SIAM J. SCI. COMPUT.
Vol. 43, No. 6, pp. B1198–B1225

A PARALLEL CUT-CELL ALGORITHM FOR THE FREE-BOUNDARY GRAD–SHAFRANOV PROBLEM*

SHUANG LIU†, QI TANG‡, AND XIAN-ZHU TANG‡

Abstract. A parallel cut-cell algorithm is described to solve the free-boundary problem of the Grad–Shafranov equation. The algorithm reformulates the free-boundary problem in an irregular bounded domain and its important aspects include a searching algorithm for the magnetic axis and separatrix, a surface integral along the irregular boundary to determine the boundary values, an approach to optimize the coil current based on a targeting plasma shape, Picard iterations with Aitken's acceleration for the resulting nonlinear problem, and a Cartesian grid embedded boundary method to handle the complex geometry. The algorithm is implemented in parallel using a standard domain-decomposition approach and a good parallel scaling is observed. Numerical results verify the accuracy and efficiency of the free-boundary Grad–Shafranov solver.

Key words. cut-cell, free-boundary problem, Grad–Shafranov equation

AMS subject classifications. 65N06, 65N55, 76W05

DOI. 10.1137/20M1385470

1. Introduction.

Tokamak fusion relies on magnetic confinement of a plasma at a temperature of around 10 keV and a particle density of $10^{20}/\mathrm{m}^3$. The force balance is achieved by running a current inside the plasma that produces a Lorentz force to counter the plasma pressure gradient. In an axisymmetric configuration like a tokamak, such an equilibrium is described by an elliptic equation for the poloidal magnetic flux, commonly known as the Grad–Shafranov equation. In addition to the plasma current, electrical currents are also carried in toroidal and poloidal magnetic field coils outside the plasma chamber to produce the confining magnetic field. The coil current can be individually adjusted to accommodate a variety of plasma pressure and current profiles in terms of plasma positioning and shaping.

In a fixed-boundary Grad–Shafranov solver, both the location of the computational boundary and the boundary condition for the poloidal magnetic flux are known. Many numerical approaches, including spectral elements [36, 24], hybridizable discontinuous Galerkin methods [45, 46], boundary integral approaches [42, 33], Hermite finite element [40, 25], and discontinuous Petrov Galerkin methods [43], have been extended to solve the fixed-boundary problem. A common use of the fixed-boundary Grad–Shafranov solver is to set the computational boundary to be the targeted last closed flux surface, so the plasma shaping is enforced by the computational boundary on which the poloidal flux is a constant. The free-boundary Grad–Shafranov solver aims to account for the coil currents as well as the plasma current on plasma

*Submitted to the journal's Computational Methods in Science and Engineering section December 11, 2020; accepted for publication (in revised form) August 25, 2021; published electronically December 13, 2021.

https://doi.org/10.1137/20M1385470

Funding: This work was supported by the U.S. Department of Energy through the Fusion Theory Program of the Office of Fusion Energy Sciences and by the Tokamak Disruption Simulation (TDS) SciDAC partnerships between the Office of Fusion Energy Science and the Office of Advanced Scientific Computing.

†Los Alamos National Laboratory, Los Alamos, NM 87545 USA (shuangliu@lanl.gov). Current address: Department of Mathematics, University of California, San Diego, La Jolla, CA 92093 USA (shl083@ucsd.edu).

‡Los Alamos National Laboratory, Los Alamos, NM 87545 USA (qtang@lanl.gov, xtang@lanl.gov).

positioning and shaping. It is an essential tool for tokamak machine design and specific experimental discharge planning. Most tokamak facilities around the world have in-house development and/or maintenance of free-boundary Grad–Shafranov solvers. There has also been more recent interest that has a focus on the numerical aspects of the free-boundary problem [23, 16, 22].

Computationally the free-boundary problem for the Grad–Shafranov equation is more challenging as the plasma shape is not known a priori. The general solution strategy involves an iterative approach that combines a fixed-boundary solver with a Green's function representation for the contribution to the poloidal magnetic flux on the computational boundary from the coil current external to the computational boundary. The solution process is iterative by nature and a Picard iteration is usually deployed to drive the convergence of the poloidal magnetic flux on the computational boundary. The choice of this fiducial computational boundary is constrained by two considerations: (1) the computational boundary shall enclose all chamber volume that current-carrying plasma can access, and (2) all external currents should be outside the computational boundary. Figure 1 shows the plasma region (colored red) and the external currents (coils and solenoids). The most natural place for such a fiducial computational boundary in a free-boundary Grad–Shafranov solver is the vacuum vessel (the blue region in Figure 1) of which the poloidal field currents are all outside, or the first wall (as indicated in Figure 1). Both first wall and vacuum vessel in modern tokamaks are strongly shaped and give rise to an irregular computational domain, for which some variation of finite element/volume on an unstructured grid would be a ready choice for the underlying fixed-boundary Grad–Shafranov solver. It is interesting to note that the essential ingredients of free-boundary Grad–Shafranov solvers, for example, both the Green's function formulation and the iterative solution

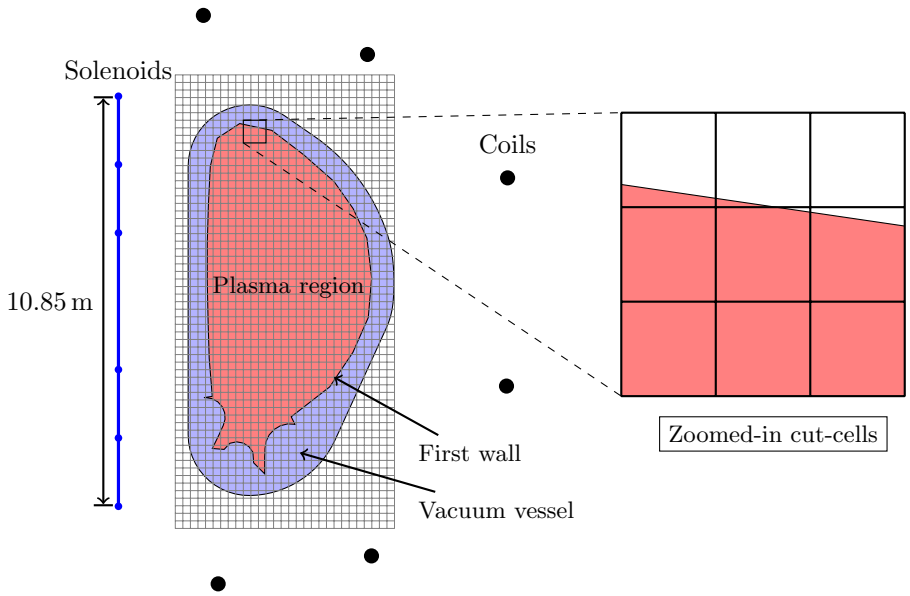


FIG. 1. *ITER tokamak geometry. Left: ITER plasma region, external currents, and an exemplar cut-cell mesh. Right: zoomed-in view. Note there are two types of external currents in ITER, i.e., six coils (point source) and solenoids of five segments (line source). The geometry uses the actual dimensions from ITER.*

procedure, were first developed with finite difference discretization over a structured grid in a regular domain [32, 30, 27, 28]. This is partly due to the considerable freedom in the placement of the fiducial computational boundary despite the two general constraints noted earlier, and partly due to the fact that many tokamaks have enough space between current-carrying plasmas and poloidal field coils for a rectangular computational boundary between them. For some of the next generation tokamaks, it becomes more desirable to have the computational boundary aligned with the better flux conserver, for example, the vacuum vessel in ITER [2], or with the first wall, such as the case with the SPARC tokamak [13], where the coils are very close to the plasma region. Also, note that it is desirable to represent the field on a structured grid, which can significantly accelerate the particle pusher in a particle-in-cell code. We find that for these applications, cut-cells (as indicated in Figure 1) can be used in combination with finite difference in a structured grid to accommodate an irregular computational domain for a free-boundary Grad–Shafranov solver.

Numerical solutions to elliptic problems with complex geometry, of which the Grad–Shafranov equilibrium is one example, have been considered by many approaches, including finite difference, finite volume, and finite element using various meshing techniques. Among those numerical approaches, the cut-cell approach has a number of advantages, which include simplifying the grid generation process for complicated geometries, enabling fast computation of the solution in parallel, and shifting the complexity of dealing with complex geometries to the discretization approach. For more details, one can consult the review in [6]. The cut-cell approach, a.k.a. the Cartesian grid embedded boundary method, generates the mesh using a background regular mesh and taking special care of cut-cells where the geometry intersects the grid. A variety of works have addressed the successfully use of the cut-cell approach for solving elliptic equations with finite volume [29, 48, 14, 15] and finite difference discretization [31, 19, 18, 35]. When discretized using the flux-divergence form, finite volume methods have the advantage of staying discretely conservative [34], which is important in solving many problems such as heat and mass transfer. In this work, we employ the cut-cell approach with a conservative discretization to address the Grad–Shafranov equation in irregular domains.

The paper is organized as follows. Section 2 gives a brief description of the Grad–Shafranov equation and the associated fixed-boundary and free-boundary problems. Section 3 discusses the cut-cell algorithm, which is a Cartesian grid embedded boundary method, as well as a level set approach to improve the efficiency of the cut-cell algorithm. This is followed by a detailed description of a free-boundary Grad–Shafranov solver, focusing on solving the free-boundary problem reformulated on a bounded domain in section 4. In section 5, the parallel implementation is described and some details of the implementation of the free-boundary Grad–Shafranov solver are presented. Finally, numerical tests are presented in section 6, demonstrating the accuracy and efficiency of the free-boundary Grad–Shafranov solver.

2. Grad–Shafranov equation. In this section, the Grad–Shafranov equation, fixed-boundary problem, and free-boundary problem are briefly introduced.

The Grad–Shafranov equation is derived from the MHD equilibrium equations given by

$$\nabla p = \mathbf{J} \times \mathbf{B}, \tag{2.1a}$$

$$\mu_0 \mathbf{J} = \nabla \times \mathbf{B}, \tag{2.1b}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{2.1c}$$

where $\mathbf{J}$ is the electric current, $\mathbf{B}$ is the magnetic field, μ_0 is the magnetic permeability, and p is the plasma pressure. Equation (2.1a) is the force balance equation. Equation (2.1b) is the Ampère's law. Equation (2.1c) is the divergence-free constraint for the magnetic field. In the case of equilibrium, the above system (2.1) holds in the entire region $\mathbb{R}^3$.

The Grad–Shafranov equation is usually written in a cylindrical coordinate system, denoted by (R, ϕ, Z), for an axisymmetric plasma such as those in a tokamak. It is well known that the magnetic field in the tokamak can be represented by

$$\mathbf{B} = \nabla\phi \times \nabla\psi + g(\psi)\nabla\phi,$$

where ψ is commonly referred to as the poloidal flux function and g is a scalar function. The Grad–Shafranov equation can be derived from (2.1) as

$$\Delta^*\psi = \mu_0 R J_\phi(R, \psi), \tag{2.2}$$

where the toroidal elliptic operator is defined as

$$\Delta^*\psi \equiv \frac{\partial^2\psi}{\partial R^2} - \frac{1}{R}\frac{\partial\psi}{\partial R} + \frac{\partial^2\psi}{\partial Z^2},$$

and the source term satisfies

$$\mu_0 R J_\phi(R, \psi) = -\left[\mu_0 R^2 \frac{d\,p(\psi)}{d\psi} + g(\psi)\frac{d\,g(\psi)}{d\psi}\right],$$

in which the plasma pressure, p, is a scalar function of ψ. Note that the toroidal elliptic operator can be rewritten in a divergence form

$$\Delta^*\psi = R\tilde{\nabla}\cdot\Big(\frac{1}{R}\tilde{\nabla}\psi\Big),$$

where $\tilde{\nabla} \equiv [\partial_R, \partial_Z]^T$. This conservative form will be used when discretizing the problem with the cut-cell approach.

Note that in a tokamak plasma, J_ϕ is carried by the magnetically confined plasma that resides inside the magnetic separatrix in the formulation of both fixed- and free-boundary Grad–Shafranov equilibria, which we shall explain next.

2.1. Fixed-boundary problem. The fixed-boundary problem refers to the situation when the boundary condition for the equilibrium problem is known. Mathematically, the problem can be simply cast as a boundary value problem given by

$$\begin{aligned} &\Delta^*\psi = \mu_0 R J_\phi(R, \psi), && (R, Z) \in \Omega, \\ &\psi = \psi_D, && (R, Z) \in \partial\Omega, \end{aligned} \tag{2.3}$$

where Ω is the physical domain with a Lipschitz boundary $\partial\Omega$. In tokamaks, the physical domain Ω corresponds to the cross section of the device bounded by the first wall or, more commonly, the last closed flux surface beyond which J_ϕ vanishes. In this work, the fixed-boundary problem is also implemented as an important tool to verify the numerical scheme.

2.2. Free-boundary problem. The free-boundary problem refers to the situation when the plasma domain, denoted as $\mathcal{P}(\psi)$, is unknown. In tokamak, the plasma domain refers to the region inside the device where the confinement actually happens,

and thus $\mathcal{P}(\psi)$ is filled with hot plasmas. In a diverted tokamak plasma, $\mathcal{P}(\psi)$ is the magnetic separatrix that demarcates the region of nested closed flux surfaces and the scrape-off layer that has magnetic field lines intercept the divertor and first wall. For a limited plasma, $\partial\mathcal{P}(\psi)$ is the last closed flux surface beyond which the flux surfaces are intercepted by one or multiple limiters that protrude from the chamber wall. It is important to note that $\mathcal{P}(\psi)$ is meaningful only when such an MHD equilibrium exists. Such a constraint poses an extra challenge to numerical algorithms, since the success of finding a numerical solution to the free-boundary problem implicitly indicates the existence of such a domain, while a perturbed numerical solution may indicate an unrealistic plasma domain, resulting in the divergence of algorithms. Therefore, it is necessary to design the numerical algorithm to be robust with respect to such a perturbation.

Assuming an unbounded domain $\mathcal{H} := [0,\infty) \times (-\infty,\infty)$, the free-boundary problem is given by

$$\Delta^*\psi = \begin{cases} \mu_0 R J_\phi(R,\psi), & (R,Z) \in \mathcal{P}(\psi), \\ \mu_0 R I_i, & (R,Z) = (R_i^c, Z_i^c),\ i = 1,2,\dots,n_c, \\ 0 & \text{otherwise}, \end{cases} \tag{2.4}$$

with the asymptotic constraint

$$\psi(0,Z) = 0 \quad \text{and} \quad \lim_{\|(R,Z)\|\to\infty} \psi(R,Z) = 0. \tag{2.5}$$

Here I_i is the external current density in the ith poloidal field coil located at $(R_i^c, Z_i^c) \in \mathcal{H}$ (see Figure 1 for the locations of coils; solenoids can be handled similarly), and $J_\phi(R,\psi)$ is the prescribed toroidal component of the plasma current density, generally a nonlinear function of ψ, in the plasma domain $\mathcal{P}(\psi)$. Note that the magnetic potential outside of the plasma region simply satisfies Poisson's equation. We further introduce the limiter domain $\mathcal{L}$, satisfying $\mathcal{P}(\psi) \subset \mathcal{L} \subset \mathcal{H}$. For this purpose, the limiter corresponds to the first wall in the tokamak device surrounding the confinement region. The limiter domain is the entire region accessible by the plasma, while the plasma domain $\mathcal{P}(\psi)$ is the region bounded by the last closed poloidal flux surface inside the limiter-bounded domain; see the discussion in [16], for instance.

The free-boundary problem is still underdetermined because (a) the source term J_ϕ (more specifically, p and g) is typically provided as a function of the normalized $\bar{\psi} \in [0,1]$ from experiment (its normalization shall be explained later); (b) the coil currents are not known and in fact they need to be optimized based on the targeted shape of $\mathcal{P}(\psi)$, along with solving the free-boundary problem. Both of the uncertainties introduce more challenges to solve the free-boundary problem. However, the uncertainties are closely related to the concept of plasma shape control in tokamaks, which is a critical topic in tokamak discharge design as well as in tokamak machine design.

To address (a), one can define a normalized flux function,

$$\bar{\psi} \equiv \frac{\psi - \psi_o}{\psi_X - \psi_o},$$

where ψ_X is the value of ψ at the plasma boundary, $\partial\mathcal{P}(\psi)$, and ψ_o is the value at the magnetic axis. In a diverted tokamak plasma, a magnetic *separatrix* demarcates the topologically distinct closed and open flux regions. Plasmas also reside in the open

flux region outside the separatrix, which is known as the scrape-off layer, but it has relatively small pressure gradient and plasma current. For the purpose of defining a Grad–Shafranov equilibrium, the plasma boundary is usually given by the separatrix or a closed flux surface in the immediate vicinity of the separatrix. In this paper, we will set ψ_X to be the ψ value of the saddle point that labels the magnetic separatrix. In a limited plasma, ψ_X labels the last closed flux surface beyond which magnetic field lines interact with a limiter protruded from a camber wall. Since ψ is the poloidal flux, it must be a monotonic function between ψ_o and ψ_X in a tokamak. The normalized flux $\bar{\psi}$ is therefore between 0 and 1 for all ψ values inside $\mathcal{P}(\psi)$. Numerical techniques to efficiently locate those points will be discussed in the following section. Note it is expected that the overall source term can have jumps on the right-hand side in (2.4). The existence of solutions to (2.4) under certain normalization and jump source terms has been considered in some PDE theory studies; see [1], for instance.

In addressing (b), one usually imposes some constraints based on a targeted plasma shape $\mathcal{P}_{\text{target}}$. The targeted plasma shape is predetermined (up to some small variation) when the fusion device is designed. The standard approach is to impose a few control points along the desired shape such that the computed poloidal flux on those points is equal to the value on the separatrix, i.e.,

$$\psi(R_k, Z_k) = \psi_X, \quad k = 1, \dots, n_p,$$

where (R_k, Z_k) is selected along the plasma boundary $\partial\mathcal{P}_{\text{target}}$ and ψ_X is determined numerically during the normalization. It essentially means we minimize the distance between the separatrix from the numerical solution and the targeted plasma shape. In the following section, we will discuss how to impose a proper optimization problem to determine the coil current based on the above constraints.

3. Cut-cell algorithm. A cut-cell algorithm for the free-boundary Grad–Shafranov problem is described in this section. The basic algorithm closely follows the cut-cell algorithm proposed for Poisson's equation in [29]. We further introduce a level set function to improve the efficiency of the cut-cell algorithm and suit the specific needs of the Grad–Shafranov equation.

Consider a uniform grid defined by

$$\mathbf{x}_{i,j} = (R_i, Z_j) = (R_{\min} + i\Delta R,\, Z_{\min} + j\Delta Z).$$

Let $\psi_{i,j}$ be a grid-point approximation to $\psi(\mathbf{x}_{i,j})$ and rewrite (2.2) as

$$\tilde{\nabla} \cdot \Big(\frac{1}{R}\tilde{\nabla}\psi\Big) = \mu_0 J_\phi(R, \psi).$$

Integrating it on the cell corresponding to the grid point (i, j) and applying the divergence theorem give

$$\frac{1}{\Delta R \Delta Z} \int_{\partial C} \Big(\frac{1}{R}\tilde{\nabla}\psi\Big) \cdot \mathbf{n}\, ds = \frac{1}{\Delta R \Delta Z} \int_C \mu_0 J_\phi(R, \psi)\, dR dZ, \quad C \in \Omega, \tag{3.1}$$

where $\mathbf{n}$ denotes the unit outward normal direction vector of ∂C. Assuming C is a full cell, (3.1) can be discretized as

$$\frac{1}{\Delta R \Delta Z}\Big[F_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}} + F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}\Big] = \mu_0 J_\phi(R_{i,j}, \psi_{i,j}), \tag{3.2}$$

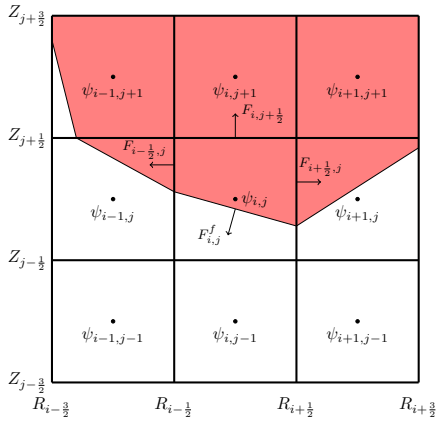


FIG. 2. *Grid points and cut-cells. The red region is the computational domain.*

where $F_{r,s}$ is the flux along the cell edge, given by

$$F_{i,j+\frac{1}{2}} = \frac{\Delta R}{R_{i,j+\frac{1}{2}}} \frac{\psi_{i,j+1} - \psi_{i,j}}{\Delta Z}, \qquad F_{i+\frac{1}{2},j} = \frac{\Delta Z}{R_{i+\frac{1}{2},j}} \frac{\psi_{i+1,j} - \psi_{i,j}}{\Delta R}.$$

Note that (3.2) on the full cell is a standard second-order discrete operator.

If C is a cut-cell as shown in Figure 2, introducing a volume fraction $\Lambda_{i,j} \in [0,1]$, the resulting discretization of (3.1) can be written as

$$\frac{1}{\Delta R \Delta Z \Lambda_{i,j}} \left[F_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}} + F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j} - F^f_{i,j} \right] = \mu_0 J_\phi(R_{i,j}, \psi_{i,j}), \tag{3.3}$$

where $F^f_{i,j}$ is the flux along the boundary in the cut-cell. Note that $F_{r,s}$ can be 0 if the corresponding edge is not involved in the control volume.

We further introduce a level set function for efficiently distinguishing interior cell, exterior cell, and cut-cell as well as determining whether a cell edge is a partial edge or a full edge. We first construct a level set function $\rho(R,Z)$ given by

$$\rho(R,Z) = \begin{cases} +d, & (R,Z) \in \Omega^c, \\ 0, & (R,Z) \in \partial\Omega, \\ -d, & (R,Z) \in \Omega, \end{cases}$$

where d is the shortest distance from the point (R,Z) to the boundary $\partial\Omega$. For convenience, let $\rho_{i+\frac{1}{2},j+\frac{1}{2}}$ denote $\rho(\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}})$, which is evaluated on the corners of each cell.

For the boundary described by an analytical expression, defining a level set value on a given point is straightforward and therefore it is ignored. However, in a practical problem of a tokamak geometry, the geometry is described by a set of points on the boundary, and thus it is less obvious to define the value of level set function on a point. Here, we present how to define the level set function for a general irregular domain Ω. Given a set of data $\{(r_1,z_1),(r_2,z_2),(r_3,z_3),\ldots,(r_n,z_n)\}$ which describes the boundary of the irregular domain, we connect all the points as a polygon. Let

$$\bar{r} = \sum_{i=1}^{n} \frac{r_i}{n}, \quad \bar{z} = \sum_{i=1}^{n} \frac{z_i}{n}, \quad \bar{P} = (\bar{r},\bar{z}), \quad P_i = (r_i,z_i), \quad i = 1,2,\ldots,n.$$

To compute d for a given point $P = (r, z)$, we locate the boundary point P_i which has the shortest distance to P. Then we determine d as the shorter distance between P and the two boundary segments connected with P_i.

The next step is to decide whether P is in or out of domain if $d \neq 0$. We choose (r_1, z_1) as a starting point and compute the rotated angle θ_i from $\overrightarrow{\bar{P}P_1}$ to $\overrightarrow{\bar{P}P_i}$ where $\theta_i \in (-\pi, \pi]$. For any given point $P = (r, z)$, by checking the value of the angle θ rotating from $\overrightarrow{\bar{P}P_1}$ to $\overrightarrow{\bar{P}P}$, we can get the interval $[\theta_i, \theta_{i+1}]$ where θ belongs. We denote $P_{intersect}$ as the intersection point of the line passing through P and $\bar{P}$ and the line passing through P_i and P_{i+1}. By comparing values of the distance between $\bar{P}$ and $P_{intersect}$ and the distance between $\bar{P}$ and P, we can determine whether the point P is in or out of domain. Some care is needed for the corner point where the curve becomes not convex.

The precomputed level set function can be used to distinguish the cell category. For example, in Figure 2, since ρ at each corner of the cell $(i, j-1)$ is positive, we determine the cell $(i, j-1)$ is an exterior cell. Similarly, the cell $(i, j+1)$ is an interior cell in the domain Ω, as ρ values at all the corners of the cell are negative. The cell (i, j) is a cut-cell which contains part of the boundary as signs of ρ value at each corner are not the same. The precomputed level set function can also be used to evaluate some geometry quantities associated with the cut-cells [38, 37]. For a partial edge, we can use the information of the level set function ρ at the two endpoints to approximate its aperture. Supposing $\mathbf{x}_f = (R_{i+\frac{1}{2}}, Z_f) \in \partial\Omega$, assuming two points $\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}} \in \Omega$ and $\mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2}} \in \Omega^c$ border $\mathbf{x}_f$, we use the formula given in [12],

$$Z_{i+\frac{1}{2},j+\frac{1}{2}} - Z_f = \frac{\rho_{i+\frac{1}{2},j+\frac{1}{2}} \triangle Z}{\rho_{i+\frac{1}{2},j+\frac{1}{2}} - \rho_{i+\frac{1}{2},j-\frac{1}{2}}},$$
$$Z_f - Z_{i+\frac{1}{2},j-\frac{1}{2}} = \frac{-\rho_{i+\frac{1}{2},j-\frac{1}{2}} \triangle Z}{\rho_{i+\frac{1}{2},j+\frac{1}{2}} - \rho_{i+\frac{1}{2},j-\frac{1}{2}}},$$

to evaluate the distances between $\mathbf{x}_f$ and two points $\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}$ and $\mathbf{x}_{i+\frac{1}{2},j-\frac{1}{2}}$. Therefore, as an example, the fraction of the right edge of the partial cell (i, j) in Figure 2, denoted as $a_{i+\frac{1}{2},j}$, is given by

$$a_{i+\frac{1}{2},j} = \frac{\rho_{i+\frac{1}{2},j+\frac{1}{2}}}{\rho_{i+\frac{1}{2},j+\frac{1}{2}} - \rho_{i-\frac{1}{2},j+\frac{1}{2}}}.$$

After calculating the apertures of each cell edge, the area of the front, denoted as A^f, in the cut-cells can be evaluated based on the relationship

$$A^f_{i,j}\mathbf{n}^{in}_{i,j} = \Delta R(a_{i+\frac{1}{2},j} - a_{i-\frac{1}{2},j})\hat{i} + \Delta Z(a_{i,j+\frac{1}{2}} - a_{i,j-\frac{1}{2}})\hat{j},$$

where $\mathbf{n}^{in}$ is the inward-facing normal at the boundary in the cut-cell. Note that for interior cells, since all aperture values are 1, A^f is also applicable to the interior cells as $A^f = 0$.

Next, we discuss the discretization of (3.3) on cut-cells. $F_{i,j+\frac{1}{2}}$, $F_{i,j-\frac{1}{2}}$, $F_{i+\frac{1}{2},j}$, and $F_{i-\frac{1}{2},j}$ are evaluated at the midpoint of cell edge covered by Ω. To evaluate flux on a partial edge, in order to keep a second-order accurate approximation of the fluxes through cell edges, a linear interpolation between values at the midpoints of full edges is used. For example, in Figure 2, the flux $F_{i+\frac{1}{2},j}$ is evaluated at the midpoint of the

partial edge centered at $\mathbf{x}_{i+\frac{1}{2},j}$ by linearly interpolating the flux at neighboring full edge centered at $\mathbf{x}_{i+\frac{1}{2},j+1}$, which is stated as the following:

$$F_{i+\frac{1}{2},j} = \frac{1}{R_{i+\frac{1}{2},j}} \Delta Z \left[\frac{1 + a_{i+\frac{1}{2},j}}{2} \frac{(\psi_{i+1,j} - \psi_{i,j})}{\Delta R} + \frac{1 - a_{i+\frac{1}{2},j}}{2} \frac{(\psi_{i+1,j+1} - \psi_{i,j+1})}{\Delta R} \right].$$

The flux F^f is evaluated at the midpoint of the boundary covered by the cut-cell. A three-point gradient stencil is proposed in [29] to evaluate the normal component of the gradient of ψ and deployed in the current work. The readers are referred to [29] for the details of the discretization and other techniques such as small cell ignorance, which are also deployed here.

Finally, we discuss how to evaluate the source term of the Grad–Shafranov equation along with the cut-cells. Note that $\mu_0 J_\phi(R_{i,j}, \psi_{i,j})$ should be evaluated at the centroid of the covered part in each cut-cell. There are three types of cut-cells covered by the physical domain, triangle, trapezoid, and pentagon, based on the piecewise-linear representation of the boundary in each cut-cell. With the geometry information gathered, we can now turn the problem into computing the centroid of a convex and closed polygon. Suppose a convex and closed polygon is defined by the ordered vertices $\{(r_1, z_1), (r_2, z_2), \ldots, (r_{n+1}, z_{n+1})\}$ with $(r_{n+1}, z_{n+1}) = (r_1, z_1)$, the centroid of a convex polygon is computed as

$$C_{polygon} = \frac{1}{3} \left(\frac{\sum_{i=1}^{n} (r_i + r_{i+1})(r_i z_{i+1} - r_{i+1} z_i)}{\sum_{i=1}^{n} (r_i z_{i+1} - r_{i+1} z_i)}, \frac{\sum_{i=1}^{n} (z_i + z_{i+1})(r_i z_{i+1} - r_{i+1} z_i)}{\sum_{i=1}^{n} (r_i z_{i+1} - r_{i+1} z_i)} \right),$$

which will be used during the evaluation of the source term.

4. Free-boundary Grad–Shafranov solver. A common approach to solve the free-boundary problem (2.4) is to reformulate the problem in a bounded computational domain Ω; see [26, 16, 28], for instance. The requirement for Ω is that it should enclose the plasma domain $\mathcal{P}(\psi)$, but all the poloidal field coils, including those for vertical stability control and divertor flux shaping, should be outside Ω. This is to accommodate a Green's function approach in calculating the coil current contribution to the Grad–Shafranov equilibrium through the poloidal flux ψ on $\partial\Omega$. As the plasma domain $\mathcal{P}(\psi)$ is unknown a priori, the bounded domain Ω, in practice, should contain at least the limiter-bounded domain $\mathcal{L}$. The precise choice for $\partial\Omega$ constrains the numerical discretization for the Grad–Shafranov solver and motivates the cut-cell approach in section 3.

In this section we will focus on the overall scheme for the free-boundary Grad–Shafranov solver. The main algorithm is based on a Picard iteration,

$$\begin{aligned} \Delta^* \psi &= \mathcal{S}(\psi^{\text{old}}), \quad (R, Z) \in \Omega, \\ \psi &= \mathcal{D}(\psi^{\text{old}}), \qquad (R, Z) \in \partial\Omega, \end{aligned}$$

where the operators $\mathcal{S}$ and $\mathcal{D}$ stand for the steps to determine the source term and boundary values from the old solution ψ^{old}, respectively. The details of $\mathcal{S}$ and $\mathcal{D}$ will be given in the following discussion. Note that unlike a fixed-boundary problem [43], Newton's method is challenging to employ here, which will be further discussed in subsection 4.2.

The discussions will start from determining the source term from an old solution, i.e., the operator $\mathcal{S}$, which includes a search algorithm to locate ψ_o and ψ_X and a barycentric interpolation to interpolate the given source profile. After the source

term is determined, the approaches to determine the boundary condition, i.e., the operator $\mathcal{D}$, are discussed. Then a minimization problem is introduced to optimize the coil current density based on the old solution. Finally, the full algorithm based on the Picard iteration with Aitken's acceleration is summarized. Note that all the approaches in this section are applicable to a free-boundary solver on a rectangular domain and some of them were actually initially proposed for the rectangular domain.

4.1. Evaluate the source term. Evaluating the source term in the free-boundary problem is a multistep process. Since ψ_o and ψ_X are to be found from the solution, Grad–Shafranov equilibria are usually specified by prescribing the source term as functions of $\bar{\psi}$, which is always well-defined for any trial solution that has varying ψ_o and ψ_X. The most straightforward case has a known $g(\bar{\psi})$ and $p(\bar{\psi})$ profile, often supplemented by integral constraints of specified plasma beta and plasma toroidal current. A practically useful alternative is to specify the safety factor profile, $q(\bar{\psi})$, for $\bar{\psi} \in [0, 0.95]$.

Search for magnetic axis and saddle points. Since the trial solution of the Grad–Shafranov equation gives ψ on grid points, it is necessary to locate ψ_o and ψ_X first. As discussed in section 2, ψ_o and ψ_X correspond to the critical points where $\|\nabla\psi\| = 0$. Therefore, a simple minimization problem can be defined to find them,

$$\min_{R,Z} \|\nabla\psi\|^2. \tag{4.1}$$

To simplify the procedure, we use a two-step search algorithm to locate those points. We first evaluate the numerical gradient on all the grid points in order to identify several candidate points with small gradient values. We choose 10 candidate points in our solver (here 10 points are chosen to locate all the saddle points and the magnetic axis and they often converge to one of those points; during the searching process, if the point locates out of the computational domain, we filter out the corresponding candidate). Newton's method for optimization is then used to solve (4.1) using the candidate points as the initial guess. During the optimization, the Newton's method needs a continuous representation of ψ for any (R, Z), for which a barycentric interpolation formula is used. Since the initial guesses are very good, the Newton's method typically only needs a few iterations to converge. After all the critical points are found, a further check is performed to remove the duplicate critical points. Finally, ψ_o and ψ_X are determined based on the Hessian of ψ. If the critical point has $\partial_{rr}\psi\,\partial_{zz}\psi - (\partial_{rz}\psi)^2 > 0$, it is a local minimum or maximum, which corresponds to the magnetic axis (ψ_o), and if it has $\partial_{rr}\psi\,\partial_{zz}\psi - (\partial_{rz}\psi)^2 < 0$, it is a saddle point corresponding to the separatrix (ψ_X). We comment that the above algorithm does not guarantee finding a global minimum for a general solution. However, for a physical MHD equilibrium, the solution has a well-defined local minimum problem as we shall see in the numerical section, and thus the above algorithm works well. In the case when multiple local minimum/maximum points are found, we determine the solution is invalid. When that happens in the free-boundary solver, it means the solver has diverged and the iteration will stop. In the case when multiple saddle points are found, if the poloidal flux is convex inside the plasma region, we choose the smallest (or largest if ψ is concave) value among all the saddle points. Note the algorithm described above is a critical piece in the free-boundary solver and the subroutine has to be called routinely in the solver. As long as all the saddle points and the magnetic axis are successfully located, this ensures an accurate evaluation of the source term. Therefore, the number of candidate points (10 in our case) has no impact on the accuracy.

Given the plasma pressure profile and the toroidal field function profile as functions of the poloidal flux on a set of known points, finding a proper interpolation method to describe the source term in the Grad–Shafranov equation is very important. We note, however, at least on an equispaced case, polynomial interpolation is not proper, no matter what form we use, due to a Runge phenomenon of high order. It is reasonable to shift to rational interpolation [7, 8, 17, 47]. In our work, the barycentric rational interpolation (a fourth-order rational interpolation) is employed to evaluate the plasma pressure and the toroidal field function of the poloidal flux. This should allow highly accurate representation of the source term on grid points from numerical data profiles of $p(\bar{\psi})$ and $g(\bar{\psi})$.

4.2. Determine ψ_b on $\partial\Omega$. The approach to determine the boundary condition value from the external coil source is well known [26, 16, 28]. We give a brief discussion on two choices from our experiments. Note that there is a known Green's function for the toroidal elliptic operator Δ^*

$$G(\mathbf{R};\mathbf{R}') = \frac{1}{2\pi}\frac{\sqrt{RR'}}{k}[(2-k^2)K(k) - 2E(k)],$$

where $K(k)$ and $E(k)$ are complete elliptic integrals of the first and the second kind [44] and k is given by

$$k^2 = \frac{4RR'}{(R+R')^2 + (Z-Z')^2}.$$

A straightforward approach based on the Green's third identity defines the boundary value as

$$\psi_b(R',Z') = -\int_\Omega \mu_0 G(R,Z;R',Z')\,\tilde{J}_\phi(R,Z)\,drdz - \sum_{i=1}^{n_c}\mu_0 G(R_i^c,Z_i^c;R',Z')I_i, \tag{4.2}$$

where the source is given by $\tilde{J}_\phi(R,Z) = J_\phi(R,\psi^{\text{old}})$. A more efficient approach that only computes a line integral was proposed in [49]. It involves solving the elliptic problem with the same source term but a homogeneous boundary condition,

$$\begin{aligned} &\Delta^* U = \mu_0 R\,\tilde{J}_\phi(R,Z), && (R,Z)\in\Omega,\\ &U = 0, && (R,Z)\in\partial\Omega. \end{aligned} \tag{4.3}$$

Integrating the identity

$$\nabla\cdot\left[U\frac{1}{R^2}\nabla G(\mathbf{x};\mathbf{x}')\right] - \nabla\cdot\left[G(\mathbf{x};\mathbf{x}')\frac{1}{R^2}\nabla U\right] = \frac{1}{R^2}U\Delta^* G(\mathbf{x};\mathbf{x}') - \frac{1}{R^2}G(\mathbf{x};\mathbf{x}')\Delta^* U$$

over the computational domain Ω leads to the formulations for both boundary and interior points,

$$\psi_b(R',Z') = -\int_{\partial\Omega}\frac{dl}{R}G(R,Z;R',Z')\frac{\partial U}{\partial n} - \sum_{i=1}^{n_c}\mu_0 G(R_i^c,Z_i^c;R',Z')I_i, \tag{4.4}$$

$$\begin{aligned}\psi_{interior}(R',Z') = &-\int_{\partial\Omega}\frac{dl}{R}G(R,Z;R',Z')\frac{\partial U}{\partial n}\\ &-\sum_{i=1}^{n_c}\mu_0 G(R_i^c,Z_i^c;R',Z')I_i + U(R',Z').\end{aligned} \tag{4.5}$$

The second approach comes at the cost of inverting an elliptic operator, which only accounts for an extra cost of $O(N \log N)$ thanks to the algebraic multigrid (AMG) preconditioner we used. Thus, we typically choose the second approach in our solver for its efficiency. The second approach can also provide an initial guess on the entire domain through (4.5) in the initialization stage, which is useful to provide a better initial guess and speed up the convergence. One issue is that the Green's function is singular when $\mathbf{R}' = \mathbf{R}$, which is more problematic in the line integration of the second approach. In the implementation, when $\mathbf{R}'$ is too close to $\mathbf{R}$, we perturb $\mathbf{R}'$ with a small distance of ϵ, which typically happens in one or two cells. It appears to be sufficient to produce a smooth boundary condition.

Note that when Newton's method is applied to the free-boundary problem, (4.2) is more suitable to use in its nonlinear residual. This, however, results in a globally coupled system due to the global operator in (4.2). The Jacobian matrix assembly and its inversion are thus expensive. Moreover, AMG typically cannot solve such a system involving nonlocal boundary conditions. All those facts contribute to the challenge of employing Newton's method here. To resolve the issue due to ψ_b, [21] proposed to extend the computational domain to include all the external currents. This results in a much less banded system, which is easier to assemble, and the success of Newton's method but at the cost of solving on a domain much larger than that in the current work.

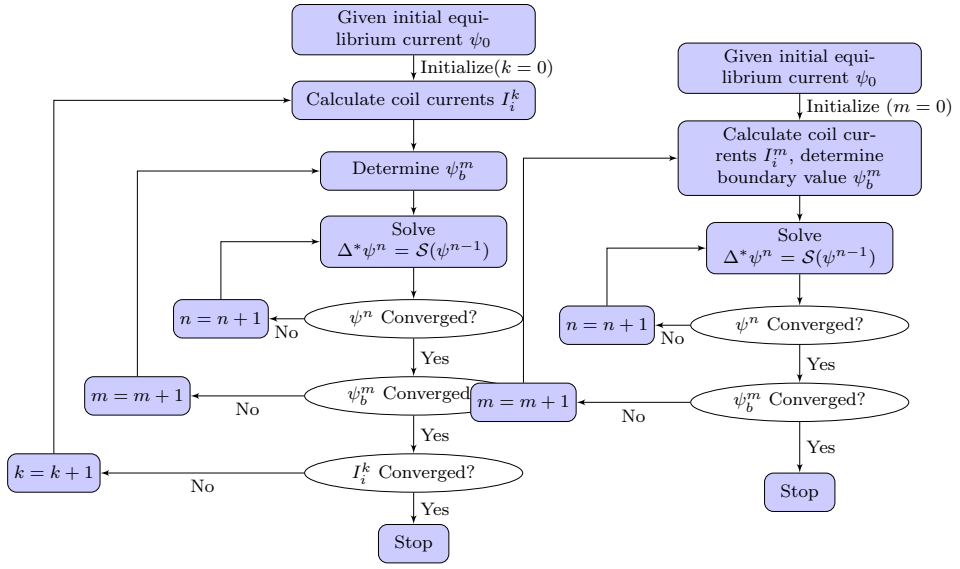
4.3. Determine coil currents. The last important piece in the algorithm is to determine the coil current after a solution ψ is given. It is important to adjust the coil current based on the solution so that the resulting plasma has the intended position and shape. We therefore optimize the current density through a minimization problem based on some constraints imposed through the targeted plasma shape. Given n_c coils in the device and the desired plasma region $\mathcal{P}_{\text{target}}$, we first select n_p points along its boundary $\partial\mathcal{P}_{\text{target}}$ such that the points are evenly spaced and $n_p > n_c$. Equation (4.5) provides a good approximation to ψ at those points for any current density. The search algorithm on the other hand provides the current separatrix value ψ_X. Minimizing the ψ values at those points with respect to ψ_X gives a minimization problem to find the required coil currents,

$$\mathbf{I}_c = \arg\min_{\mathbf{I}_c} \left\{ \gamma \sum_{k=1}^{n_c} {I_k}^2 + \sum_{j=1}^{n_p} \left[-\int_{\partial\Omega} \frac{dl}{R} G(R, Z; R'_j, Z'_j) \frac{\partial U}{\partial n} - \sum_{i=1}^{n_c} \mu_0 G(R^c_i, Z^c_i; R'_j, Z'_j) I_i + U(R'_j, Z'_j) - \psi_X \right]^2 \right\}, \tag{4.6}$$

where $\mathbf{I}_c$ is a vector of size n_c, consisting of the current values I_i, and the first term is a penalty term to prevent the system from becoming ill-posed.

We found that the resulting current density is sensitive to the choice of γ. Part of the reason is that the problem setup is based on practical units and the density values can be dramatically different from each other in different coils. To address that, we found the generalized cross validation (GCV) approach in [20] provides a good estimate of γ. To utilize GCV, the minimization problem (4.6) is written in the form of

$$\mathbf{I}_c = \arg\min_{\mathbf{I}_c} \left(\gamma \left\| \mathbf{I}_c \right\|^2 + \frac{1}{n_p} \left\| X\, \mathbf{I}_c - \mathbf{y} \right\|^2 \right),$$



(a) Three-loop algorithm (b) Two-loop algorithm

FIG. 3. *Flowcharts for the free-boundary Grad–Shafranov solvers.*

where (4.6) indicates X is a matrix of size $n_p \times n_c$, and $\mathbf{y}$ is a vector of size n_p. The GVC approach suggests the best γ should be computed as the minimizer of $V(\gamma)$, given by

$$V(\gamma) = \frac{1}{n_p}\|(I - A(\gamma))\mathbf{y}\|^2 \Big/ \left[\frac{1}{n_p}\,\mathrm{Trace}(I - A(\gamma))\right]^2,$$

where $A(\gamma) \equiv X(X^TX + n_p\gamma I)^{-1}X^T$ and I is an identity matrix of size $n_p \times n_p$. In the initial experiment, we implement a simple search algorithm to locate the optimized value of γ in each iteration for determining the current. We found that the optimized γ falls within the range of $[10^{-16}, 10^{-13}]$ for the practical ITER equilibrium problem. Therefore, to avoid recalculating γ in each iteration, we fix $\gamma = 10^{-15}$ in all of our numerical examples, which is sufficient to provide less sensitive current values in the final solver. The current values of the equilibrium are found in the range of $[10^3, 10^7]$, and thus the regularization term is not negligible with the given γ value.

4.4. Full algorithm. With all the important pieces laid out in the previous sections, a full algorithm for the free-boundary Grad–Shafranov equation will be described. In particular, two versions of the algorithm will be discussed and an additional acceleration technique will be adopted to improve the robustness and efficiency of the solver.

Three-loop algorithm. The first version of the algorithm would consist of three loops as the main steps (Figure 3(a)): the innermost loop inverts the operator Δ^* with a given boundary condition ψ_b; the outer loop will adjust ψ_b until some convergence threshold is reached for some fixed current density $\{I_i\}$ (see Figure 1 for the current locations); then the outermost loop would adjust the coil currents such that the magnetic separatrix will align with the control points. Although this idea appears to be plausible and in fact it is the first algorithm we implemented, we found that this version of the algorithm is sensitive to a small perturbation of the current, which could easily result in the failure of convergence of the two inner loops. One of the

fundamental issues in this approach is that there is an underlying assumption that for any given current density, there exists a corresponding equilibrium, despite not satisfying the plasma domain constraint. This, however, may not be true in general, and even if there exists such an equilibrium, its g and p profiles could be very different from the profiles being used, and therefore the evaluated source term could be very inaccurate, which could contribute to the divergence of the solver. Therefore, we do not pursue further using this version of the algorithm in the current work.

Two-loop algorithm [30]. The second version of the algorithm consists of two loops as the main steps (Figure 3(b)): the inner loop still inverts the operator Δ^* with a given boundary condition ψ_b while the outer loop simultaneously updates ψ_b and $\{I_i\}$ until some threshold with respect to ψ_b is reached. This version turns out to be much more robust than the previous version, which is the main algorithm we used in the free-boundary solver. Its robustness can be further improved by the acceleration techniques discussed below.

Aitken's acceleration. As previously discussed, the general procedure of the algorithm is a Picard iteration, and its drawback is its slow convergence. A common approach to accelerate its convergence is through underrelaxation, i.e.,

$$\psi^{new} = (1-\alpha)\psi^{old} + \alpha\tilde{\psi}^{new},$$

where α is the underrelaxation coefficient, typically between 0 and 1. The optimal value of α, however, is problem dependent and for nonlinear problems it can vary during the iterations. The optimal relaxation value can be approximated through the well-known Aitken's acceleration [41]. The version we adopted is given by

$$\begin{aligned} D\psi^{n+1} &= \psi^n - \tilde{\psi}^{n+1}, \\ \lambda^{n+1} &= \lambda^n + (\lambda^n - 1)\frac{(D\psi^n - D\psi^{n+1})^T D\psi^{n+1}}{\|D\psi^n - D\psi^{n+1}\|^2}, \\ \alpha &= 1 - \max(\min(\lambda^{n+1}, \lambda_{\max}), \lambda_{\min}), \\ \psi^{n+1} &= (1-\alpha)\psi^n + \alpha\tilde{\psi}^{n+1}, \end{aligned} \tag{4.7}$$

where $\tilde{\psi}^{n+1}$ is the solution directly solved from the Picard iteration, and $\lambda_{\min} = 0$ and $\lambda_{\max} = 0.95$ are typically chosen. Reference [9] suggests a preset value of $\lambda = 0.3$ in the first iteration, which is adopted in the current work. Note that the technique fits into the current framework very well due to its simplicity and computational expedience, and we use it to accelerate both the inner and outer loops. It is found that the Aitken's acceleration greatly improves the convergence of the full algorithm, which will be demonstrated in the numerical section. We note, however, that there are other techniques such as Anderson's acceleration and nonlinear GMRES available for the Picard iteration. We plan to explore more acceleration techniques in future work.

Finally, we summarize the full algorithm in Algorithm 4.1, which consists of two main loops, Aitken's accelerations and all the important steps described in the previous sections. Note that step 2 corresponds to all the steps related to the cut-cell algorithm, which will be addressed carefully in the next section. In the nonlinear solver, we set the relative difference of the boundary values $\epsilon_{out} = 1\mathrm{e}-4$ as the convergence criterion for the outer loop and the relative difference of the solution values $\epsilon_{in} = 1\mathrm{e}-3$ as the convergence criterion for the inner loop. For Krylov linear solvers in PETSc, we set the relative tolerance of convergence as $1\mathrm{e}-5$ and the absolute tolerance of convergence as $1\mathrm{e}-8$.

Algorithm 4.1. Free-boundary Grad–Shafranov solver

Choose an irregular domain Ω to reformulate the free-boundary problem (2.4) and (2.5)
Construct a level set function $\tilde{\phi}$ and prepare a cut-cell mesh
Given an initial equilibrium ψ^0, solve (4.3) for $U(R,Z)$
Calculate coil currents I_i^0 according to (4.6)
Determine boundary value ψ_b^0 according to (4.4)
`//Outer loop`
for $m = 0$ to $m_{\max}$ **do**
  `//Inner loop`
  for $n = 0$ to $n_{\max}$ **do**
    Search ψ_o and ψ_X in ψ^n
    Solve $\Delta_h^* \tilde{\psi}^{n+1} = \mathcal{S}(\psi^n)$ with the boundary condition ψ_b^m
    if n=0 **then**
      λ_{in}^0=0.3
      $\alpha_{in} = 1 - \lambda_{in}^0$
    else
      $\lambda_{in}^{n+1} = \lambda_{in}^n + (\lambda_{in}^n - 1)\frac{(D\psi^n - D\psi^{n+1})^T D\psi^{n+1}}{\|D\psi^n - D\psi^{n+1}\|^2}$ with $D\psi^{n+1} = \psi^n - \tilde{\psi}^{n+1}$
      $\alpha_{in} = 1 - \max(\min(\lambda_{in}^{n+1}, \lambda_{\max}), \lambda_{\min})$
    end if
    $\psi^{n+1} = (1 - \alpha_{in})\psi^n + \alpha_{in}\tilde{\psi}^{n+1}$
    Check convergence by $\Delta\psi = \frac{\|\psi^{n+1} - \psi^n\|_{\infty,\Omega}}{\|\psi^0\|_{\infty,\Omega}}$
    if $(\Delta\psi < \epsilon_{in})$ **then**
      break
    end if
  end for
  Given ψ^{n+1}, solve (4.3) for $U(R,Z)$
  Update coil currents I_i^m according to (4.6)
  Determine boundary value $\tilde{\psi}_b^m$ according to (4.4)
  if m=0 **then**
    λ_{out}^0=0.3
    $\alpha_{out} = 1 - \lambda_{out}^0$
  else
    $\lambda_{out}^{m+1} = \lambda_{out}^m + (\lambda_{out}^m - 1)\frac{(D\psi_b^m - D\psi_b^{m+1})^T D\psi_b^{m+1}}{\|D\psi_b^m - D\psi_b^{m+1}\|^2}$ with $D\psi_b^{m+1} = \psi_b^m - \tilde{\psi}_b^{m+1}$
    $\alpha_{out} = 1 - \max(\min(\lambda_{out}^{m+1}, \lambda_{\max}), \lambda_{\min})$
  end if
  $\psi_b^{m+1} = (1 - \alpha_{out})\psi_b^m + \alpha_{out}\tilde{\psi}_b^{m+1}$
  Check convergence by $\Delta\psi_b = \frac{\|\psi_b^{m+1} - \psi_b^m\|_{\infty,\partial\Omega}}{\|\psi_b^0\|_{\infty,\partial\Omega}}$
  if $\Delta\psi_b < \epsilon_{out}$ **then**
    break
  end if
end for

5. Parallel implementation. Some aspects of the implementation of the free-boundary Grad–Shafranov solver with the cut-cell algorithm will be discussed in this section. The free boundary Grad–Shafranov solver described in this work is implemented in parallel under the PETSc framework [3] using a standard domain-decomposition approach. All the vectors, arrays, and matrices use the parallel distributed data structure provided by PETSc and its communication between subdomains

is based on the message-passing interface (MPI). All the linear and nonlinear solvers described in this work are implemented through PETSc and the elliptic operator, Δ_h^*, is preconditioned with AMG preconditioners to improve its efficiency and scalability.

The data structure is based on a Cartesian structured mesh (DMDA) provided by PETSc. The solution is stored as grid points while its control volume is straightforwardly implied. This is a common approach to adapt a finite volume algorithm in a finite difference code; see [10], for instance. To develop a cut-cell algorithm based on DMDA, all the grid points are distinguished into three categories of active points, inactive points, and cut-cell points, based on its underlying control volume. If the control volume has the full cell size, then it is an active point; otherwise it is either a cut-cell point (if the volume fraction is between 0 and 1) or an inactive point (if the volume fraction is 0). Note that the approach distinguishing active and inactive grid points is commonly used in overlapping grids, where some of the grid points are not involved in the discretization; see [4, 5], for instance. Since the scheme is based on a five-point stencil, the discrete operator on the active points is well defined, and meanwhile a redundant equation is imposed on all the inactive points when discretizing the elliptic operator on those points (simply solving $a\psi_{\text{inactive}} = 0$ where a is a large constant). The main difference between the free-boundary solvers on a standard Cartesian mesh and on cut cells thus lies in the additional treatment related to the cut-cell points. Involving a redundant equation into the discretization leads to an operator of the same size as the standard operator on a Cartesian grid. All the approaches described above guarantee many PETSc functions for DMDA can be directly used in the cut-cell algorithm, which eases some data structure allocations and implementations of linear or nonlinear operators. It also provides a rather straightforward path for the future improvement through geometry multigrid.

To further ease the treatment related to cut-cells, several two-dimensional (2D) arrays are precomputed to store the geometry information and the information related to level set function. For example, volume fractions and area fractions of each cell, apertures of each edge, and midpoint values of each cut edge and interpolation points are precomputed before performing the Picard iterations, which greatly improves the solver efficiency.

There are some details associated with the main algorithm worth mentioning. The search algorithm in parallel needs first to identify candidate points through performing a search on the subdomain and then gathers all the candidate points into the root processor to finalize the locations of ψ_x and ψ_o. Some MPI collective communications are necessary to successfully locate those points. Another important aspect is to perform an efficient line integral along the boundary. The efficiency of computing the line integral can be improved by precomputing certain weights associated with the Green's function and the quadratures. For instance, the terms changing in (4.4) in each iteration are $\frac{\partial U}{\partial n}$ and I_i, and we therefore precompute the weights associated with $\frac{dl}{R}G(R, Z; R', Z')$ and $\mu_0 G(R_i^c, Z_i^c; R', Z')$ and store them in two distributed dense matrices. Parallel matrix-vector operations are called to evaluate the boundary values ψ_b efficiently during the iterations.

Finally, to verify the performance of the full free-boundary Grad–Shafranov solver, a strong scaling study is performed. In the study, a cut-cell mesh based on a Cartesian mesh of size 512×1024 is used. We verify the performance of the solver using different numbers of processors up to 32. Figure 4 shows the strong scaling result and a good scaling is observed. The corresponding parallel efficiency for 2, 4, 8, 16, and 32

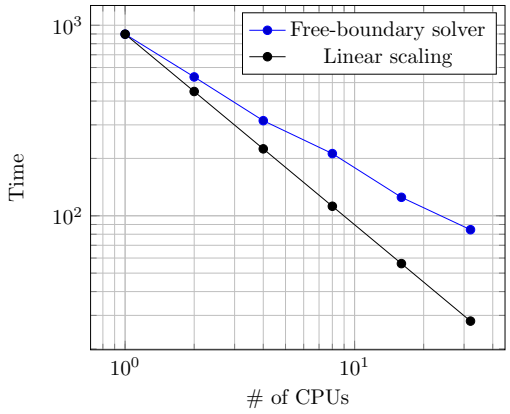


FIG. 4. *Strong scaling result of the free-boundary cut-cell Grad–Shafranov solver. The computational times of one and up to* 32 *CPUs are presented. A good parallel scaling up to* 32 *processors is observed. A cut-cell mesh based on a Cartesian mesh of* 512×1024 *is used in the strong scaling study. Details of the problem setup can be found in the last free-boundary example in the numerical section.*

processors is 83.9%, 71.1%, 52.9%, 44.9%, and 33.2%. Note that since the algorithm is based on Picard iteration fundamentally, we do not expect such an algorithm would be scalable when the number of processors increases to some large number. Nevertheless, the performance of the algorithm is decent and 32 processors are likely sufficient for a 2D steady-state problem. In the scaling study, the full algorithm described in Algorithm 4.1 is used and all the elliptic operators use an AMG preconditioner provided by PETSc. Details of the problem setup can be found in the free-boundary cut-cell example in the numerical section later.

6. Numerical results. The performance of the cut-cell algorithm, the free-boundary Grad–Shafranov solver, and Aitken's acceleration is demonstrated through several numerical tests. We start with the accuracy test of the cut-cell solver through two examples: the Grad–Shafranov equation with a linear source term and the Grad–Shafranov equation with a nonlinear source term. Numerical examples of the free-boundary problem in a rectangular domain and in the limiter-bounded domain $\mathcal{L}$ are presented separately to show the performance of the free-boundary Grad–Shafranov solver and cut-cell algorithm. Finally, we compare the performance of the Picard iteration with different underrelaxation coefficients and Aitken's acceleration.

6.1. Convergence study for the fixed-boundary cut-cell solver. In this section, two numerical examples are presented to demonstrate the accuracy of the cut-cell algorithm. Here we use two fixed-boundary problems with cut-cells to verify its accuracy. These cases are chosen for the known analytical solutions. Note that compared with a free-boundary solver, the fixed-boundary solver does not have the surface integral and the barycentric interpolation, both of which can be verified easily through standalone tests.

In the first example, we consider the linear Soloviev profiles from [11]. This test is used to compare the accuracy of the solution ψ solved by our fixed-boundary Grad–Shafranov solver and the analytical solution. The second example considers a manufactured solution with a nonlinear source. We use it to demonstrate the accuracy of our scheme in nonlinear problems.

6.1.1. Linear case. Consider a linear Grad–Shafranov equation of

$$\Delta^*\psi = -\frac{1}{R}\frac{\partial\psi}{\partial R} + \frac{\partial^2\psi}{\partial R^2} + \frac{\partial^2\psi}{\partial Z^2} = R^2,$$

which has an exact solution in the form of

$$\psi(R,Z) = \frac{R^4}{8} + D_1 + D_2R^2 + D_3(R^4 - 4R^2Z^2),$$

where the parameters D_1, D_2, and D_3 are determined so that the contour $\phi = 0$ represents a reasonable plasma cross section.

Reference [42] introduced three characteristic quantities describing the shape of the cross section in a magnetic confinement device: the inverse aspect ratio ε, the elongation κ, and the triangularity δ. We adopt the same manufactured solutions in the accuracy test. The parameters are determined by the linear system

$$\begin{bmatrix} 1 & (1+\varepsilon)^2 & (1+\varepsilon)^4 \\ 1 & (1-\varepsilon)^2 & (1-\varepsilon)^4 \\ 1 & (1-\delta\varepsilon)^2 & (1-\delta\varepsilon)^4 - 4(1-\delta\varepsilon)^2\kappa^2\varepsilon^2 \end{bmatrix} \begin{bmatrix} D_1 \\ D_2 \\ D_3 \end{bmatrix} = -\frac{1}{8}\begin{bmatrix} (1+\varepsilon)^4 \\ (1-\varepsilon)^4 \\ (1-\delta\varepsilon)^4 \end{bmatrix},$$

of which the equations correspond to the boundary conditions of $\psi(1+\varepsilon, 0) = 0$, $\psi(1-\varepsilon, 0) = 0$, and $\psi(1-\delta\varepsilon, \kappa\varepsilon) = 0$, respectively. The test is taken with the ITER-like configuration of $\varepsilon = 0.32$, $\kappa = 1.7$, $\delta = 0.33$. In particular, the computational boundaries are described by Chebyshev nodes along the R direction and Z coordinates are then determined by solving $\psi = 0$. The cut-cell meshes are then accordingly generated.

Numerical errors and corresponding convergence rates are reported in Table 1. The numerical solution and an example cut-cell mesh are presented in Figure 5. We observed a second-order accuracy for all the error norms.

6.1.2. Nonlinear case. We consider the same ITER geometry as for the linear test in the previous section but with a nonlinear source term. A manufactured solution

$$\psi(R,Z) = \sin(K_R(R+R_0))\cos(K_ZZ)$$

can be constructed for the nonlinear Grad–Shafranov equation of

$$\Delta^*\psi = -F(R,Z,\psi),$$

where the source term $F(R,Z,\psi)$ is given by

TABLE 1

L_1-errors, L_2-errors, L_∞-errors, and corresponding convergence rates of ψ in the linear accuracy test using the cut-cell mesh. The grid points in the meshes represent all the grid points in a Cartesian grid including both active and inactive points.

$N_x \times N_y$	L_1-error	Order	L_2-error	Order	L_∞-error	Order
31×41	2.310e-01		1.276e-02		1.519e-03	
61×81	5.925e-02	1.96	3.361e-03	1.92	5.513e-04	1.46
121×161	1.467e-02	2.01	8.327e-04	2.01	9.177e-05	2.59
241×321	3.774e-03	1.96	2.155e-04	1.95	2.229e-05	2.04
481×641	1.086e-03	1.80	6.198e-05	1.80	5.422e-06	2.04

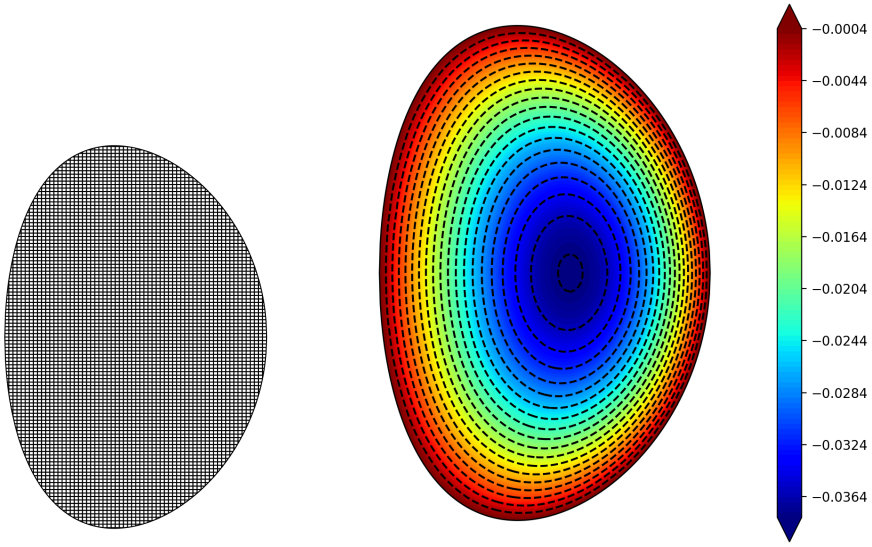


FIG. 5. *Linear accuracy test. Left: cut-cell mesh on a base mesh of* 121×161 *(only active points are shown) and geometry. Right: corresponding numerical solution* ψ.

TABLE 2

L_1-errors, L_2-errors, L_∞-errors, and corresponding convergence rates of ψ in the nonlinear accuracy test using cut-cell algorithm. The grid points in the meshes represent all the grid points in a Cartesian grid including both active and inactive points.

$N_x \times N_y$	L_1-error	Order	L_2-error	Order	L_∞-error	Order
31×41	1.486e+00		8.493e-02		1.359e-02	
61×81	4.476e-01	1.73	2.584e-02	1.72	4.480e-03	1.60
121×161	1.340e-01	1.74	7.688e-03	1.75	8.288e-04	2.43
241×321	3.371e-02	1.99	1.912e-03	2.01	2.154e-04	1.94
481×641	7.825e-03	2.11	4.429e-04	2.11	4.148e-05	2.38

$$F(R,Z,\psi) = (K_R^2 + K_Z^2)\psi + \frac{K_R}{R}\cos(K_R(R+R_0))\cos(K_Z Z) + R\Big[\sin^2(K_R(R+R_0))\cos^2(K_Z Z) - \psi^2 + \exp(-\sin(K_R(R+R_0))\cos(K_Z Z)) - \exp(-\psi)\Big],$$

and the coefficients are $K_R = 1.15\pi$, $K_Z = 1.15$, and $R_0 = -0.5$.

A nonlinear solver (Picard iteration accelerated with Aitken's) is used in this example. Numerical errors and corresponding convergence rates are reported in Table 2. The solution and an example cut-cell mesh are presented in Figure 6. Again, as expected, we observe a second-order accuracy of space discretization.

6.2. Examples of the free-boundary solver. In this section, numerical solutions from the free-boundary Grad–Shafranov solver are presented. One particular interest will be the performance of the solver in keeping the targeted shape of $\mathcal{P}(\psi)$, i.e., the shape control. Numerical tests are scattered in the following two examples, a rectangular domain and a limiter-bounded domain.

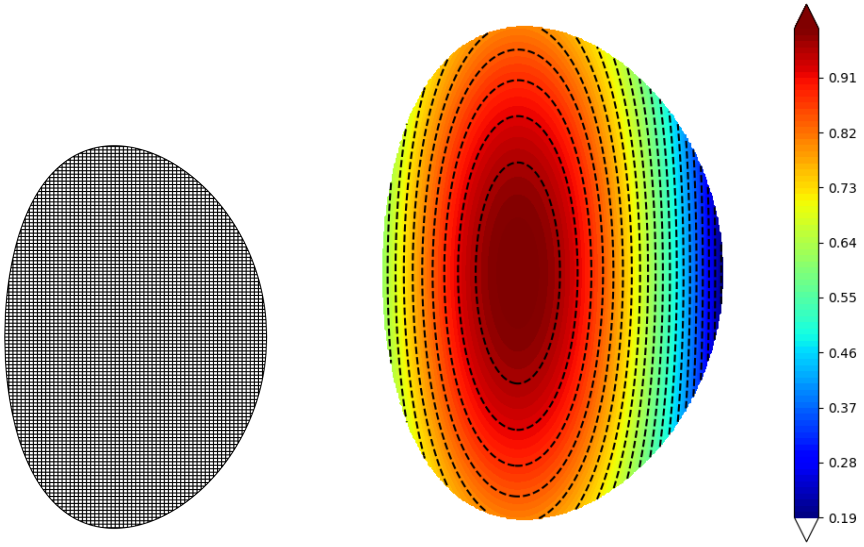
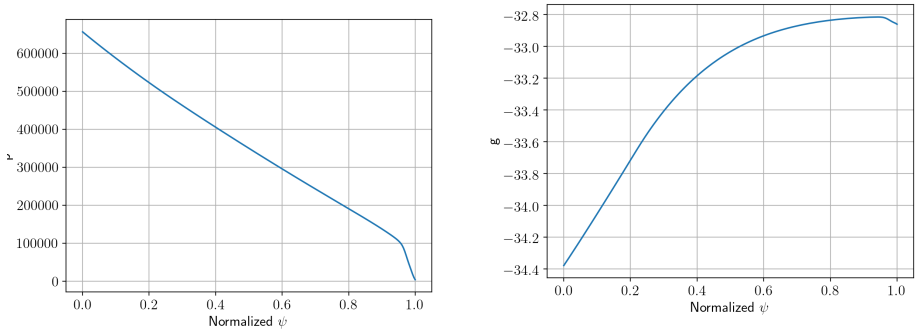


FIG. 6. *Nonlinear accuracy test. Left: cut-cell mesh on a base mesh of* 121×161 *(only active points are shown) and geometry. Right: corresponding numerical solution* ψ.



(a) Pressure (p) profile of normalized ψ

(b) Poloidal current (g) profile of normalized ψ

FIG. 7. *Numerical* $p(\bar{\psi})$ *and* $g(\bar{\psi})$ *profiles from the equilibrium data generated in* [39].

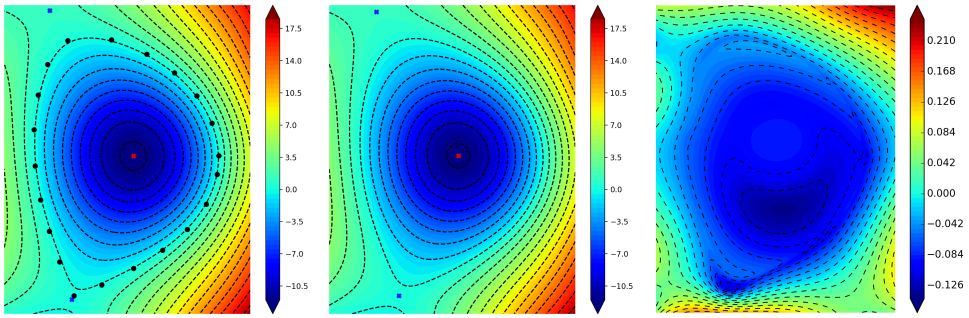
The problem is based on prescribed numerical $p(\bar{\psi})$ and $g(\bar{\psi})$ profiles, given in Figure 7, for a proposed ITER discharge at 15 MA toroidal plasma current [39], which carries the ITER reference number ABT4ZL. In addition to the numerical $p(\bar{\psi})$ and $g(\bar{\psi})$ profiles, there is also the numerical solution for $\psi(R, Z)$ from a different Grad–Shafranov solver [39], which we will use for the initial data. The coil currents corresponding to the equilibrium were not given and they will be found by our free-boundary Grad–Shafranov solver in this test. The ITER poloidal field coil locations, however, are fixed and given in Table 3. We will solve or, more accurately, resolve this free-boundary Grad–Shafranov equilibrium problem in two ways. One is based on a Cartesian mesh over a rectangular domain and the other one is based on a cut-cell mesh that has the first wall as the computational boundary.

TABLE 3

Coil information for the free-boundary problem. The data is based on the ITER configurations. The total current in the coil is set to current timing the value of turns. The type "coil" refers to a point source. The type "solenoid" refers to a coil with a positive length for which the range of length is also given. The unit is meters.

Coils	Type	R	Z	Turns
PF1	Coil	3.9431	7.5741	248.6
PF2	Coil	8.2851	6.5398	115.2
PF3	Coil	11.9919	3.2752	185.9
PF4	Coil	11.9630	-2.2336	169.9
PF5	Coil	8.3908	-6.7269	216.8
PF6	Coil	4.3340	-7.4665	459.4

Coils	Type	R	(Z_{min}, Z_{max})	Turns
CS1	Solenoid	1.696	(-5.415, -3.6067)	553
CS2	Solenoid	1.696	(-3.6067, -1.7983)	553
CS3	Solenoid	1.696	(-1.7983, 1.8183)	1106
CS4	Solenoid	1.696	(1.8183, 3.6267)	553
CS5	Solenoid	1.696	(3.6267, 5.435)	553

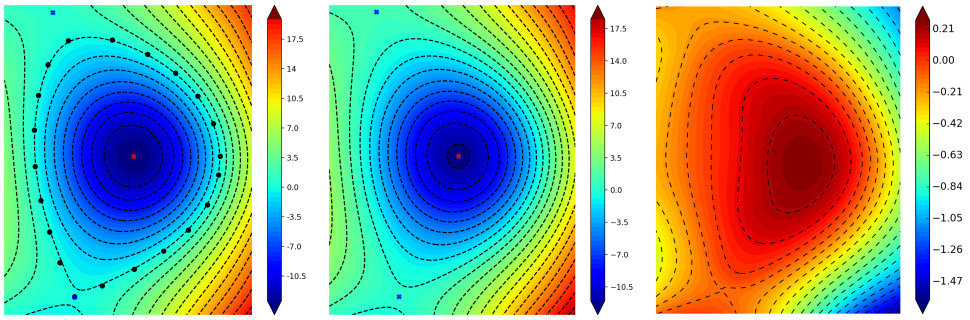


(a) Converged ψ (b) ψ^0 from initial data (c) $\psi - \psi^0$

FIG. 8. *Rectangular geometry:* (a) *converged magnetic flux from free-boundary Grad–Shafranov solver,* (b) *magnetic flux from initial equilibrium data file,* (c) *the difference between magnetic flux from free-boundary Grad–Shafranov solver and initial equilibrium data.*

6.2.1. Rectangular domain. The first example uses a rectangular computational domain that contains the limiter regions but excludes the current regions. This is a conventional approach implemented in many practical Grad–Shafranov solvers. We present this case to verify our implementation of the full solver as well as to compare the results with the solutions from the cut-cell solver, which will be discussed in the next section.

The computational domain is $\Omega = \{(R, Z) \in \mathcal{H} \,|\, 3.55 \leq R \leq 8.88, -3.84 \leq Z \leq 4.92\}$ and the grid used in this case is a Cartesian grid of size 196×375. The proposed free-boundary Grad–Shafranov solver is used to solve the problem on the rectangular domain Ω. The converged magnetic flux from the free-boundary Grad–Shafranov solver and magnetic flux from the equilibrium data file are presented in Figure 8. Black dots in Figure 8(a) represent the shape control points on the targeted magnetic separatrix of the fixed plasma shape from the initial data. Twenty-one points are selected along $\psi = \psi_X$ in the initial data as the control points. We can observe from Figure 8(a) that the converged magnetic flux from the free-boundary Grad–Shafranov solver keeps the predetermined plasma shape very well. The solution is comparable to the initial data and its difference is presented in Figure 8(c). Note that the largest difference is around the corner points. This is expected since the computational domain is very close to two current coils at $(8.3908, -6.7269)$ and $(8.2851, 6.5398)$.



(a) Converged ψ (b) ψ^0 from initial data (c) $\psi - \psi^0$

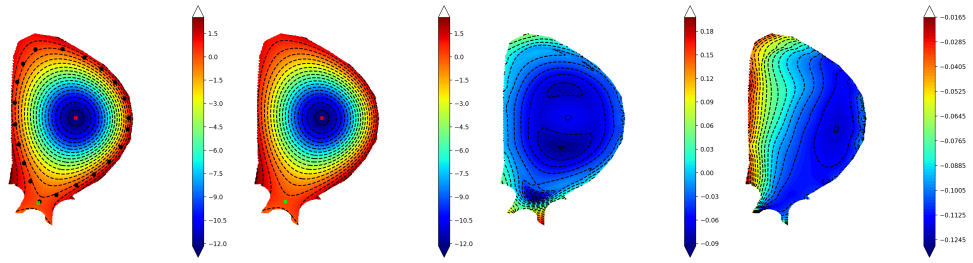
FIG. 9. *Rectangular geometry:* (a) *converged magnetic flux from free-boundary Grad–Shafranov solver,* (b) *magnetic flux from initial equilibrium data file,* (c) *the difference between magnetic flux from free-boundary Grad–Shafranov solver and the initial data. The pressure is dropped to* 80% *of the given profile.*

The magnetic axes ψ_o are presented in Figures 8(a) and 8(b) as red cross points, which are clearly local minima in the solutions. There are two candidate of ψ_X points shown in Figures 8(a) and 8(b), using blue cross points. It is clear that those are saddle points of the solutions. Based on the criteria we discussed in the algorithm section, the point in the bottom portion of the domain is chosen as ψ_X. It is seen that the selected ψ_X point in the solution is very close to the control points. Both ψ_o and ψ_X points are found to change slightly throughout the iterations.

Another interesting question to investigate is whether the converged magnetic flux from the free-boundary Grad–Shafranov solver can keep the predetermined plasma shape if we modify the source term of the Grad–Shafranov equation. This is a test to verify the effectiveness of the shape control techniques we implemented in the solver. As an example, we drop the pressure to 80% of the original pressure profile and solve the same free-boundary problem using the solver. Its results are presented in Figure 9. The converged magnetic flux from the free-boundary Grad–Shafranov solver and magnetic flux from the equilibrium data file are also compared in Figure 9. It indicates that the converged magnetic flux still keeps the predetermined plasma shape. It also suggests that the free-boundary Grad–Shafranov solver performs well in keeping the targeted shape of $\mathcal{P}(\psi)$ in the rectangular domain.

6.2.2. Limiter-bounded domain $\mathcal{L}$. In the second example, by the combining cut-cell algorithm, we directly choose the irregular limiter-bounded domain $\mathcal{L}$ as Ω to solve the free boundary problem. The domain $\mathcal{L}$ consists of a bunch of straight lines due to the design of ITER, which will be used to construct the cut-cell mesh. This example is used to verify the full algorithm we proposed in this work.

We create a Cartesian mesh of size 216×493 in the domain $\{(R, Z) \in \mathcal{H} \,|\, 3.0 \leq R \leq 8.88, -5.53 \leq Z \leq 6.0\}$, which contains the irregular limiter-bounded domain $\mathcal{L}$. The cut-cell mesh is created accordingly, and the total number of active cells is 44668, which includes 1266 cut-cells. The proposed free-boundary Grad–Shafranov solver combining the cut-cell algorithm is used to solve the problem on the limiter-bounded domain $\mathcal{L}$. The converged magnetic flux from the solver and magnetic flux from the equilibrium data file are presented in Figure 10. Black dots in Figure 10(a) represent the shape control points on targeted magnetic separatrix of the fixed plasma shape from the initial data. The same 21 points in subsection 6.2.1 are selected along



(a) Converged $\psi_{\mathrm{cut-cell}}$ (b) ψ^0 from initial data (c) $\psi_{\mathrm{cut-cell}} - \psi^0$ (d) $\psi_{\mathrm{box}} - \psi_{\mathrm{cut-cell}}$

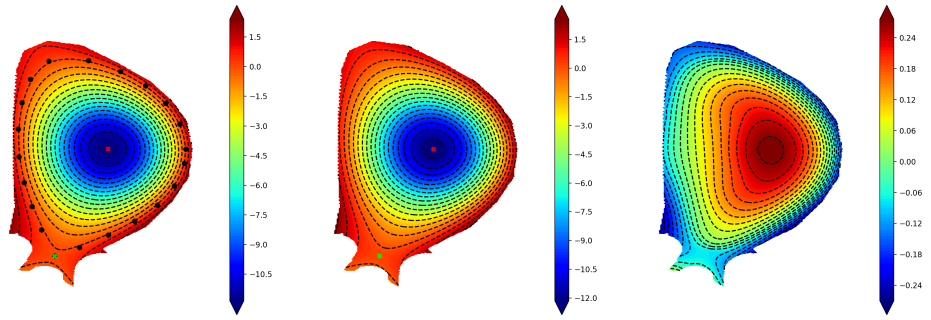
FIG. 10. *Limiter geometry:* (a) *converged magnetic flux from the free-boundary Grad–Shafranov solver with the cut-cell algorithm,* (b) *magnetic flux from the initial equilibrium data file,* (c) *difference between magnetic flux from the free-boundary Grad–Shafranov solver with the cut-cell algorithm and the initial equilibrium data file,* (d) *difference between the solutions of the rectangular domain and the limiter-bounded domain.*

$\psi = \psi_X$ in the initial data as the control points. We can observe from Figure 10(a) that the converged magnetic flux from free-boundary Grad–Shafranov solver keeps the predetermined plasma shape very well. The solution is comparable to the initial data and its difference is presented in Figure 10(c). Note that the largest difference is around the bottom corner, which is due to the nonsmooth boundary in the computational domain. The magnetic axes ψ_o are presented in Figures 10(a) and 10(b) as red cross points, which are clearly local minima in the solutions. Different from the example in the rectangular domain, only one ψ_X point is shown in Figures 10(a) and 10(b), using a green cross point. This is caused by the choice of the limiter-bounded domain $\mathcal{L}$, as the initial plasma domain is bounded by the last closed poloidal flux line inside the limiter-bounded domain. It is clear that the ψ_X point is the saddle point of the solutions. Both ψ_o and ψ_X are found to change slightly throughout the iterations.

In Figure 10(d), the converged magnetic fluxes of the free-boundary problem on the regular domain and the limiter-bounded domain are compared to each other. It is found that the difference is small, which indicates both solvers produce the same equilibrium. When solving on the rectangular domain, the outer iteration is 12, and the average inner iteration is 32, while on the limiter-bounded domain the outer iteration is 11, and the average inner iteration is 9.

In this example, we again investigate whether the converged magnetic flux from the solver can keep the predetermined plasma shape in the limiter-bounded domain $\mathcal{L}$ when we modify the source term of the Grad–Shafranov equation. Similarly, we drop the pressure to 80% of the original pressure profile and solve the same free-boundary problem using the solver combining with the cut-cell algorithm. Its results are presented in Figure 11. The converged magnetic flux from the free-boundary Grad–Shafranov solver and the magnetic flux from the equilibrium data file are also presented in Figure 11. It indicates that the converged magnetic flux still keeps the predetermined plasma shape. It also suggests that the free-boundary Grad–Shafranov solver performs well in keeping the targeted shape of $\mathcal{P}(\psi)$ in the limiter-bounded domain $\mathcal{L}$.

6.3. Performance of Aitken's acceleration. In the final examples, we focus on the demonstration of the efficiency of Aitken's acceleration. Results from Aitken's acceleration are compared to results from the Picard iteration with fixed underrelax-



(a) Converged $\psi_{\text{cut-cell}}$ (b) ψ^0 from initial data (c) $\psi_{\text{cut-cell}} - \psi^0$

FIG. 11. *Limiter geometry:* (a) *converged magnetic flux from the free-boundary Grad–Shafranov solver with the cut-cell algorithm,* (b) *magnetic flux from the initial equilibrium data file,* (c) *difference between magnetic flux from the free-boundary Grad–Shafranov solver with the cut-cell algorithm and the initial equilibrium data file. The pressure is dropped to* 80% *of the given profile.*

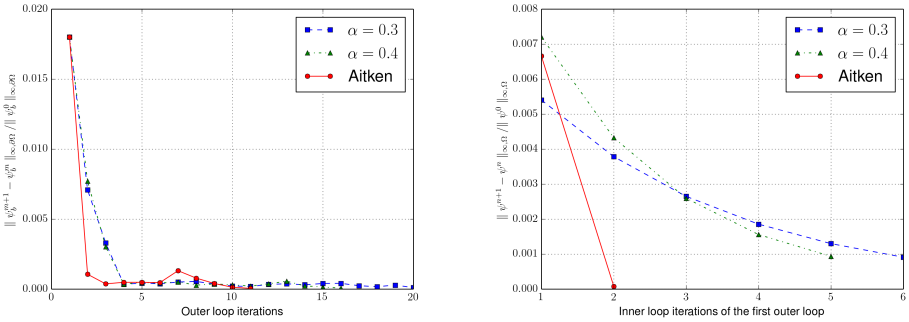


FIG. 12. *Comparison of convergence history of the free-boundary solver algorithm with Aitken's acceleration and Picard iterations. Left: convergence history of the outer loop. Right: convergence history of the first inner loop. The convergence histories of two fixed underrelaxation parameters,* $\alpha = 0.3$ *and* $\alpha = 0.4$*, are also presented. The problem is solved on a rectangular computational domain with mesh size of* 196×375*. Note that Aitken's acceleration significantly improves the convergence of Picard iterations with a predetermined underrelaxation.*

ation coefficients α in the free-boundary Grad–Shafranov solver. As presented in Algorithm 4.1, both inner and outer loops use underrelaxations, through either Aitken's acceleration or a fixed underrelaxed parameter. We consider both the rectangular geometry and the geometry with the limiter-bounded domain in the previous test. Aitken's acceleration is found to converge much faster than Picard iterations with fixed underrelaxation coefficients in both the inner and outer loops.

6.3.1. Rectangular domain. In this example, we test the efficiency of Aitken's acceleration when compared with the Picard iteration with fixed underrelaxation coefficients α on the rectangular domain with mesh size of 196×375. Here we set the range $[\lambda_{min}, \lambda_{max}] = [0, 0.95]$.

Figure 12 shows the convergence histories of the Picard iteration with different under-relaxation coefficients α and Aitken's acceleration. The results of two parameters, $\alpha = 0.3$ and $\alpha = 0.4$, are presented for comparison. The left figure shows that the free-boundary Grad–Shafranov solver algorithm with underrelaxed Picard iterations need to take 16 or 20 outer loops to satisfy the desired convergent threshold

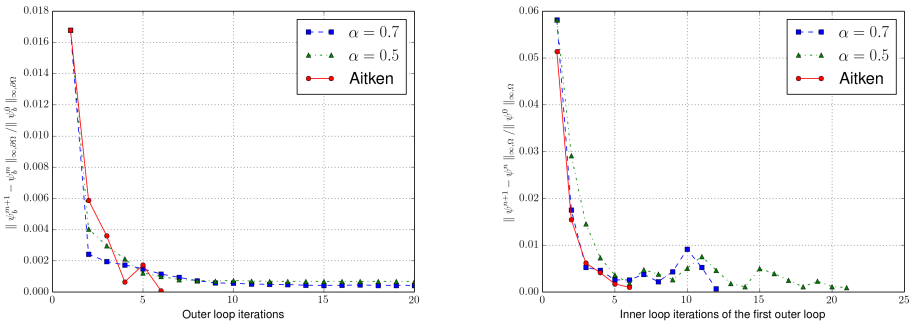


FIG. 13. *Comparison of convergence history of the free-boundary solver algorithm with Aitken's acceleration and Picard iterations. Left: convergence history of the outer loop. Right: convergence history of the first inner loop. The convergence histories of two fixed underrelaxation parameters,* $\alpha = 0.7$ *and* $\alpha = 0.5$*, are also presented. The problem is solved on the limiter-bounded domain* $\mathcal{L}$ *with a cut-cell mesh of size* 216×493 *(base Cartesian mesh). Note that Aitken's acceleration significantly improves the convergence of Picard iterations with a predetermined underrelaxation.*

for the boundary value ψ_b, while Aitken's acceleration needs only 11 iterations. On the other hand, the right figure shows the convergence histories in the first inner loop. To have a fair comparison, the first inner loop is chosen since all the first inner loops use the same initial guess. It is found that the Aitken's acceleration needs only two iterations to satisfy the the desired convergence tolerance for the magnetic flux ψ, while the underrelaxed Picard iterations need to take two to three times more iterations. Therefore, the performance of Aitken's acceleration in the inner loops is also better than that of Picard iterations.

6.3.2. Limiter-bounded domain $\mathcal{L}$. The final example focuses on the efficiency of Aitken's acceleration on the limiter-bounded domain $\mathcal{L}$ with a cut-cell mesh of size 216×493 (base Cartesian mesh). Here we set the range $[\lambda_{min}, \lambda_{max}] = [0, 0.7]$.

Figure 13 shows the convergence histories of Picard iterations with different underrelaxation coefficients α and Aitken's acceleration. The results of two parameters, $\alpha = 0.7$ and $\alpha = 0.5$, are presented for comparison. We set the the maximum iteration to be 40 for the outer loop. The underrelaxed Picard iterations fail to reach the desired tolerance before reaching the maximum iteration, while the Aitken's acceleration needs only 6 iterations to reach the same tolerance. The left figure presents the maximum relative difference of ψ_b in the first 20 iterations. The right figure shows the first inner loop, for which the Aitken's acceleration needs only 6 iterations, while the under-relaxed Picard iterations take 12 or 21 iterations.

In conclusion, we find that the Aitken's acceleration could significantly accelerate Picard iterations in both inner and outer loops. Other acceleration techniques may also show similar performance, which will be studied in future work.

7. Conclusions. This work discusses the development of a parallel free-boundary Grad–Shafranov solver. As traditionally done, the free-boundary problem is reformulated in a bounded computational domain that encloses all current-carrying plasmas. The coil current contribution to the plasma equilibrium enters through the value of the magnetic flux that is evaluated by Green's function methods. The main focus here is to solve the free-boundary problem of the nonlinear Grad–Shafranov equation in an irregular computational domain by combining the cut-cell algorithm with a regular mesh. A key application of free-bounday Grad–Shafranov solvers is to find the

required coil currents for precision plasma positioning and shaping. This is cast into a proper optimization problem that determines the coil current with targeted plasma shape and for which a cut-cell algorithm is described. To accelerate Picard iterations commonly used in previous works, we also propose an improvement based on Aitken's acceleration. This is applied to both the inner and outer loops of the algorithm. The proposed algorithm is implemented in parallel under the PETSc framework. A strong scaling study of the free-boundary Grad–Shafranov solver with the cut-cell algorithm demonstrates its parallel performance.

A series of numerical tests is presented to demonstrate the accuracy of the cut-cell algorithm and the efficiency of the free-boundary Grad–Shafranov solver. In particular, a refinement study based on manufactured solutions confirms a second-order accuracy of the cut-cell implementation, and tokamak examples, including that of ITER, are presented to verify the effectiveness and efficiency of the full algorithm. Moreover, numerical results demonstrate that the converged magnetic flux keeps the predetermined plasma domain shape even with a different pressure profile in the source term of the Grad–Shafranov equation in both the rectangular domain and the irregular limiter-bounded domain. It is also found that the Aitken's acceleration we employed in the algorithm is more efficient than the Picard iteration with fixed underrelaxation. In conclusion, numerical results shows a good performance of the free-boundary cut-cell Grad–Shafranov solver.

Acknowledgments. The authors wish to thank Dr. Yueqiang Liu of General Atomics for sharing the Grad–Shafranov equilibrium of ITER reference number ABT4ZL. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility.

REFERENCES

[1] A. Ambrosetti, M. Calahorrano, and F. Dobarro, *Remarks on the Grad-Shafranov equation*, Appl. Math. Lett., 3 (1990), pp. 9–11.

[2] M. Ariola, A. Pironti, and A. Portone, *Vertical stabilization and plasma shape control in the iter-feat tokamak*, in Proceedings of the International Conference on Control Applications, Los Alamitos, CA, 2000, IEEE Computer Society, pp. 401–405.

[3] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al., *PETSc Users Manual*, Argonne National Laboratory, 2019, https://ora.ox.ac.uk/objects/uuid:fa2b9e7c-1c58-429c-90fd-f780a3c3dc7d.

[4] J. W. Banks, W. D. Henshaw, D. W. Schwendeman, and Q. Tang, *A stable partitioned FSI algorithm for rigid bodies and incompressible flow. part* II*: General formulation*, J. Comput. Phys., 343 (2017), pp. 469–500.

[5] J. W. Banks, W. D. Henshaw, D. W. Schwendeman, and Q. Tang, *A stable partitioned fsi algorithm for rigid bodies and incompressible flow in three dimensions*, J. Comput. Phys., 373 (2018), pp. 455–492.

[6] M. Berger, *Cut cells: Meshes and solvers*, in Handbook of Numerical Methods for Hyperbolic Problems, Handb. Numer. Anal. 18, Elsevier, Amsterdam, 2017, pp. 1–22.

[7] J.-P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Comput. Math. Appl., 15 (1988), pp. 1–16.

[8] J.-P. Berrut, R. Baltensperger, and H. D. Mittelmann, *Recent developments in barycentric rational interpolation*, J. Comput. Appl. Math., 259 (2005), pp. 95–107.

[9] I. Borazjani, L. Ge, and F. Sotiropoulos, *Curvilinear immersed boundary method for simulating fluid structure interaction with complex* 3*D rigid bodies*, J. Comput. Phys., 227 (2008), pp. 7587–7620.

[10] D. L. Brown, W. D. Henshaw, and D. J. Quinlan, *Overture: An object-oriented framework for solving partial differential equations*, in International Conference on Computing in Object-Oriented Parallel Environments, Springer, New York, 1997, pp. 177–184.

[11] A. J. Cerfon and J. P. Freidberg, *"One size fits all" analytic solutions to the Grad–Shafranov equation*, Phys. Plasmas, 17 (2010), 032502.

[12] S. Chen, B. Merriman, S. Osher, and P. Smereka, *A simple level set method for solving stefan problems*, J. Comput. Phys., 135 (1997), pp. 8–29.
[13] A. J. Creely, M. J. Greenwald, S. B. Ballinger, D. Brunner, J. Canik, J. Doody, T. Fülöp, D. T. Garnier, R. Granetz, T. K. Gray, et al., *Overview of the SPARC tokamak*, J. Plasma Phys., 86 (2020), 865860502.
[14] D. Devendran, D. Graves, and H. Johansen, *A Higher-Order Finite-Volume Discretization Method for Poisson's Equation in Cut Cell Geometries*, preprint, https://arxiv.org/abs/1411.4283, 2014.
[15] D. Devendran, D. Graves, H. Johansen, and T. Ligocki, *A fourth-order cartesian grid embedded boundary method for poisson's equation*, Commun. Appl. Math. Comput. Sci., 12 (2017), pp. 51–79.
[16] B. Faugeras and H. Heumann, *FEM-BEM coupling methods for tokamak plasma axisymmetric free-boundary equilibrium computations in unbounded domains*, J. Comput. Phys., 343 (2017), pp. 201–216.
[17] M. S. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numer. Math., 107 (2007), pp. 315–331.
[18] F. Gibou and R. Fedkiw, *A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the Stefan problem*, J. Comput. Phys., 202 (2005), pp. 577–601.
[19] F. Gibou, R. P. Fedkiw, L.-T. Cheng, and M. Kang, *A second-order-accurate symmetric discretization of the poisson equation on irregular domains*, J. Comput. Phys., 176 (2002), pp. 205–227.
[20] G. H. Golub, M. Heath, and G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.
[21] H. Heumann, J. Blum, C. Boulbe, B. Faugeras, G. Selig, P. Hertout, E. Nardon, J.-M. Ané, S. Brémond, and V. Grandgirard, *Quasi-static free-boundary equilibrium of toroidal plasma with cedres++: Computational methods and applications*, J. Plasma Phys., 81 (2015), 35.
[22] H. Heumann and F. Rapetti, *A finite element method with overlapping meshes for free-boundary axisymmetric plasma equilibria in realistic geometries*, J. Comput. Phys., 334 (2017), pp. 522–540.
[23] M. Honda, *Simulation technique of free-boundary equilibrium evolution in plasma ramp-up phase*, Comput. Phys. Commun., 181 (2010), pp. 1490–1500.
[24] E. Howell and C. R. Sovinec, *Solving the grad–shafranov equation with spectral elements*, Comput. Phys. Commun., 185 (2014), pp. 1415–1421.
[25] G. Huysmans, J. Goedbloed, W. Kerner, et al., *Isoparametric bicubic hermite elements for solution of the grad-shafranov equation*, Internat. J. Modern Phys. C, 2 (1991), pp. 371–376.
[26] S. Jardin, *Computational Methods in Plasma Physics*, CRC Press, Boca Raton, FL, 2010.
[27] S. C. Jardin, N. Pomphrey, and J. Delucia, *Dynamic modeling of transport and positional control of tokamaks*, J. Comput. Phys., 66 (1986), pp. 481–507.
[28] Y. M. Jeon, *Development of a free-boundary tokamak equilibrium solver for advanced study of tokamak equilibria*, J. Korean Phys. Soc., 67 (2015), pp. 843–853.
[29] H. Johansen and P. Colella, *A cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys., 147 (1998), pp. 60–85.
[30] J. L. Johnson, H. Dalhed, J. Greene, R. Grimm, Y. Hsieh, S. Jardin, J. Manickam, M. Okabayashi, R. Storer, A. Todd, et al., *Numerical determination of axisymmetric toroidal magnetohydrodynamic equilibria*, J. Comput. Phys., 32 (1979), pp. 212–234.
[31] Z. Jomaa and C. Macaskill, *The embedded finite difference method for the poisson equation in a domain with an irregular boundary and dirichlet boundary conditions*, J. Comput. Phys., 202 (2005), pp. 488–506.
[32] K. Lackner, *Computation of ideal mhd equilibria*, Comput. Phys. Commun., 12 (1976), pp. 33–44.
[33] J. Lee and A. Cerfon, *ECOM: A fast and accurate solver for toroidal axisymmetric mhd equilibria*, Comput. Phys. Commun., 190 (2015), pp. 72–88.
[34] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Vol. 3, Springer, New York, 1992.
[35] R. J. Leveque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.
[36] H. Li and P. Zhu, *Solving the Grad-Shafranov equation using spectral elements for tokamak equilibrium with toroidal rotation*, Comput. Phys. Commun., 260 (2020), 107264.
[37] S. Liu, Y. Du, and X. Liu, *Numerical studies of a class of reaction–diffusion equations with stefan conditions*, Int. J. Comput. Math., 97 (2020), pp. 959–979.

[38] S. Liu and X. Liu, *Numerical methods for a two-species competition-diffusion model with free boundaries*, Mathematics, 6 (2018), 72.
[39] Y. Liu, R. Akers, I. Chapman, Y. Gribov, G. Hao, G. Huijsmans, A. Kirk, A. Loarte, S. Pinches, M. Reinke, et al., *Modelling toroidal rotation damping in iter due to external 3D fields*, Nuclear Fusion, 55 (2015), 063027.
[40] H. Lütjens, A. Bondeson, and O. Sauter, *The Chease code for toroidal MHD equilibria*, Comput. Phys. Commun., 97 (1996), pp. 219–260.
[41] D. Mok, W. Wall, and E. Ramm, *Accelerated iterative substructuring schemes for instationary fluid-structure interaction*, Comput. Fluid Solid Mech., 2 (2001), pp. 1325–1328.
[42] A. Pataki, A. J. Cerfon, J. P. Freidberg, L. Greengard, and M. O'Neil, *A fast, high-order solver for the Grad–Shafranov equation*, J. Comput. Phys., 243 (2013), pp. 28–45.
[43] Z. Peng, Q. Tang, and X.-Z. Tang, *An adaptive discontinuous Petrov–Galerkin method for the Grad–Shafranov equation*, SIAM J. Sci. Comput., 42 (2020), pp. B1227–B1249.
[44] A. D. Polyanin and A. V. Manzhirov, *Handbook of Mathematics for Engineers and Scientists*, CRC Press, Boca Raton, FL, 2006.
[45] T. Sánchez-Vizuet and M. E. Solano, *A hybridizable discontinuous Galerkin solver for the grad–shafranov equation*, Comput. Phys. Commun., 235 (2019), pp. 120–132.
[46] T. Sánchez-Vizuet, M. E. Solano, and A. J. Cerfon, *Adaptive hybridizable discontinuous Galerkin discretization of the Grad-Shafranov equation by extension from polygonal subdomains*, Comput. Phys. Commun., (2020), 107239.
[47] C. Schneider and W. Werner, *Some new aspects of rational interpolation*, Math. Comput., 47 (1986), pp. 285–299.
[48] P. Schwartz, M. Barad, P. Colella, and T. Ligocki, *A cartesian grid embedded boundary method for the heat equation and poisson's equation in three dimensions*, J. Comput. Phys., 211 (2006), pp. 531–550.
[49] K. von Hagenow and K. Lackner, eds., *Proceedings of the 7th Conference on the Numerical Simulation of Plasmas*, Naval Research Lab, Washington, DC, 1975.