterations of (2.1) are applied to ensure

Perturbation order	Mean flux surface error $\left(\frac{1}{V}\int_V \|\boldsymbol{r}_{\text{true}} - \boldsymbol{r}_{\text{perturbed}}\|\,\mathrm{d}^3\boldsymbol{r}\right)$
Initial tokamak	0.100 m
First order	0.031 m
Second order	0.026 m

TABLE 1. Mean flux surface position error versus perturbation order for boundary perturbation from a circular tokamak to a heliotron.

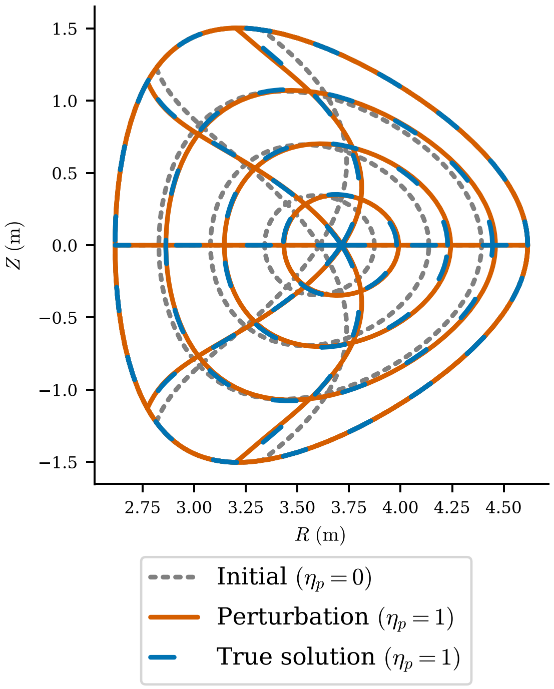

FIGURE 2. Comparison of perturbed equilibrium versus true solution for a change in pressure from $\beta = 0\,\%$ to $\beta = 3\,\%$. The first-order perturbation (red) captures the majority of the differences between the initial (grey) and true (blue) solutions. Second- and third-order effects (not shown for clarity) are visually very similar but significantly reduce the resulting force balance error as shown in table 2.

convergence. As shown in figure 1 and table 1, the first order perturbation step captures the dominant effects of the shaping, while the second order correction has a smaller effect.

Similarly, when computing finite-pressure equilibria it can be difficult to estimate the Shafranov shift *a priori* for generating a good initial guess. DESC avoids this by first computing a zero-pressure solution, for which a good initial guess can generally be found by simply scaling the boundary surface. The finite pressure is then added back in as a perturbation, which then often only requires a small number of further iterations to reach convergence as shown in figure 2. In these and the following examples, the rotational transform profile is held fixed during the perturbations, resulting in a change in the toroidal current as the pressure and boundary are varied. In this example, the toroidal current before

Perturbation order	Normalized force error ($\lvert F\rvert/\lvert\nabla p\rvert$)
First order	87.6 %
Second order	29.6 %
Third order	9.7 %

TABLE 2. Normalized force balance error versus perturbation order for a pressure increase from $\beta = 0\,\%$ to $\beta = 3\,\%$.

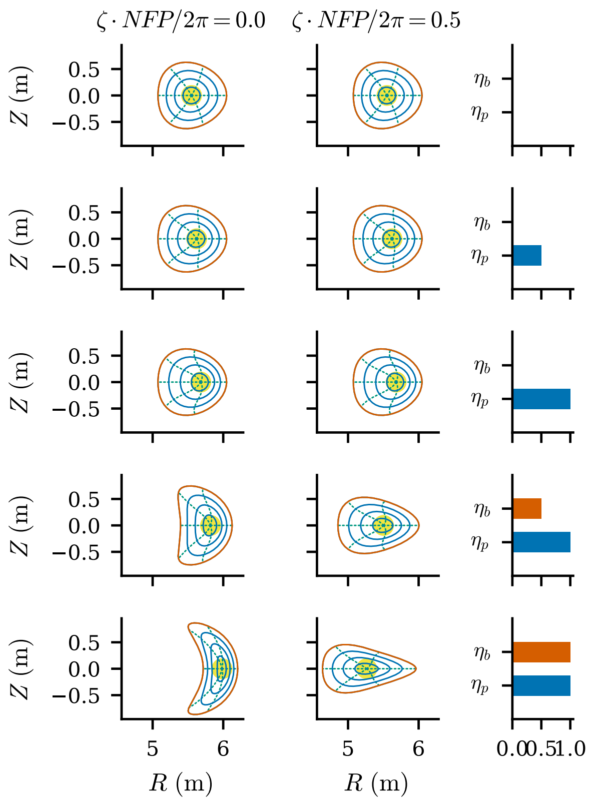

FIGURE 3. Continuation solution for W7-X-like equilibrium. Starting from a zero-pressure 2-D equilibrium, the pressure is increased to $\beta \sim 3\,\%$ in a series of two steps. Then a 3-D perturbation is applied to the boundary, broken up into four steps (two shown) to arrive at the final finite-beta 3-D solution.

and after (not shown) is largely similar, with a small amount of additional current required to support the increased pressure.

While in general the two parameters could be varied in any order, or simultaneously when solving for a 3-D finite-beta equilibrium, we have found that first varying η_p followed by η_b to be more efficient. Varying η_p while holding η_b fixed at 0 allows the pressure perturbations to be done on a 2-D axisymmetric configuration, which reduces the computational cost and ensures the existence of good flux surfaces. After reaching a high-resolution finite-beta axisymmetric equilibrium, 3-D modes are added to the

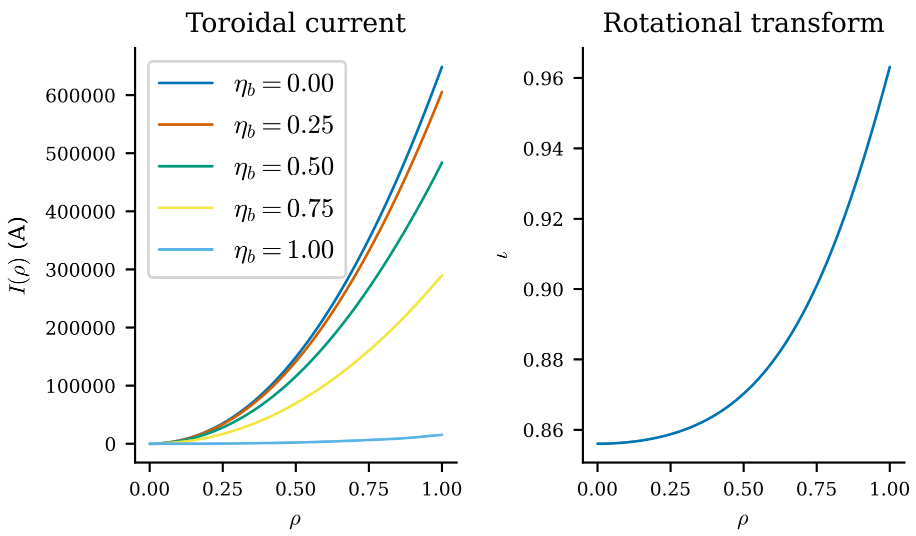

FIGURE 4. Toroidal current and rotational transform for W7-X-like equilibrium with varying boundary ratios. When $\eta_b = 0$, corresponding to an axisymmetric solution, a large toroidal current is required to generate the rotational transform (which is held fixed as the boundary is varied). Increasing the 3-D shaping allows more of the rotational transform to come from external fields, reducing the plasma current. Note that the current in the final step is still non-zero, as the rotational transform being held fixed was chosen for illustration, and does not correspond to the vacuum rotational transform on W7-X.

basis functions and the boundary is perturbed to give the final desired 3-D finite-beta equilibrium.

This procedure is demonstrated in figure 3, where the initial solution is an axisymmetric zero-pressure tokamak. The pressure is then increased, as demonstrated by the Shafranov shift, followed by 3-D deformation of the boundary shape to arrive at a W7-X-like configuration. After each large perturbation, a small number of regular Newton iterations of (2.1) are performed to ensure that an equilibrium has been reached (recall that in the derivation of the perturbations, it was assumed that the initial state before perturbing was an equilibrium). This method was also used to compute the solutions shown in Part 1 (Panici *et al.* 2022), where a detailed analysis of the force error is given.

Further insight can be gained by looking at the toroidal current profile of the W7-X-like equilibrium as the boundary ratio is varied in figure 4. At each step we keep the rotational transform profile fixed, and so as expected the axisymmetric case ($\eta_b = 0$) requires a large toroidal current to generate the poloidal field. As we increase the 3-D shaping, more of the rotational transform is generated by axis torsion and boundary shaping, reducing the required plasma current to near zero.

An important feature to note about the aforementioned continuation method is that the solution at each step (after any necessary Newton iterations of (2.1) to refine the solution) is in fact the ‘exact’ solution to the equilibrium problem with perturbed parameters, and is hence still a valid equilibrium that satisfies MHD force balance and may have desirable physics properties worth studying that may be absent in the final solution. By applying different perturbations and varying different parameters, one can explore whole families of solutions that are ‘nearby’ to a starting equilibrium.

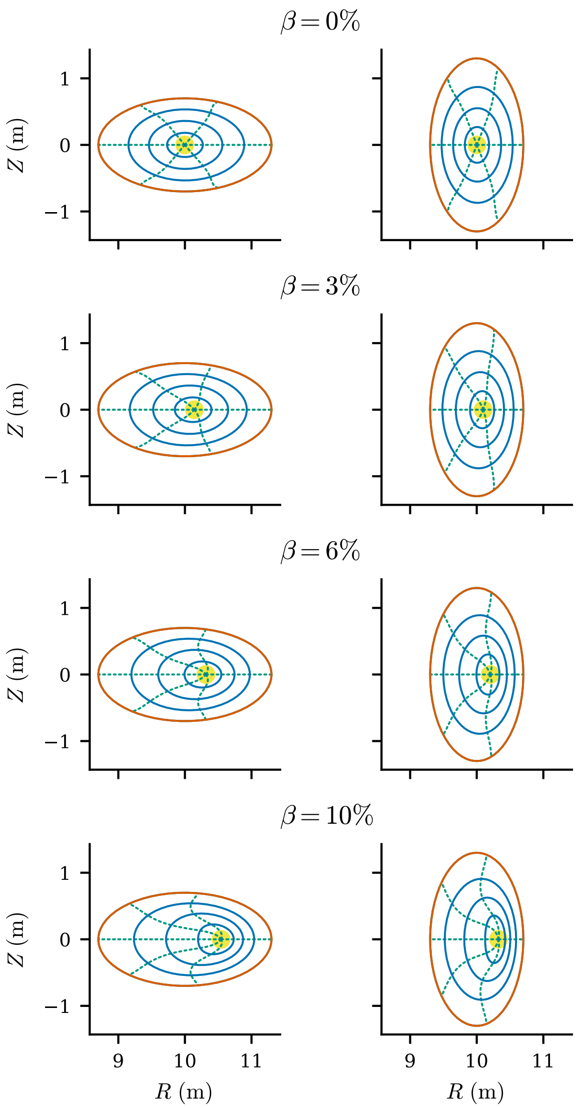

FIGURE 5. Flux surfaces on the $\zeta = 0$ and $\zeta = \pi$ planes for a helical stellarator. The initial equilibrium had $\beta = 0\,\%$, and a series of second-order perturbations were applied to increase the pressure rather than solving a new equilibrium from scratch each time.

5. Other applications

5.1. *Parameter scans*

Another important feature of continuation methods is revealing how solutions change as parameters of the problem are varied. In the previous section we used this to find single equilibria, but the method can also be used to explore families of equilibria related by a parameter or group of parameters. In this, we find equilibria 'nearby' to an equilibrium already found, such as examining the same boundary shape at different values of β or boundary perturbations of varying magnitude and shape such as resonant magnetic perturbations in a tokamak. Traditionally this requires re-solving the equilibrium for each new value of the parameter. An alternative approach is to apply a perturbation to an

β	Time (s) without perturbations	Time (s) with perturbations
1	164	78.0
2	174	16.3
3	164	14.4
4	122	12.6
5	127	17.8
6	122	19.1
7	121	15.2
8	134	13.6
9	140	16.0
10	138	18.3
Total	23.4 min	3.68 min

TABLE 3. Computation times for different values of β in the pressure scan shown in figure 5, both with and without perturbations, starting from a zero-pressure solution ($\beta = 0$, not shown). The first perturbation takes longer as the time includes just-in-time compilation for the specific problem being solved, resulting in significantly faster times for the subsequent perturbation steps which reuse the same compiled code. The 'without perturbations' method is already able to take advantage of compiled code from the initial solve and does not see any significant speed-up at subsequent iterations. All computation was done on an AMD Ryzen 7 PRO 4750 U with 32 GB memory.

initial equilibrium and find the corresponding changes in the flux surfaces. After applying the first perturbation, we re-linearize about the new state and perturb again, and so step through different equilibria for the cost of a single linear system solve at each step. This represents significant computational savings compared with solving the full nonlinear problem for each value of the parameter.

Figure 5 shows the flux surfaces on the $\zeta = 0$ and $\zeta = \pi$ planes for a helical stellarator for the same pressure profile scaled to different values of β. The initial solution was at $\beta = 0\,\%$, and second-order perturbations were applied sequentially to step the pressure up to $\beta = 10\,\%$. As in previous examples, the rotational transform is held fixed as the pressure is increased, leading to an increase in the toroidal current. The flux surfaces are indistinguishable from those obtained by solving from scratch at each value of β, and the computational cost is significantly reduced, as shown in table 3. Performing such a scan requires only a few lines of code, as shown in Listing 1.

```python
import numpy as np
import desc.io
from desc.equilibrium import EquilibriaFamily
eq0 = desc.io.load("heliotron_vacuum_solution.h5")[-1]
eqf = EquilibriaFamily(eq0)
# this corresponds to a change in beta of ~1
dp = np.array([ 1800., 0., -3600., 0., 1800.])
for i in range(10):
 eqf.append(eqf[-1].perturb(dp=dp, order=1))
 # polish off solution
 eqf[-1].solve(maxiter=5)
```

Listing 1 Code example for β scan.

As another example (figure 6), a D-shaped tokamak was used as the starting point for variations in the rotational transform profile. The initial equilibrium had a rotational

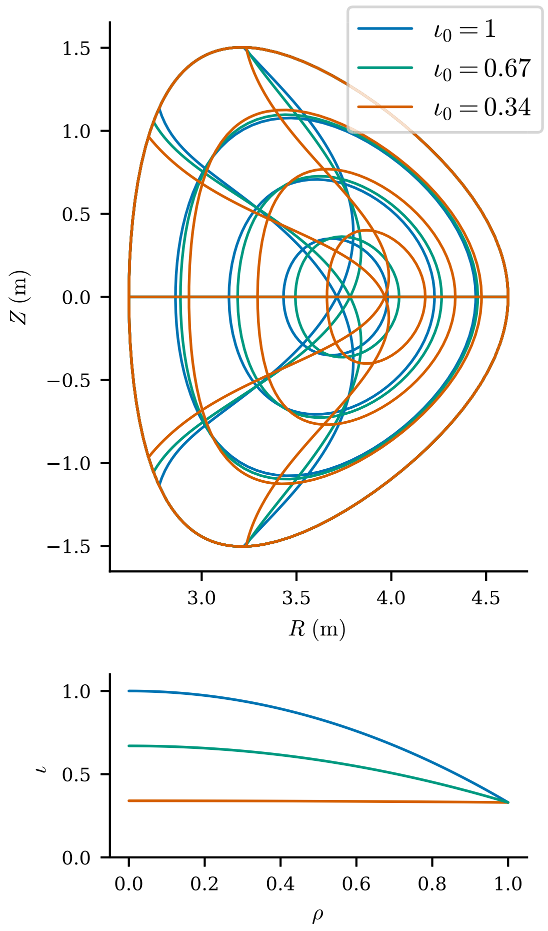

FIGURE 6. Top: Flux surfaces on the $\zeta = 0$ plane for a D-shaped tokamak with varying axis rotational transform. The initial equilibrium (blue) corresponds to $\iota_0 = 1$, and perturbations were used to reduce the axis rotational transform without recomputing the entire equilibrium solution. Bottom: Rotational transform profiles corresponding to flux surfaces shown above.

transform on axis of $\iota_0 = 1$, and perturbations were applied to reduce this down to $\iota_0 = 0.34$. As in the previous examples, the prescribed profiles at each step are pressure and rotational transform. As the rotational transform is decreased, we see increased Shafranov shift and deformation of the flux surfaces as the reduced poloidal field struggles to contain the pressure ($\beta \sim 3\,\%$ in this example).

As in the pressure scan example, the solution obtained by perturbation is indistinguishable by eye from the solution obtained by solving the full equilibrium problem, and significantly faster. Once an initial equilibrium is solved, parameter scans can be performed an order of magnitude faster using perturbations than other approaches which require a full solution at each step. The code to perform such a scan is shown in Listing 2.

```
import numpy as np
import desc.io
eq0 = desc.io.load("DSHAPE.h5")[-1]
# perturbing a 2nd order polynomial basis for iota
di = np.array([-0.33, 0, 0.33])
```

```
eq1 = eq0.perturb(di=di, order=2)
eq1.solve(maxiter=5)
eq2 = eq1.perturb(di=di, order=2)
eq2.solve(maxiter=5)
```

Listing 2 Code example for perturbing ι profile.

5.2. *Optimization*

The perturbation techniques described previously can also be adapted for optimization, where instead of choosing a particular change in the parameters $\Delta \boldsymbol{c}$ we instead let $\Delta \boldsymbol{c}$ be a free parameter that is chosen to minimize a cost function $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{c})$, such as quasi-symmetry error (Boozer 1983; Helander 2014), coil complexity (Zhu *et al.* 2018*b*, 2018*a*; McGreivy, Hudson & Zhu 2020) or fast particle confinement (Nemov *et al.* 2008; Velasco *et al.* 2021):

$$\Delta \boldsymbol{c}^* = \arg\min_{\Delta \boldsymbol{c}} \boldsymbol{g}(\boldsymbol{x} + \Delta \boldsymbol{x}, \boldsymbol{c} + \Delta \boldsymbol{c}) \quad \text{s.t.}\ \boldsymbol{f}(\boldsymbol{x} + \Delta \boldsymbol{x}, \boldsymbol{c} + \Delta \boldsymbol{c}) = 0, \tag{5.1}$$

where as before $\Delta \boldsymbol{x}$ is an implicit function of $\Delta \boldsymbol{c}$.

This finds the step in parameter space that most decreases the cost function $\boldsymbol{g}$ while maintaining approximate force balance. After applying this perturbation step, a small number of Newton iterations of (2.1) are used to reconverge to the correct equilibrium, without the need for a full 'cold start' equilibrium solve. This single 'warm start' equilibrium solve in DESC can be contrasted with the method of STELLOPT (Lazerson *et al.* 2020) or SIMSOPT (Landreman *et al.* 2021) where at each optimization step a series of N cold start equilibrium solves must be performed to find the descent direction, where N is the number of variables being optimized.

In the optimization literature, this is part of a more general class of methods for constrained optimization. Like the perturbations described in § 3, this method can also be extended to higher order. This extension and further details of this method of optimization and applications to quasi-symmetry are given in Part 3 (Dudt *et al.* 2022).

6. Conclusions

We have demonstrated a new technique for computing stellarator equilibria, and for exploring how those equilibria change as parameters are varied. The methods are computationally efficient, and offer significant speed-ups compared to existing techniques, and in many cases offer possibilities that have not existed before. An important future application of these methods is in exploring the connections between different classes of equilibria, such as how tokamaks bifurcate into stellarators, and how different classes of quasi-symmetric stellarators may be related.

Acknowledgements

Editor Per Helander thanks the referees for their advice in evaluating this article.

Funding

This work was supported by the US Department of Energy under contract numbers DE-AC02-09CH11466 and DE- SC0022005 and Field Work Proposal No. 1019. The United States Government retains a non-exclusive, paid-up, irrevocable, world-wide licence to publish or reproduce the published form of this paper, or allow others to do so, for United States Government purposes.

Declaration of interests

The authors report no conflict of interest.

Data availability statement

The source codes to generate the results and plots in this study are openly available in DESC at https://github.com/PlasmaControl/DESC or http://doi.org/10.5281/zenodo.4876504.

REFERENCES

ALLGOWER, E.L. & GEORG, K. 1990 *Numerical Continuation Methods: An Introduction*. Springer.

BERNSTEIN, I.B., FRIEMAN, E.A., KRUSKAL, M.D. & KULSRUD, R.M. 1958 An energy principle for hydromagnetic stability problems. *Proc. R. Soc. Lond.* A **244** (1236), 17–40, citation Key: Bernstein1958.

BOOZER, A.H. 1983 Transport and isomorphic equilibria. *Phys. Fluids* **26** (2), 496–499.

BOOZER, A.H. 2015 Stellarator design. *J. Plasma Phys*. **81** (6), 515810606.

BRADBURY, J., FROSTIG, R., HAWKINS, P., JOHNSON, M.J., LEARY, C., MACLAURIN, D., NECULA, G., PASZKE, A., VANDERPLAS, J., WANDERMAN-MILNE, S., *et al.* 2018 JAX: composable transformations of Python+NumPy programs.

DAUPHIN, Y., PASCANU, R., GULCEHRE, C., CHO, K., GANGULI, S. & BENGIO, Y. 2014 Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Adv. Neural Inf. Process. Syst.* **27**.

DREVLAK, M., BEIDLER, C.D., GEIGER, J., HELANDER, P. & TURKIN, Y. 2019 Optimisation of stellarator equilibria with rose. *Nucl. Fusion* **59** (1), 016010.

DUDT, D., CONLIN, R., PANICI, D. & KOLEMEN, E. 2022 The desc stellarator code suite part iii: quasi-symmetry optimization.

DUDT, D.W. & KOLEMEN, E. 2020 Desc: a stellarator equilibrium solver. *Phys. Plasmas* **27** (10), 102513.

GANDER, W. 1985 On Halley's iteration method. *Am. Math. Mon.* **92** (2), 131–134.

GLASSER, A.H. 2016 The direct criterion of newcomb for the ideal MHD stability of an axisymmetric toroidal plasma. *Phys. Plasmas* **23** (7), 072505, citation Key: Glasser2016.

GLASSER, A., KOLEMEN, E. & GLASSER, A.H. 2018 A Riccati solution for the ideal MHD plasma response with applications to real-time stability control. *Phys. Plasmas* **25** (3), 032507, citation Key: Glasser2018.

GUNDERSEN, G. & STEIHAUG, T. 2010 On large-scale unconstrained optimization problems and higher order methods. *Optim. Meth. Softw*. **25** (3), 337–358.

HARRIS, C.R., MILLMAN, K.J., VAN DER WALT, S.J., GOMMERS, R., VIRTANEN, P., COURNAPEAU, D., WIESER, E., TAYLOR, J., BERG, S., SMITH, N.J., *et al.* 2020 Array programming with numpy. *Nature* **585** (7825), 357–362.

HELANDER, P. 2014 Theory of plasma confinement in non-axisymmetric magnetic fields. *Rep. Prog. Phys.* **77** (8), 087001.

HELANDER, P., BEIDLER, C.D., BIRD, T.M., DREVLAK, M., FENG, Y., HATZKY, R., JENKO, F., KLEIBER, R., PROLL, J.H.E., TURKIN, Y., *et al.* 2012 Stellarator and tokamak plasmas: a comparison. *Plasma Phys. Control. Fusion* **54** (12), 1–33.

HIRSHMAN, S.P. & WHITSON, J.C. 1983*a* Steepest-descent moment method for three-dimensional magnetohydrodynamic equilibria. *Phys. Fluids* **26** (12), 3553–3568.

HIRSHMAN, S.P. & WHITSON, J.C. 1983*b* Steepest-descent moment method for three-dimensional magnetohydrodynamic equilibria. *Phys. Fluids* **26** (12), 3553–3568.

HOWELL, J.S. 2009 Computation of viscoelastic fluid flows using continuation methods. *J. Comput. Appl. Maths* **225** (1), 187–201.

HUDSON, S.R., DEWAR, R.L., HOLE, M.J. & MCGANN, M. 2012 Non-axisymmetric, multi-region relaxed magnetohydrodynamic equilibrium solutions. *Plasma Phys. Control. Fusion* **54** (1), 014005.

LANDREMAN, M., MEDASANI, B., WECHSUNG, F., GIULIANI, A., JORGE, R. & ZHU, C. 2021 Simsopt: a flexible framework for stellarator optimization. *J. Open Sour. Softw*. **6** (65), 3525.

LAZERSON, S., SCHMITT, J., ZHU, C., BRESLAU, J. ALL STELLOPT DEVELOPERS & USDOE OFFICE OF SCIENCE 2020 Stellopt, version 2.7.5.

MCGREIVY, N., HUDSON, S.R. & ZHU, C. 2020 Optimized finite-build stellarator coils using automatic differentiation. *Nucl. Fusion* **61** (2), 026020.

NEMOV, V.V., KASILOV, S.V., KERNBICHLER, W. & LEITOLD, G.O. 2008 Poloidal motion of trapped particle orbits in real-space coordinates. *Phys. Plasmas* **15** (5), 052501.

NOCEDAL, J. & WRIGHT, S. 2006 *Numerical Optimization*. Springer Science & Business Media.

PANICI, D., CONLIN, R., DUDT, D.W. & KOLEMEN, E. 2022 The desc stellarator code suite part i: quick and accurate equilibria computations.

PARK, J.-K., BOOZER, A.H., MENARD, J.E., GAROFALO, A.M., SCHAFFER, M.J., KAYE, S.M., GERHARDT, S.P. & SABBAGH, S.A. 2009 Importance of plasma response to nonaxisymmetric perturbations in tokamaks. *Phys. Plasmas* **16** (5), 056115.

PARK, J.K., BOOZER, A.H. & GLASSER, A.H. 2007 Computation of three-dimensional tokamak and spherical torus equilibria. *Phys. Plasmas* **14** (5), 0–9.

RICHTER, S.L. & DECARLO, R.A. 1983 Continuation methods: theory and applications. *IEEE Trans. Syst. Man Cybern.* **4**, 459–464.

SPONG, D.A., HIRSHMAN, S.P., WHITSON, J.C., BATCHELOR, D.B., CARRERAS, B.A., LYNCH, V.E. & ROME, J.A. 1998 J* optimization of small aspect ratio stellarator/tokamak hybrid devices. *Phys. Plasmas* **5** (5), 1752–1758.

VAN DER WALT, S., COLBERT, S.C. & VAROQUAUX, G. 2011 The numpy array: a structure for efficient numerical computation. *Comput. Sci. Engng* **13** (2), 22–30.

VELASCO, J.L., CALVO, I., MULAS, S., SANCHEZ, E., PARRA, F.I., CAPPA, A. & THE W7-X TEAM 2021 A model for the fast evaluation of prompt losses of energetic ions in stellarators. *Nucl. Fusion* **61** (11), 116059.

ZHU, C., HUDSON, S.R., LAZERSON, S.A., SONG, Y. & WAN, Y. 2018*a* Hessian matrix approach for determining error field sensitivity to coil deviations. *Plasma Phys. Control. Fusion* **60** (5), 054016.

ZHU, C., HUDSON, S.R., SONG, Y. & WAN, Y. 2018*b* New method to design stellarator coils without the winding surface. *Nucl. Fusion* **58**, 016008.