J. Plasma Phys. (2024), *vol*. 90, 905900501
Published by Cambridge University Press

doi:10.1017/S0022377824000655

Stellarator optimization with constraints

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

[1]Princeton University, Princeton, NJ 08544

(Received 16 March 2024; revised 6 May 2024; accepted 7 May 2024)

In this work we consider the problem of optimizing a stellarator subject to hard constraints on the design variables and physics properties of the equilibrium. We survey current numerical methods for handling these constraints, and summarize a number of methods from the wider optimization community that have not been used extensively for stellarator optimization thus far. We demonstrate the utility of new methods of constrained optimization by optimizing a quasi-axisymmetric stellarator for favourable physics properties while preventing strong shaping of the plasma boundary, which can be difficult to create with external current sources.

Key words: fusion plasma, plasma confinement, plasma simulation

1. Introduction

Stellarator optimization is naturally full of constraints that must be satisfied by an optimized configuration. Many of these constraints are due to engineering requirements, such as reasonable aspect ratio, minimum coil-plasma distance, or maximum curvature of the coils or plasma boundary. Others come from physics requirements, such as having a prescribed rotational transform at the boundary for an island divertor (Feng *et al.* 2006), having a stable magnetic well (Landreman & Jorge 2020) or ensuring the net current is zero. There are also constraints to enforce self-consistency between different models of the plasma, such as requiring that the prescribed current in an ideal magnetohydrodynamic (MHD) equilibrium solver matches with the actual bootstrap current from a kinetic calculation (Landreman, Buller & Drevlak 2022), or that the coils correctly cancel the normal field on the plasma boundary. Perhaps the most important constraint is that any optimized configuration must actually be in MHD equilibrium.

A generic optimization problem can be written as

$$\left.\begin{array}{ll} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}_E(\boldsymbol{x}) = 0 \\ & \boldsymbol{l} \leq \boldsymbol{g}_I(\boldsymbol{x}) \leq \boldsymbol{u} \end{array}\right\}, \tag{1.1}$$

where $\boldsymbol{x} \in \mathcal{R}^n$ is a vector of decision variables, $f : \mathcal{R}^n \to \mathcal{R}^1$ is the objective function, $\boldsymbol{g}_E : \mathcal{R}^n \to \mathcal{R}^{m_1}$ are equality constraints, $\boldsymbol{g}_I : \mathcal{R}^n \to \mathcal{R}^{m_2}$ are inequality constraints and $\boldsymbol{l}, \boldsymbol{u} \in$

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

$\mathcal{R}^{m_2}$ are lower and upper bounds on the inequality constraints (which may be infinite for one-sided constraints). By introducing slack variables $\boldsymbol{s}$, we can transform the inequality constraint into an equality constraint and simple bounds, i.e.

$$\left.\begin{array}{ll}\min\limits_{\boldsymbol{x},\boldsymbol{s}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}_E(\boldsymbol{x}) = 0 \\ & \boldsymbol{g}_I(\boldsymbol{x}) - \boldsymbol{s} = 0 \\ & \boldsymbol{l} \leq \boldsymbol{s} \leq \boldsymbol{u}\end{array}\right\}, \tag{1.2}$$

which if we absorb $\boldsymbol{s}$ into $\boldsymbol{x}$ and combine the constraints into a single function $\boldsymbol{g}$, we can write in standard form as

$$\left.\begin{array}{ll}\min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) = 0 \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\end{array}\right\}. \tag{1.3}$$

In § 2 we summarize methods for handling constraints that are already in use in stellarator optimization such as linear constraint projection and sum of squares. In § 3 we introduce several methods from the wider optimization literature and discuss their properties and relevance for the stellarator optimization field. In § 4 we detail the implementation of a new augmented Lagrangian optimization algorithm into the DESC code and demonstrate its use with an example problem in § 5.

2. Existing methods

2.1. *Linear equality constraints*

Constraints where the decision variables $\boldsymbol{x}$ appear only linearly can be handled efficiently by suitably redefining variables. A problem with linear constraints can be written as

$$\left.\begin{array}{ll}\min\limits_{\boldsymbol{x}\in R^n} & f(\boldsymbol{x}) \\ \text{s.t.} & A\boldsymbol{x} - \boldsymbol{b} = 0\end{array}\right\}, \tag{2.1}$$

where $A \in R^{m\times n}$ and $\boldsymbol{b} \in R^m$, $m \leq n$ are the coefficients that define the linear constraint. We can then decompose $\boldsymbol{x} = \boldsymbol{x}_p + Z\boldsymbol{y}$ where $\boldsymbol{x}_p \in R^n$ is any particular solution to the underdetermined system $A\boldsymbol{x} = \boldsymbol{b}$ (typically taken to be the least-norm solution), $Z \in R^{n\times(n-m)}$ is a (typically orthogonal) representation for the null space of A, such that $AZ = 0$, and $\boldsymbol{y} \in R^{n-m}$ is an arbitrary vector. This allows us to write the problem as

$$\min_{\boldsymbol{y}\in R^{n-m}} \quad f(\boldsymbol{x}_p + Z\boldsymbol{y}), \tag{2.2}$$

which is now an unconstrained problem, where any value for $\boldsymbol{y}$ is feasible, and the full solution $\boldsymbol{x}^*$ can be recovered as $\boldsymbol{x}_p + Z\boldsymbol{y}^*$.

While simple to deal with, linear equality constraints are very limited in their applicability. Most constraints that are of interest in stellarator optimization are nonlinear, making linear equality constraints only useful for enforcing simple boundary conditions on the plasma boundary and profiles.

2.2. *Sum of squares*

Perhaps the simplest way to (approximately) solve (1.3) is with a quadratic penalty method, where the objective and constraints are combined into a single function, which is then

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

minimized without constraints:

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) + w_1 \|\boldsymbol{g}(\boldsymbol{x})\|^2 + w_2 \|I_{(\boldsymbol{l},\boldsymbol{u})}(\boldsymbol{x})\|^2. \tag{2.3}$$

Here w_1 and w_2 are weights and $I_{(\boldsymbol{l},\boldsymbol{u})}(\boldsymbol{x})$ is a 'deadzone' function:

$$I_{(\boldsymbol{l},\boldsymbol{u})}(\boldsymbol{x}) = \begin{Bmatrix} 0 & \text{if } \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \\ \max(\boldsymbol{x} - \boldsymbol{u}, \boldsymbol{l} - \boldsymbol{x}) & \text{otherwise} \end{Bmatrix}. \tag{2.4}$$

This formulation is used by a number of codes for optimizing stellarator equilibria, such as SIMSOPT, STELLOPT, ROSE and DESC. Similar techniques have also been used for coil design in FOCUS (Kruger *et al.* 2021). While simple to implement, this formulation has a number of drawbacks.

(i) The weights w must be determined by the user, often through extensive trial and error. This can be automated by iteratively increasing the weights until the constraint violation is reduced to an acceptable level, as in Bindel, Landreman & Padidar (2023), though this leads to the next problem.

(ii) Formally, the constraints are only satisfied as $w \to \infty$ and reducing constraint violation to an acceptable level can often require very large values of the weights, leading to a badly scaled problem that can slow convergence.

(iii) The max term in the deadzone function introduces a discontinuity in the second derivative of the objective, which can hinder progress of some optimization methods that assume the objective and constraints to be at least C2. This can be partially alleviated by using a quartic or exponential penalty instead of quadratic, as in Kruger *et al.* (2021), though this makes the problem more nonlinear and can reduce the efficiency of many optimizers that assume a locally linear or quadratic model function.

2.3. *Projection methods*

The constraint of MHD equilibrium is most commonly dealt with by using a fixed boundary equilibrium solver at each step in the optimization. Given the plasma boundary and profiles at the latest optimization step, the equilibrium solver is run to determine the full solution throughout the plasma volume, and this is then used to evaluate the objective function. This is akin to a projection method, where after each step of the optimizer, the solution is projected back onto the feasible region $\{\boldsymbol{x} \mid \boldsymbol{g}(\boldsymbol{x}) = 0\}$. This also suffers from a number of drawbacks.

(i) The equilibrium must be resolved at each step of the optimizer, possibly multiple times if finite differences are used to evaluate derivatives. This can be made somewhat more efficient if a warm start is used based on the previous accepted step (Conlin *et al.* 2023; Dudt *et al.* 2023), though this is still often the most expensive part of the optimization loop.

(ii) Projection type methods are limited in applicability to constraints for which a projection operator can be defined. The MHD equilibrium solvers can serve this function for equilibrium constraints, but a projection operator for a general nonlinear constraint is not known.

(iii) Enforcing that the constraints are satisfied at each step of the optimization may be too strict and cause the optimizer to stop at a poor local minimum.

This last point is illustrated in figure 1. Points where the constraint curve $g(\boldsymbol{x}) = 0$ is tangent to the contours of f are local minima of the constrained problem, where following

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

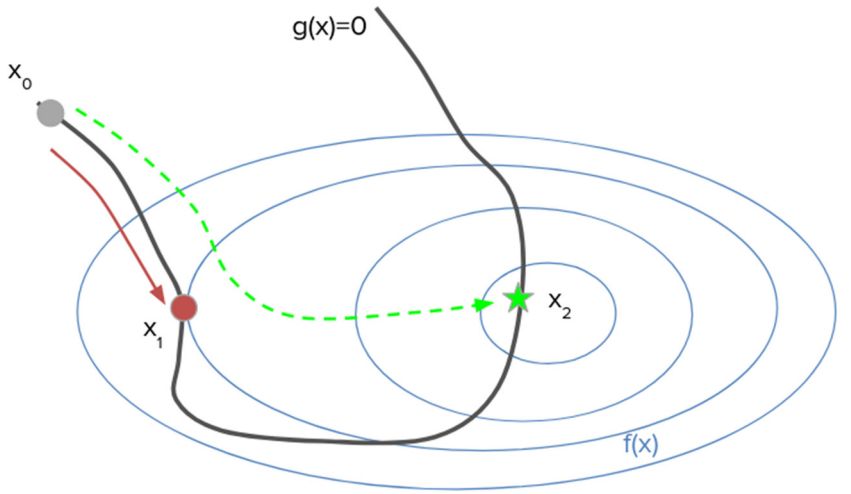

FIGURE 1. Sketch of optimization landscape (contours of f in blue) with constraints (curve of $g(x) = 0$ in black). Starting from x_0 and enforcing the constraints exactly at each step will follow the red path and end at x_1. If we allow ourselves to temporarily violate the constraints, we can follow the green path and arrive at the better solution x_2.

the constraint curve will lead to higher objective function values, but moving in a direction to decrease the objective function will violate the constraints. If we enforce that the constraint is satisfied at each step, we will follow the path in red and stop at the local minimum x_1. However, if we are allowed to temporarily violate the constraints, we can make rapid progress towards the better local minimum x_2.

Some may worry that not enforcing MHD equilibrium at each step may lead the optimizer in directions that are not dynamically accessible during an actual plasma discharge, where the plasma is assumed to be in equilibrium; however, it is important to note that the path taken through phase space during optimization in general has no connection to the path taken during an actual discharge, and just because the optimizer finds a path that violates equilibrium does not mean there is not a nearby path that satisfies it. Dynamical accessibility is an important aspect of stellarator optimization, but the expense of time-dependent simulations preclude their use inside an optimization loop, so are generally limited to post-optimization validation and analysis.

3. Methods for constrained optimization

We consider a general constrained optimization problem of the form

$$\left.\begin{array}{ll} \min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}_E(\boldsymbol{x}) = 0 \\ & \boldsymbol{g}_I(\boldsymbol{x}) \geq 0 \end{array}\right\}, \tag{3.1}$$

where, as before, $\boldsymbol{g}_E$ and $\boldsymbol{g}_I$ denote the equality and inequality constraints, respectively. Most analysis and algorithms start from the Lagrangian for this problem, which is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}_E, \boldsymbol{y}_I) = f(\boldsymbol{x}) - \boldsymbol{y}_E^T \boldsymbol{g}_E(\boldsymbol{x}) - \boldsymbol{y}_I^T \boldsymbol{g}_I(\boldsymbol{x}), \tag{3.2}$$

where $\boldsymbol{y}_E$ and $\boldsymbol{y}_I$ are the Lagrange multipliers associated with the equality and inequality constraints, respectively. The conditions for a given set $(\boldsymbol{x}^*, \boldsymbol{y}_E^*, \boldsymbol{y}_I^*)$ to be a solution to (3.1)

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

are the so called Karush–Kuhn–Tucker (KKT) conditions (Nocedal & Wright 2006):

$$\nabla_x \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{y}_E^*, \boldsymbol{y}_I^*) = \nabla f - \boldsymbol{y}_E^T \nabla \boldsymbol{g}_E - \boldsymbol{y}_I^T \nabla \boldsymbol{g}_I = 0, \tag{3.3a}$$

$$\boldsymbol{g}_E(\boldsymbol{x}^*) = 0, \tag{3.3b}$$

$$\boldsymbol{g}_I(\boldsymbol{x}^*) \geq 0, \tag{3.3c}$$

$$\boldsymbol{y}_I^* \geq 0, \tag{3.3d}$$

$$\boldsymbol{y}_E^* \cdot \boldsymbol{g}_E(\boldsymbol{x}^*) = 0, \tag{3.3e}$$

$$\boldsymbol{y}_I^* \cdot \boldsymbol{g}_I(\boldsymbol{x}^*) = 0. \tag{3.3f}$$

The first condition is reminiscent of the first-order optimality condition for an unconstrained problem, $\nabla f(\boldsymbol{x}^*) = 0$, with the objective function f replaced by the Lagrangian $\mathcal{L}$; however, it is important to note that we do not seek a minimum of the Lagrangian. It can be shown that solutions to (3.3) always lie at a saddle point where the Lagrangian is minimized with respect to the primal variables $\boldsymbol{x}$ and maximized with respect to the dual variables $\boldsymbol{y}$.

It is useful to take a moment to consider the meaning and utility of the Lagrange multipliers $\boldsymbol{y}$. A first insight into the meaning of Lagrange multipliers can be found from the last of the KKT conditions in (3.3*f*), i.e.

$$\boldsymbol{y}_I^* \cdot \boldsymbol{g}_I(\boldsymbol{x}^*) = 0, \tag{3.4}$$

meaning that either the constraint is zero or its associated multiplier is zero for each constraint. For a problem in the form of (3.1), an inequality constraint not equal to zero is said to be 'non-binding' or 'not active', meaning that the constraint has no effect on the solution, and the minimum would not change if the constraint were removed. This is contrasted with a 'binding' or 'active' constraint that is preventing the solution from improving the objective f. This suggests that the Lagrange multipliers contain information about how much the constraints are affecting the solution. This can be made more explicit by considering the following problem:

$$\left.\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.t.} & g(x) = 0 \end{array}\right\}. \tag{3.5}$$

The Lagrangian for this problem is

$$\mathcal{L}(x, y) = f(x) - y g(x). \tag{3.6}$$

We can now consider what happens if we perturb the constraint $g = 0$ to $g + \delta g = 0$.

From (3.3*a*) we see that at an optimum x^* the change in the Lagrangian must be zero to first order, which gives

$$\delta f = y \delta g. \tag{3.7}$$

Here we see that the Lagrange multipliers tell us how much our objective can improve by relaxing our constraints, or the so called 'marginal cost' of enforcing the constraint $g = 0$. Additionally, the sign of the multipliers tells us which direction we should relax the constraints to improve the objective.

We can use a similar approach to consider trade offs between conflicting constraints, where if we have constraints g_1 and g_2 that cannot both be satisfied, the ratio of their multipliers y_1/y_2 tells us how much we can expect to improve g_1 by relaxing g_2 and *vice versa*.

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

A number of methods have been developed for solving constrained optimization problems, which we briefly summarize in the following subsections. More information can be found in standard texts on numerical optimization (Conn, Gould & Philippe 2000; Nocedal & Wright 2006)

3.1. *Interior point methods*

A popular class of methods for dealing with inequality constraints are the so-called 'interior point' methods, or sometimes referred to as 'log barrier' methods. These methods solve problems of the form

$$\left.\begin{array}{ll}\min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) = 0 \\ & 0 \leq \boldsymbol{x}\end{array}\right\}, \tag{3.8}$$

(any problem of the form of (1.3) can be transformed into this form by appropriate change of variables). The bound on the variables is replaced by a logarithmic penalty term:

$$\left.\begin{array}{ll}\min\limits_{\boldsymbol{x}} & f(\boldsymbol{x}) - \mu \sum\limits_i \log x_i \\ \text{s.t.} & \boldsymbol{g}(\boldsymbol{x}) = 0\end{array}\right\}. \tag{3.9}$$

This problem is then solved repeatedly with an equality constrained optimization method such as sequential quadratic programming (SQP, see the following section) for a decreasing sequence of barrier parameters μ. For finite values of μ, the log term pushes the solution into the interior of the feasible set (hence the name), and the solution is allowed to approach the boundary as the barrier gets sharper as μ approaches zero.

The interior point method is the basis of a number of popular commercial and open-source optimization codes (Wächter & Biegler 2006; Biegler & Zavala 2009), but we have found that in practice they do not perform well on the types of problems commonly encountered in stellarator optimization, for reasons that will be discussed further in § 4

3.2. *Sequential quadratic programming methods*

Another popular class of methods are based on sequentially minimizing an approximate model of the objective and constraints. Typically, the objective is modelled as quadratic and the constraints are modelled as linear, leading to a standard quadratic program. If the current estimate of the solution is $\boldsymbol{x}_k$, we seek the next iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}$ by solving the following subproblem:

$$\left.\begin{array}{rl}\min\limits_{\boldsymbol{p}} & \boldsymbol{p}^T H \boldsymbol{p} + \boldsymbol{g}^T \boldsymbol{p} \\ \text{s.t.} & A_E \boldsymbol{p} = -\boldsymbol{g}_E(\boldsymbol{x}_k) \\ \boldsymbol{l} - \boldsymbol{g}_I(\boldsymbol{x}_k) \leq & A_I \boldsymbol{p} \leq \boldsymbol{u} - \boldsymbol{g}_I(\boldsymbol{x}_k)\end{array}\right\}. \tag{3.10}$$

Here $\boldsymbol{g}$ and H are the gradient and Hessian of f and A_E, and A_I are the Jacobian matrices of the equality and inequality constraints $\boldsymbol{g}_E$ and $\boldsymbol{g}_I$.

One common issue with SQP methods is the possible inconsistency of the linearized constraints, such that the subproblem (3.10) has no solution. This problem can be made worse by the inclusion of a trust region constraint, as is often used to ensure global convergence. To overcome this, the full step $\boldsymbol{p}$ is typically broken into two sub-steps, one of which attempts to achieve feasibility with respect to the linearized constraints as much as possible, while the other attempts to reduce the objective function. A merit function is then

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

used to balance between improving the objective and decreasing the constraint violation (Conn *et al.* 2000).

Similar to interior point methods, SQP methods form the core of a number of commercial and open-source optimization codes (Gill, Murray & Saunders 2005; Johnson & Schueller 2021). However, we similarly see that the performance of existing implementations is limited on the types of problems encountered in practice.

3.3. *Active set methods*

Active set methods makes use of the fact that if one knows ahead of time which inequality constraints are active at the solution, then we can turn those into equality constraints and ignore all other inequality constraints, leading to a purely equality constrained problem that is often easier to solve. Active set methods make a guess for which constraints are active at the solution and iteratively update this set based on the results of sequential equality constrained optimizations, commonly using SQP methods.

These methods can be very efficient for small problem sizes, however, because the guess of the active constraints is updated at each iteration, the required number of steps grows with the number of constraints (often linearly, but can be exponential in the worst case), making them far less efficient for problems of moderate to large size.

3.4. *Augmented Lagrangian methods*

For the quadratic penalty method considered in (2.3), one can show that the constraint violation is inversely proportional to the penalty parameter

$$g_i(\boldsymbol{x}_k) \approx -y_i^*/\mu_k, \tag{3.11}$$

where y_i^* is the true Lagrange multiplier (from the solution to (3.3)) associated with the constraint g_i and μ_k is the penalty parameter. We see that if we want to satisfy the constraints exactly, we must have $\mu \to \infty$, which as discussed can be numerically intractable. However, if we consider the 'augmented' Lagrangian

$$\mathcal{L}_A(\boldsymbol{x}, \boldsymbol{y}, \mu) = f(\boldsymbol{x}) - \boldsymbol{y}^T \boldsymbol{g}(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{g}(\boldsymbol{x})\|^2, \tag{3.12}$$

where $\boldsymbol{y}$ are estimates of the true Lagrange multipliers, one can show that now (Nocedal & Wright 2006)

$$g_i(\boldsymbol{x}_k) \approx -(y_i^* - y_i^k)/\mu_k. \tag{3.13}$$

So that if our estimate of the true Lagrange multipliers is accurate, we can significantly reduce the constraint violation without requiring the penalty parameter to grow without bound. This also gives us a rule to update our estimate of the multipliers:

$$y_i^{k+1} = y_i^k - \mu_k g_i(\boldsymbol{x}_k). \tag{3.14}$$

Minimizing the augmented Lagrangian with fixed $\boldsymbol{y}$ and μ is an unconstrained or bound constrained problem, for which standard gradient or Newton-based methods can be used. From this solution we obtain an approximation to the true solution of the constrained problem. We can then update the multipliers and penalty term and re-optimize, eventually converging to the true solution. This is the basic method used in several widely used codes (Conn, Gould & Toint 2013; Fowkes *et al.* 2023) and can also be used as a generic technique to handle constraints for algorithms that otherwise would only be able to solve unconstrained problems (Johnson & Schueller 2021).

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

One can also construct an augmented Lagrangian for a least squares problem. Because $L(\boldsymbol{x}, \boldsymbol{y}, \mu)$ is only ever minimized with respect to $\boldsymbol{x}$, one can add any quantity independent of $\boldsymbol{x}$ without affecting the result. In particular, if we add $\|\boldsymbol{y}\|^2/2\mu$, we get

$$\mathcal{L}_A(\boldsymbol{x}, \boldsymbol{y}, \mu) = \tfrac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|^2 - \boldsymbol{y}^T\boldsymbol{g}(\boldsymbol{x}) + \mu/2\|\boldsymbol{g}(\boldsymbol{x})\|^2 + \|\boldsymbol{y}\|^2/2\mu. \tag{3.15}$$

We can then complete the square to obtain

$$\mathcal{L}_A(\boldsymbol{x}, \boldsymbol{y}, \mu) = \tfrac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|^2 + \tfrac{1}{2}\| - \boldsymbol{y}/\sqrt{\mu} + \sqrt{\mu}\boldsymbol{g}(\boldsymbol{x})\|^2. \tag{3.16}$$

This allows us to use standard unconstrained or bound constrained least squares routines to solve the augmented Lagrangian subproblem, which has a number of benefits.

(i) Super-linear convergence can be obtained using only first derivatives of $\boldsymbol{f}$ and $\boldsymbol{g}$ (provided the residual is sufficiently small at the solution, which in practice is very often the case) as opposed to standard Newton methods that require second derivatives of the objective and constraints.
(ii) The quadratic subproblem that arises in least squares is always convex, for which the exact solution can be found efficiently.

These benefits can also be obtained using other quasi-Newton methods such as BFGS, which also require only first derivatives and have convex subproblems, but in practice we have found that in the common case where the problem has a natural least squares structure, using a least squares method significantly outperforms BFGS in both time and number of iterations required to reach a solution.

4. Constrained optimization in DESC

As previously mentioned there are a wide variety of commercial and open-source codes for constrained optimization, many of which have been interfaced to the DESC stellarator optimization code (Dudt & Kolemen 2020; Conlin *et al.* 2023; Dudt *et al.* 2023, 2024; Panici *et al.* 2023). In practice, we have found that most of the existing methods perform somewhat poorly on the types of problems encountered in stellarator optimization. We hypothesize this is for a number of reasons.

(i) Many commercial solvers assume a particular structure for the problem, such as quadratic, conic and semi-definite programming solvers. Others rely on convexity, partial separability or other conditions that generally do not hold in stellarator optimization.
(ii) Many codes are designed and optimized for solving very-high-dimensional problems ($O(10^6)$ variables or more) but with a high degree of sparsity, and make extensive use of sparse and iterative linear algebra methods. In stellarator optimization problems we generally do not have significant sparsity due to the wide use of spectral methods, making these methods inefficient.
(iii) The types of problems in stellarator optimization are often poorly scaled, due to a wide range of scales in physical units, and due to spectral discretizations. This can lead to failure of iterative linear algebra subproblems without a proper preconditioner (which may not be easy to determine) and can also slow progress of the overall optimization because the optimization landscape is strongly anisotropic (meaning the objective is much more sensitive to changes in certain directions in parameter space than others, such as a long narrow valley).
(iv) No existing implementations that we are aware of take advantage of graphics processing unit (GPU) acceleration for the underlying linear algebra. This is partially

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

by design, as GPUs are often of little benefit for the types of large sparse problems existing codes are designed for. However, as previously mentioned, the problems in stellarator optimization tend to be moderately sized and dense, for which GPU linear algebra can provide a significant benefit.

Because of this, we have implemented a new optimizer based on the least squares augmented Lagrangian formalism from § 3.4, and a sketch of the algorithm is found in algorithm 1. Future work could include re-implementations of SQP or interior point methods to better handle dense and poorly scaled linear systems as these methods can be very powerful, though existing implementations do not seem to perform well.

Before applying the augmented Lagrangian method, we first transform any inequality constraints into equality constraints plus bounds using slack variables as in (1.3). The augmented Lagrangian is then used to handle the equality constraints, while the bound constraints are handled by using a Coleman–Li type trust region method (Coleman & Li 1996) where the trust region is adjusted at each iteration to account for the bounds, ensuring that iterates remain strictly feasible with respect to the bound constraints.

Algorithm 1 Augmented Lagrangian algorithm (adapted from Conn *et al*. 2000; Nocedal & Wright 2006)

Require: $\mu_0 > 0, \omega_* > 0, \eta_* > 0$
  $\mu_k \leftarrow \mu_0$
  $\omega_k \leftarrow 1/\mu_0$
  $\eta_k \leftarrow 1/\mu_0^{0.1}$
  $k \leftarrow 1$
  while $\|\nabla \mathcal{L}_A(\boldsymbol{x}_k, \boldsymbol{y}_k, \mu_k)\| > \omega_*$ or $\|\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{y}_k, \mu_k)\| > \eta_*$ **do**
    $\boldsymbol{x}_{k+1} \leftarrow \arg\min_{\boldsymbol{x}} \mathcal{L}_A(\boldsymbol{x}_k, \boldsymbol{y}_k, \mu_k)$ ▷ using bound constrained method, solved to tolerance ω_k
    if $\|\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{y}_k, \mu_k)\| < \eta_k$ **then** ▷ Constraints are making good progress, update multipliers
      $\boldsymbol{y}_{k+1} \leftarrow \boldsymbol{y}_k - \mu_k \boldsymbol{g}(\boldsymbol{x}_{k+1})$
      $\mu_{k+1} \leftarrow \mu_k$
      $\omega_{k+1} \leftarrow \omega_k/\mu_{k+1}$
      $\eta_{k+1} \leftarrow \eta_k/\mu_{k+1}$
    else ▷ Need to improve constraint satisfaction, increase penalty term
      $\boldsymbol{y}_{k+1} \leftarrow \boldsymbol{y}_k$
      $\mu_{k+1} \leftarrow 10\mu_k$
      $\omega_{k+1} \leftarrow 1/\mu_{k+1}$
      $\eta_{k+1} \leftarrow 1/\mu_{k+1}$
    end if
    $k \leftarrow k + 1$
  end while

The precise rules for updating the tolerances and penalty parameters can be varied, see (Conn *et al.* 2000) for a more complex version, though we have found that in practice the results and performance do not significantly depend on the choice of hyperparameters. These rules are effectively automatically finding the correct weights to manage the trade-off between improving the objective value and constraint feasibility that would normally require extensive user trial and error.

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

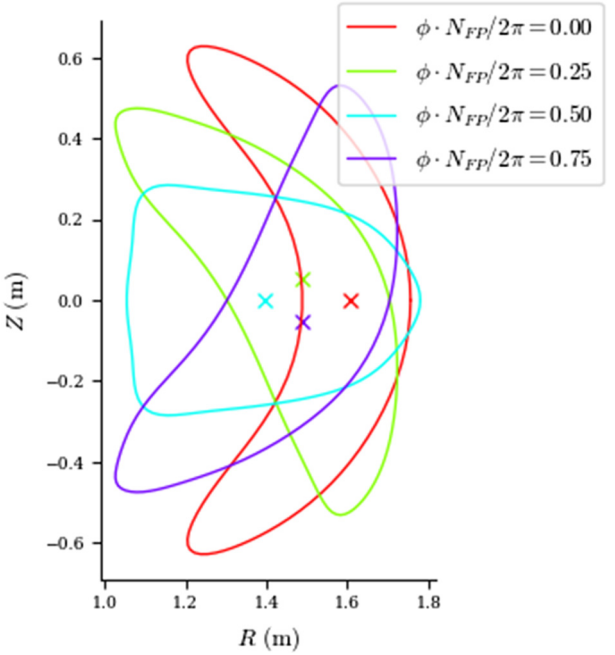

FIGURE 2. Plasma boundary of NCSX at several cross-sections, with the 'bean' section at $\phi = 0$.

5. Examples

Many optimized stellarators have very strongly shaped boundaries, and a 'bean' shaped cross-section is quite common, appearing in designs such as NCSX (shown in figure 2), W7-X, ARIES-CS and many recent designs such as the 'precise' quasi-symmetric equilibria of Landreman and Paul (Landreman & Paul 2022). These strongly indented shapes can be difficult to create with external coils or magnets, often requiring coils that are very close on the inboard concave side (Landreman & Boozer 2016; Kappel, Landreman & Malhotra 2024).

We can attempt to alleviate this problem with constrained optimization by properly constraining the curvature of the surface. We consider a surface parameterized by the poloidal angle θ and toroidal angle ζ, with parameterization $\boldsymbol{r}(\theta, \zeta)$. The covariant basis vectors are given by $\boldsymbol{r}_\alpha(\theta, \zeta) = \partial_\alpha \boldsymbol{r}(\theta, \zeta)$ for $\alpha \in (\theta, \zeta)$. From these we can construct the 'first fundamental form' as (Do Carmo 2016)

$$\mathcal{F} = \begin{pmatrix} E & F \\ F & G \end{pmatrix}, \tag{5.1}$$

where the coefficients are dot products of the first derivative of the position vector:

$$E = \boldsymbol{r}_\theta \cdot \boldsymbol{r}_\theta, \tag{5.2}$$

$$F = \boldsymbol{r}_\theta \cdot \boldsymbol{r}_\zeta, \tag{5.3}$$

$$G = \boldsymbol{r}_\zeta \cdot \boldsymbol{r}_\zeta. \tag{5.4}$$

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

Similarly, we can define the 'second fundamental form' as

$$\mathcal{S} = \begin{pmatrix} L & M \\ M & N \end{pmatrix}, \tag{5.5}$$

where the coefficients are given by the projection of the second derivatives of the position vector onto the surface normal:

$$L = \boldsymbol{r}_{\theta\theta} \cdot \boldsymbol{n}, \tag{5.6}$$

$$M = \boldsymbol{r}_{\theta\zeta} \cdot \boldsymbol{n}, \tag{5.7}$$

$$N = \boldsymbol{r}_{\zeta\zeta} \cdot \boldsymbol{n}. \tag{5.8}$$

We can then compute the principal curvatures κ_1 and κ_2 of the surface as the solutions to the generalized eigenvalue problem

$$\mathcal{S}\boldsymbol{u} = \kappa \mathcal{F}\boldsymbol{u}, \tag{5.9}$$

where the eigenvectors $\boldsymbol{u}$ are called the principal directions. The principal curvatures are the maximum and minimum of the inverse radii of curvature of all the curves that are tangent to the surface at a given point, with the sign convention that positive curvature means the surface curves towards the normal vector.

On the inboard side of a toroidal surface, we expect at least one of the principal curvatures to be positive because the surface curves towards the normal in the toroidal direction, while at least one should be negative on the outboard side. Indenting of the poloidal cross-section characteristic of the bean shape would manifest as both principal curvatures being positive on the inboard side and both negative on the outboard side.

To prevent strong indentation, we use the mean curvature, defined as $H = 1/2(\kappa_1 + \kappa_2)$, and enforce that the mean curvature is everywhere negative. On the outboard side, where both curvatures are already negative this should have minimal effect, while on the inboard it forces any indent to have a larger radius of curvature than the local major radius.

In addition to this constraint on the curvature of the plasma surface, we also include constraints to enforce MHD equilibrium, fix the major radius R_0, aspect ratio R_0/a, total enclosed volume V (which we fix equal to the volume of the initial guess, a circular torus) and include inequality constraints on the rotational transform profile ι. We also fix the profiles of pressure p and current j and the total enclosed flux Ψ to give an average field strength of 1 T. In summary, our constraints are

$$\left.\begin{array}{c} R_0 = 1 \text{ m} \\ R_0/a = 6 \\ V = V_{init} \sim 0.55 \text{ m}^3 \\ 0.43 < \iota(\rho) < 0.5 \\ H(\theta, \zeta) < 0 \text{ m}^{-1} \\ p(\rho) = 0 \text{ Pa} \\ j(\rho) = 0 \text{ A} \\ \Psi = 0.087 \text{ Tm}^2 \\ \boldsymbol{J} \times \boldsymbol{B} - \nabla p = 0 \end{array}\right\}. \tag{5.10}$$

As our objective, we use the 'two-term' metric f_C for quasi-symmetry (Isaev, Mikhailov & Shafranov 1994; Rodriguez, Paul & Bhattacharjee 2022; Dudt *et al.* 2023),

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

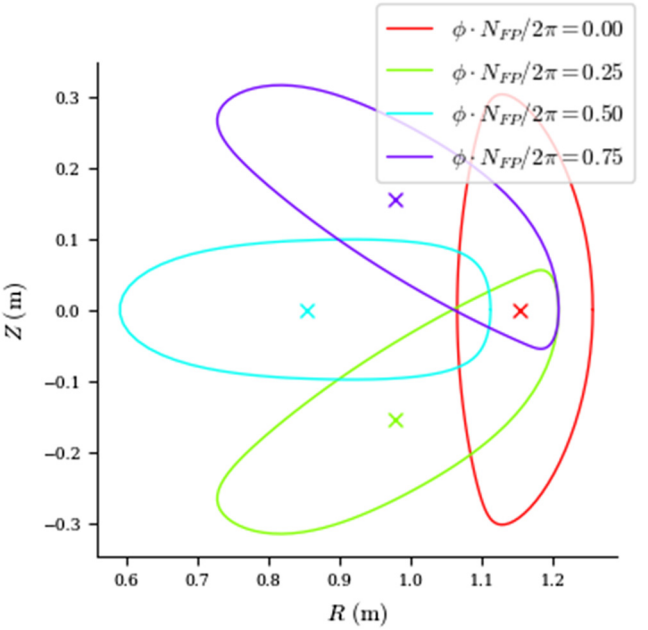

FIGURE 3. Plasma boundary of new optimized configuration with constraint on surface curvature.

targeting quasi-axisymmetry at 10 surfaces evenly spaced throughout the volume. The quasi-symmetry metric f_C is given by

$$f_C = (M\iota - N)(\boldsymbol{B} \times \nabla\psi) \cdot \nabla B - (MG + NI)\boldsymbol{B} \cdot \nabla B. \tag{5.11}$$

Here, $\boldsymbol{B}$ is the magnetic field with magnitude B, M and N give the type of quasi-symmetry (for quasi-axisymmetric, $M = 1$, $N = 0$), and G and I are the poloidal and toroidal covariant components of the field in Boozer coordinates.

The plasma boundary after optimization is shown in figure 3, and we can see that we have successfully avoided any concave regions that may be difficult to produce with external coils. In figure 6 we also see that the level of quasi-symmetry in this new configuration compares quite well with similar optimized designs such as NCSX. However, upon further analysis we find that this new configuration has a negative magnetic well, indicating it is MHD unstable.

We can attempt to rectify this by adding a constraint on the magnetic well shown in (5.12). Here $V(\rho)$ is the volume enclosed by the flux surface labelled by ρ and $\langle \cdots \rangle$ is a flux surface average. For vacuum equilibria, this is physically equivalent to the more common metric $V''(\psi) < 0$ but generalizes to finite beta as well (Landreman & Jorge 2020). The constraint is enforced at 20 surfaces evenly spaced in ρ. The definition for magnetic well used is

$$W(\rho) = \frac{V}{\langle B^2 \rangle \partial_\rho V} \partial_\rho \langle 2\mu_0 p + B^2 \rangle > 0. \tag{5.12}$$

Adding this constraint and re-running the optimization, we find that the optimizer does not converge, indicating that the constraints cannot all be satisfied at the same time. This is perhaps expected, as empirical evidence suggests that bean-like cross-sections can be

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

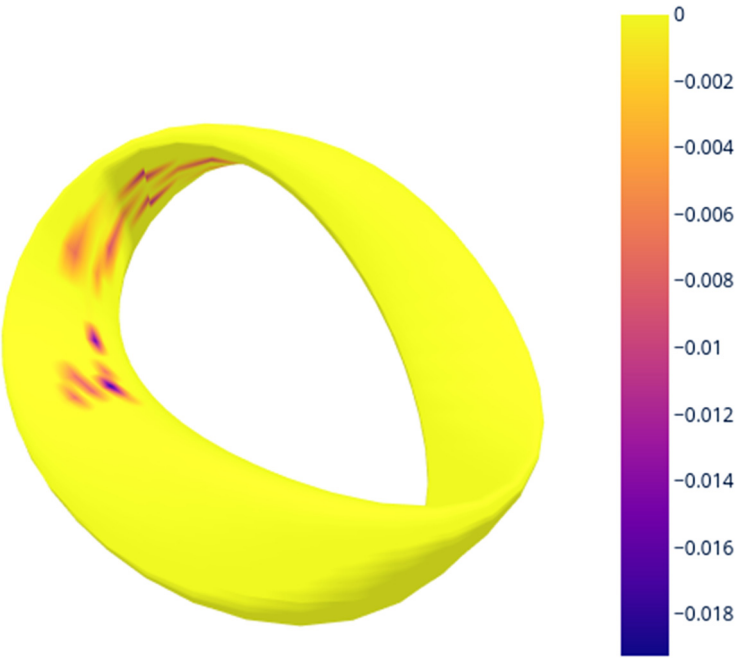

FIGURE 4. Lagrange multipliers for the curvature constraint plotted on the surface. The dark areas indicate places where the constraint is binding, and the optimizer wants to indent the boundary in those locations to improve the magnetic well.

favourable for MHD stability (Nührenberg & Zille 1986; Nührenberg 2010), though may not be strictly required from a near axis formalism (Rodríguez 2023).

However, in this case we can use the estimated Lagrange multipliers to get an idea of which constraints are the most limiting. Because the curvature constraint is enforced pointwise in real space, the Lagrange multipliers also have a local characteristic, and we can plot them over the surface to see where the optimizer would indent the surface, as shown in figure 4. As expected, the curvature constraint is most prominent on the inboard side, and the negative sign of the multipliers indicates that increasing the curvature in those regions will be favourable for stability.

Using this information, we re-run the optimization with the magnetic well constraint, but reduce the upper bound on mean curvature from 0 m^{-1} to 0.5 m^{-1}. With this relaxed constraint we are able to achieve a stable magnetic well, while maintaining most of the benefits of the curvature constraint, as shown in figure 5. This is not without some cost, as we see in figure 6 that the level of quasi-symmetry for the stable configuration is degraded somewhat compared with the unstable one, though still at reasonable levels compared with other optimized configurations such as NCSX.

6. Conclusion

In this work we have described the many ways that constraints naturally arise in stellarator optimization problems and surveyed a number of methods to solve constrained optimization problems, including methods that have not been widely applied in the field thus far. We have further developed and implemented a new constrained optimization algorithm in the DESC code and demonstrated its usefulness in designing a newly optimized equilibrium that achieves good quasi-symmetry and a stable magnetic well

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

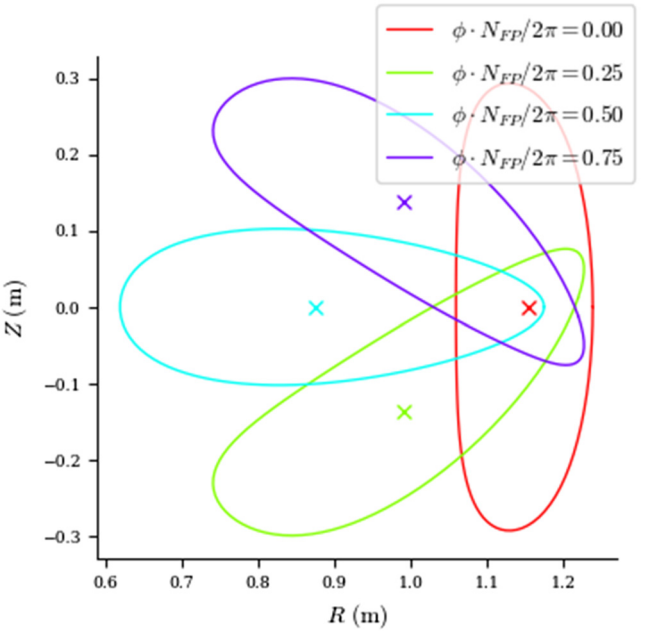

FIGURE 5. Plasma boundary of new optimized configuration with relaxed curvature constraint and stable magnetic well.

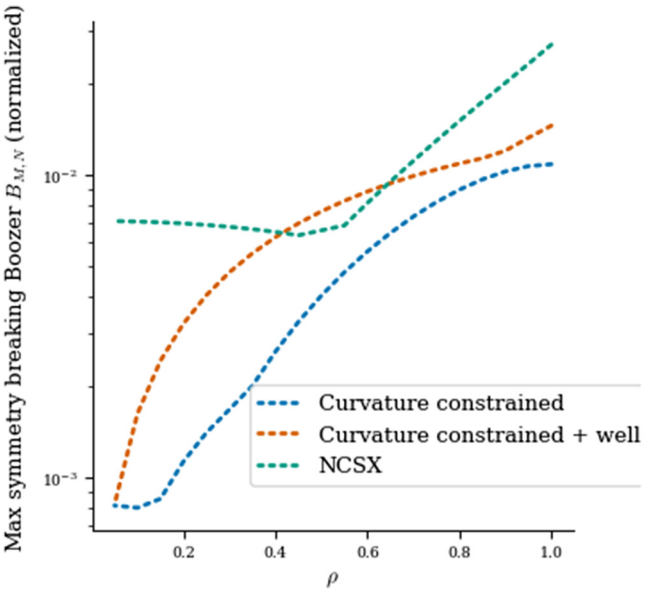

FIGURE 6. Quasi-symmetry error measured by the maximum amplitude of the symmetry breaking Boozer harmonics for the newly optimized configuration with and without a magnetic well constraint, compared with NCSX.

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press

without strong shaping of the plasma boundary. We are hopeful that these new techniques will open new avenues of possibility for the design and optimization of future stellarators, and significantly reduce the amount of guesswork and 'black magic' required when designing objectives for optimization, as well as offering insights into trade offs between different objectives and constraints.

Acknowledgements

The United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this paper, or allow others to do so, for United States Government purposes.

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

Funding

This work was supported by the U.S. Department of Energy under contract numbers DE-AC02-09CH11466, DE- SC0022005 and Field Work Proposal No. 1019.

Declaration of interest

The authors report no conflict of interest.

Data availability statement

The source code 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

BIEGLER, L.T. & ZAVALA, V.M. 2009 Large-scale nonlinear programming using IPOPT: an integrating framework for enterprise-wide dynamic optimization. *Comput. Chem. Engng* **33** (3), 575–582.

BINDEL, D., LANDREMAN, M. & PADIDAR, M. 2023 Understanding trade-offs in stellarator design with multi-objective optimization. *J. Plasma Phys*. **89** (5), 905890503.

COLEMAN, T.F. & LI, Y. 1996 An interior trust region approach for nonlinear minimization subject to bounds. *SIAM J. Optim*. **6** (2), 418–445.

CONLIN, R., DUDT, D.W., PANICI, D. & KOLEMEN, E. 2023 The DESC stellarator code suite. Part 2. Perturbation and continuation methods. *J. Plasma Phys*. **89** (3), 955890305.

CONN, A.R., GOULD, N.I.M. & PHILIPPE, L. 2000 *Toint. Trust-Region Methods*. MPS/SIAM Series on Optimization. Society for Industrial Mathematics.

CONN, A.R., GOULD, N.I.M. & TOINT, P.L. 2013 *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, vol. 17. Springer Science & Business Media.

DO CARMO, M.P. 2016 *Differential Geometry of Curves and Surfaces*, revised and updated 2nd edn. Courier Dover Publications.

DUDT, D., CONLIN, R., PANICI, D., KOLEMEN, E., UNALMIS, K. & KIM, P. 2024 DESC.

DUDT, D.W., CONLIN, R., PANICI, D. & KOLEMEN, E. 2023 The DESC stellarator code suite part 3: quasi-symmetry optimization. *J. Plasma Phys*. **89** (2), 955890201.

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

FENG, Y., SARDEI, F., GRIGULL, P., MCCORMICK, K., KISSLINGER, J. & REITER, D. 2006 Physics of island divertor plasmas in stellarators. *Contrib. Plasma Phys*. **46** (7-9), 504–514.

FOWKES, J.M., *et al.* 2023 Galahad 4.0: an open source library of Fortran packages with C and Matlab interfaces for continuous optimization. *J. Open Source Softw*. **8** (87), 4882.

GILL, P.E., MURRAY, W. & SAUNDERS, M.A. 2005 Snopt: an SQP algorithm for large-scale constrained optimization. *SIAM Rev*. **47** (1), 99–131.

ISAEV, M.Y., MIKHAILOV, M.I. & SHAFRANOV, V.D. 1994 Quasi-symmetrical toroidal magnetic systems. *Plasma Phys. Rep*. **20** (4).

JOHNSON, S.G. & SCHUELLER, J. 2021 NLopt: nonlinear optimization library. ascl-2111. Astrophysics Source Code Library.

KAPPEL, J., LANDREMAN, M. & MALHOTRA, D. 2024 The magnetic gradient scale length explains why certain plasmas require close external magnetic coils. *Plasma Phys. Control. Fusion* **66** (2), 025018.

KRUGER, T.G., ZHU, C., BADER, A., ANDERSON, D.T. & SINGH, L. 2021 Constrained stellarator coil curvature optimization with focus. *J. Plasma Phys*. **87** (2), 175870201.

LANDREMAN, M. & BOOZER, A.H. 2016 Efficient magnetic fields for supporting toroidal plasmas. *Phys. Plasmas* **23** (3).

LANDREMAN, M., BULLER, S. & DREVLAK, M. 2022 Optimization of quasi-symmetric stellarators with self-consistent bootstrap current and energetic particle confinement. *Phys. Plasmas* **29** (8).

LANDREMAN, M. & JORGE, R. 2020 Magnetic well and Mercier stability of stellarators near the magnetic axis. *J. Plasma Phys*. **86** (5), 905860510.

LANDREMAN, M. & PAUL, E. 2022 Magnetic fields with precise quasisymmetry for plasma confinement. *Phys. Rev. Lett*. **128** (3), 035001.

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

NÜHRENBERG, J. 2010 Development of quasi-isodynamic stellarators. *Plasma Phys. Control. Fusion* **52** (12), 124003.

NÜHRENBERG, J. & ZILLE, R. 1986 Stable stellarators with medium β and aspect ratio. *Phys. Lett.* A **114** (3), 129–132.

PANICI, D., CONLIN, R., DUDT, D.W., UNALMIS, K. & KOLEMEN, E. 2023 The desc stellarator code suite. Part 1. Quick and accurate equilibria computations. *J. Plasma Phys*. **89** (3), 955890303.

RODRÍGUEZ, E. 2023 Magnetohydrodynamic stability and the effects of shaping: a near-axis view for tokamaks and quasisymmetric stellarators. *J. Plasma Phys*. **89** (2), 905890211.

RODRIGUEZ, E., PAUL, E.J. & BHATTACHARJEE, A. 2022 Measures of quasisymmetry for stellarators. *J. Plasma Phys*. **88** (1), 905880109.

WÄCHTER, A. & BIEGLER, L.T. 2006 On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program*. **106**, 25–57.

https://doi.org/10.1017/S0022377824000655 Published online by Cambridge University Press