Journal of Computational Physics 305 (2016) 744–757

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Accurate derivative evaluation for any Grad–Shafranov solver

L.F. Ricketson [a], A.J. Cerfon [a,*], M. Rachh [a], J.P. Freidberg [b]

[a] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, United States
[b] Plasma Science and Fusion Center, Massachusetts Institute of Technology, Cambridge, MA 02139, United States

ARTICLE INFO

Article history:
Received 23 July 2015
Received in revised form 30 October 2015
Accepted 7 November 2015
Available online 10 November 2015

Keywords:
Plasma
Equilibrium
Magnetic confinement fusion
Grad–Shafranov equation
Finite elements
Integral equations
Quadrature by expansion

ABSTRACT

We present a numerical scheme that can be combined with any fixed boundary finite element based Poisson or Grad–Shafranov solver to compute the first and second partial derivatives of the solution to these equations with the same order of convergence as the solution itself. At the heart of our scheme is an efficient and accurate computation of the Dirichlet to Neumann map through the evaluation of a singular volume integral and the solution to a Fredholm integral equation of the second kind. Our numerical method is particularly useful for magnetic confinement fusion simulations, since it allows the evaluation of quantities such as the magnetic field, the parallel current density and the magnetic curvature with much higher accuracy than has been previously feasible on the affordable coarse grids that are usually implemented.

1. Introduction

In computational physics, one often computes fields by expressing them in terms of a potential or stream function and then solving the resulting elliptic partial differential equation for the potential or stream function. The most common examples are problems involving electrostatic fields. The electric field is expressed in terms of the electric potential, which is then found via Poisson's equation. Another common situation is one in which the Euler equations for an incompressible fluid are expressed in terms of vorticity and a stream function: the desired velocity field is obtained from derivatives of the stream function, which is computed by solving Poisson's equation. A third important example, and the focus of this article, occurs in magnetic confinement fusion. The confining magnetic field in toroidally axisymmetric geometries is expressed in terms of a stream function associated with the magnetic flux, and the equilibrium magnetic configuration is computed by solving an elliptic PDE for this stream function known as the Grad–Shafranov equation [1,2].

When potentials and stream functions are computed using conventional finite difference or finite element schemes, numerical derivatives must be calculated to evaluate the fields. As a result, the convergence of the field solution is at least one order lower than the convergence order of the potential or stream function [3,4]. If derivatives of the fields themselves are required, at least two orders of convergence are lost. In simulations of magnetic fusion plasmas, this loss of accuracy is a particularly acute issue. Certain physical parameters that have a key role on the stability and transport properties of hot plasmas, such as the parallel current density and the magnetic curvature, depend on first derivatives of the magnetic field or equivalently second derivatives of the flux. Popular magnetic equilibrium solvers based on a finite element formulation of the Grad–Shafranov equation compute the magnetic stream function with a numerical error that decreases as N^{-4}, where N is the number of grid points in either direction of the two dimensional problem [5–9]. The numerical errors for the derivatives of the magnetic field thus only decrease as N^{-2}.

* Corresponding author.
E-mail address: cerfon@cims.nyu.edu (A.J. Cerfon).

http://dx.doi.org/10.1016/j.jcp.2015.11.015
0021-9991/

There are several ways to mitigate this loss of accuracy. One can for example use higher order finite elements, as has been effectively implemented in [4]. However, in doing so, one increases the number of degrees of freedom, which leads to a larger computational cost. One can also use a spectral representation for the solution [10] or a combined modal-Green's function representation [11]. It is indeed well known that with these methods, the loss of accuracy in the computation of derivatives is not an order of convergence, but instead a constant, as demonstrated in [11,12] for the Grad–Shafranov equation. While satisfactory from a computational point of view, a weakness of such an approach is that the solvers rely on grids that are not typically the grids necessitated by existing stability, transport, or wave heating codes that take the computed equilibrium as an input [13–16]. This is the motivation for the present work. We describe a method that combines the accuracy properties of a Green's function formulation with the advantages of relying on a popular finite element solver for the computation of the solution to the Grad–Shafranov equation. The end result is the evaluation of first and second derivatives of the flux that has the same numerical accuracy and convergence properties as the flux itself while using a physically relevant grid, without having to increase the number of grid points.

Our basic idea is to obtain linear partial differential equations for the first and second partial derivatives of the potential by analytically differentiating the original PDE (say the Grad–Shafranov equation or Poisson's equation). For instance, in the case of Poisson's equation in 2-D we see that the equations for the first derivatives are given by

$$\Delta u(x,y) = f(x,y) \implies \Delta u_x = f_x(x,y), \quad \Delta u_y = f_y(x,y). \tag{1}$$

Instead of computing the derivatives of u from the output of the finite element solver, we use that same solver to solve the linear partial differential equations for the derivatives of u.

One might then think that using the same elements as the ones used to compute u automatically leads to the same order of convergence for ∇u as for u. This is only partially true in general because one also needs the boundary condition for the normal derivative of u on the boundary of the computational domain to the same accuracy as u itself. The simplest way to accomplish this task is by taking normal numerical derivatives of u on the boundary. This, however, leads to the loss of one order in accuracy for each numerical derivative taken so that we are no better off than we were by simply taking numerical derivatives of the original solution over the whole domain. In other words, what is needed on the boundary is a method to compute the Dirichlet to Neumann map that does not lead to any loss in the order of convergence of the solution. The development of such a procedure is the main new contribution of the present work.

Specifically, we construct a numerical scheme that achieves the desired goal as follows. First, we re-express the Grad–Shafranov equation as a semi-linear Poisson equation with source function $F(\mathbf{x}, u)$. We then decompose the solution to Poisson's equation as the sum of a particular solution u^p that does not satisfy the proper Dirichlet boundary condition in general, plus a homogeneous solution u^h that solves Laplace's equation and is chosen so that the full solution satisfies the proper boundary condition. We write u^p as the volume integral of the product of $F(\mathbf{x}, u)$ with the free-space Green's function for Poisson's equation in two dimensions. By differentiating under the integral sign, we have an exact expression for the normal derivative of u^p which may be evaluated using a high-order quadrature. To compute the normal derivatives u^h without ever evaluating derivatives normal to the boundary numerically, we introduce the harmonic conjugate U of u^h and use Green's second identity to derive a Fredholm integral equation of the second kind for U on the boundary of the computational domain. After solving this integral equation for U to the same order of accuracy as u, we can compute spectrally accurate tangential derivatives of U on the boundary. Since U is the conjugate gradient of u^h, this is equivalent to computing normal derivatives of u^h with spectral accuracy, which is precisely what is needed. Note that while this work is mainly motivated by magnetic fusion applications and aimed at improving the accuracy of finite element based Grad–Shafranov solvers, our numerical method relies on a reformulation of the Grad–Shafranov equation as a Poisson problem, and can therefore also be implemented in combination with Poisson solvers.

Importantly, our approach is independent of the particular finite element solver and grid used in solving the original elliptic PDE. However, the use of the harmonic conjugate means the approach is limited to two-dimensional problems. The method also requires that accurate representations of the derivatives of $F(\mathbf{x}, u)$ be available, and that the boundary of the computational domain be smooth. These constraints are satisfied for a large number of physically relevant problems. The extension to 3-D remains a topic for future research.

The structure of the article is as follows. In Section 2, we describe in more detail the general philosophy guiding our numerical method for obtaining the same order of convergence for the partial derivatives of u as for u itself. In Section 3, we present our scheme for the accurate computation of the Dirichlet to Neumann map. In Section 4, the key elements of the algorithmic implementation of our method are discussed. We demonstrate the benefits of using our scheme in combination with an existing finite element-based Grad–Shafranov solver in Section 5, in which we focus on two magnetic confinement fusion relevant examples. We also test our numerical method for the Poisson–Boltzmann equation on an ellipse-like domain. Finally, we summarize our work and suggest ideas for further development in Section 6.

2. Setup and general idea

The Grad–Shafranov equation is given by [1,2]

$$r\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial \psi}{\partial r}\right) + \frac{\partial^2 \psi}{\partial z^2} = -\mu_0 r^2 \frac{dp}{d\psi} - \frac{1}{2}\frac{dI^2}{d\psi}, \tag{2}$$

where ψ is the magnetic flux function, $p(\psi)$ the plasma pressure, and $I(\psi)$ is the vertical current through a loop with constant ψ. The components of the magnetic field can be recovered from ψ as follows:

$$B_r = -\frac{1}{r}\frac{\partial \psi}{\partial z}, \quad B_\varphi = \frac{I(\psi)}{r}, \quad B_z = \frac{1}{r}\frac{\partial \psi}{\partial r}. \tag{3}$$

In general, the right-hand side of Eq. (2) is a nonlinear function of ψ, so that the equation has to be solved iteratively [11]. Without any penalty in terms of computational cost, one may simplify the left-hand side by the change of variables

$$u := \frac{\psi}{\sqrt{r}}. \tag{4}$$

This converts the Grad–Shafranov equation into the following semi-linear Poisson equation:

$$\Delta u = -\mu_0 r \frac{dp(\sqrt{r}u)}{du} - \frac{1}{2r}\frac{dI^2(\sqrt{r}u)}{du} + \frac{3}{4}\frac{u}{r^2} := F(\mathbf{x}, u) \tag{5}$$

where Δ denotes the cartesian-coordinate Laplacian – i.e.

$$\Delta = \frac{\partial^2}{\partial r^2} + \frac{\partial^2}{\partial z^2} \tag{6}$$

and $\mathbf{x} := (r, z) \in \mathbb{R}^2$. It is worth noting that the transformation (4) is only advantageous in regions bounded away from the $r = 0$ axis. Fortunately, this requirement holds uniformly for domains corresponding to a large class of magnetic confinement fusion applications, tokamaks in particular.

We wish to solve equation (5) in a smooth, bounded domain D, subject to homogeneous Dirichlet boundary conditions – i.e. $u = 0$ on ∂D (which is equivalent to $\psi = 0$ on ∂D). To emphasize the generality of our approach, we de-emphasize the fusion specific variables and accordingly relabel the coordinates from $(r, z) \to (x, y)$. A typical approach to solving (5) is to use fixed-point iteration, computing u^k – the approximation of u at the kth iteration – by

$$\Delta u^k = F\left(\mathbf{x}, u^{k-1}\right), \tag{7}$$

with some specified initial guess u^0. One stops the iteration when

$$\left\| \Delta u^n - F\left(\mathbf{x}, u^{n-1}\right) \right\| \leq \varepsilon, \tag{8}$$

for some reasonable norm and specified error tolerance ε. When $F(\mathbf{x}, u)$ is proportional to u, a modified version of the well-known inverse iteration method [17] may be used to avoid the trivial solution $u := 0$ [11,18]. Regardless of the iterative procedure, at each step, the Poisson equation (7) has to be solved for u^n. While there are many techniques to solve (7) the derivatives procedure introduced in this paper is independent of the specific technique chosen. Stated differently our aim is to present a method for the accurate computation of derivatives that can be used with *any* existing finite element-based Poisson or Grad–Shafranov code. Even so, in our numerical tests, we must choose a particular technique in order to obtain results. We use the isoparametric, bicubic Hermite finite element formulation used for the Grad–Shafranov equation in, for example, [5–9]. The method gives fourth order accuracy in u (and thus ψ) and, as expected, third and second order accuracy in the first and second derivatives, respectively. We shall describe the chosen FEM scheme in sufficient detail in section 4 to understand the structure of the algorithm and the numerical results. We refer the reader interested in more details to the sources referenced above.

The fundamental idea behind our technique is very simple. Say we wish to compute u_x. We differentiate (5) to get

$$\Delta u_x - F_u(\mathbf{x}, u)u_x = F_x(\mathbf{x}, u). \tag{9}$$

If the original numerical solution u is known, then this is a linear elliptic PDE for u_x, to which the same finite element method may be applied. In this way we in principle achieve the same order of accuracy as we did for u. This process may be repeated by further differentiation of the starting equation, at which point another linear elliptic PDE for the second derivative may be solved to the same order of accuracy. Moreover, the original equation (5) was nonlinear, and thus required iteration over many FEM solves. In contrast, the equations for the derivatives are linear, requiring only one FEM solve. The computation of the derivatives is thus a small additional burden relative to the original computation.

The one critical and computationally challenging point missing from the above discussion is boundary data. In order to solve (9), we need advance knowledge of u_x on ∂D. Of course, knowing u to fourth order, we can easily use finite differences to compute u_x on the boundary to third order, but this will lead to third order errors everywhere in the domain as a consequence of the trace theorem – see e.g. p. 272 in [19]. We are back where we started – we know u to order k, but can only find u_x to order $k-1$. However, there is a key difference now in that the function we need to differentiate is restricted to the boundary. Note that on the boundary, u, u_x, or any higher derivatives, are smooth, periodic functions of a single surface variable. It is thus an ideal target for *spectral* differentiation, i.e. the mode by mode differentiation of the Fourier representation of the functions on the boundary. If we can formulate the problem so that the only numerical derivatives required are spectral derivatives, we can find u_x on the boundary to the same order of accuracy as u [20]. Admittedly, one

may lose a constant scale factor in the accuracy of u_x relative to the accuracy of u, since spectral differentiation inherently increases the size of high frequency modes, but this is far preferable compared to a decrease in order of accuracy.

In the following section, we demonstrate that through the use of integral equations and a clever change of variables, it is indeed possible to compute any derivatives of u on ∂D to the same order of accuracy as u itself while taking only spectral derivatives of periodic functions.

3. Boundary data for derivatives

To begin, it is convenient to decompose ∇u on the boundary into components u_n and u_t, which are normal and tangent to ∂D, respectively. Clearly, knowledge of the shape of ∂D allows u_x and u_y to be computed given u_n and u_t. Throughout the article, we will use as our convention the *inward* normal direction and the *counter-clockwise* tangential direction unless otherwise noted.

The tangential derivative is straightforward. Direct spectral differentiation of u restricted to the boundary yields u_t to the same order of accuracy as u. In our particular case, of course, $u_t \equiv 0$ is trivial to compute since we consider homogeneous Dirichlet boundary data, but this will not be the case for second derivatives. The challenge lies in computing u_n. Intuitively, the difficulty comes from the fact that u_n is sensitive to the behavior of u *off* the boundary, while u_t is not. This intuition is manifested mathematically by the fact that Green's second identity gives

$$\int_{\partial D} G(\mathbf{x}, \mathbf{x}') u_n(\mathbf{x}')\, dl' = \frac{1}{2} u(\mathbf{x}) - \int_D G(\mathbf{x}, \mathbf{x}') F(\mathbf{x}, u(\mathbf{x}'))\, d\mathbf{x}' \tag{10}$$

for any point $\mathbf{x} \in \partial D$, where $G(\mathbf{x}, \mathbf{x}') = \log(\|\mathbf{x} - \mathbf{x}'\|)/2\pi$ is the free space Green's function of the Laplace operator. While this does give an integral equation for u_n in terms of known quantities, as desired, it is an integral equation of the first kind – the unknown only appears inside an integral – which is ill-conditioned. Attempting to solve equation (10) for u_n numerically is thus ill-advised at best, and in practice leads to very poor accuracy.

What is needed is a transformation that converts normal derivatives to tangential derivatives, since the tangential derivatives are straightforward to compute spectrally. For any harmonic function ϕ (i.e. $\Delta\phi = 0$), there is just such a transformation: the harmonic conjugate of ϕ, which we denote by Φ. It is defined by

$$\nabla^\perp \Phi = \nabla \phi, \tag{11}$$

where $\nabla^\perp = (-\partial_y, \partial_x)^T$. Clearly, $\phi_n = \Phi_t$, $\phi_t = -\Phi_n$ and $\Delta\phi = \Delta\Phi = 0$. The fact that ϕ is harmonic is essential to this transformation: if $\Delta\phi \neq 0$, then the mixed partials of Φ cannot be equal, indicating the nonexistence of such a Φ.

Unfortunately, u is not harmonic. However, it is easily decomposed into harmonic and anharmonic parts. We write $u = u^p + u^h$, where u^p is given by

$$u^p(\mathbf{x}) = \int_D G(\mathbf{x}, \mathbf{x}') F(\mathbf{x}', u(\mathbf{x}'))\, d\mathbf{x}'. \tag{12}$$

Here again $G(\mathbf{x}, \mathbf{x}') = \log(\|\mathbf{x} - \mathbf{x}'\|)/2\pi$ is the free-space Green's function of the Laplace operator. We often abbreviate this G when the arguments can be understood. The function u^p is the anharmonic contribution to the total u. Meanwhile, the harmonic contribution u^h corresponds to the homogeneous solution and satisfies

$$\Delta u^h = 0, \qquad u^h\big|_{\partial D} = -u^p\big|_{\partial D}. \tag{13}$$

We can now separate the problem into the computation u_n^p and u_n^h. Importantly, u^h is harmonic, so it has a harmonic conjugate U^h satisfying $U_t^h = u_n^h$. The strategy is thus to compute U^h on ∂D and to take its spectral tangential derivative to find u_n^h. To find u_n^p, we differentiate (12) analytically and evaluate the resulting integral numerically. Note finally that Eq. (13) is relatively simple because u has homogeneous boundary data. In general, the boundary condition on u^h would be $u^h\big|_{\partial D} = u|_{\partial D} - u^p\big|_{\partial D}$. This general form will be necessary when computing higher order derivatives.

The following two subsections lay out the details of computing u_n^p and U^h (from which u_n^h is recovered). Since evaluating u_t^p – which can be done concurrently with evaluation of u_n^p – turns out to be a key ingredient in finding U^h, we first describe the procedure for u_n^p. In all that follows, the goal is to find values of u_n^p and u_n^h at points $\mathbf{x}_j \in \partial D$ that are evenly spaced in the arc-length variable s. The details of computing the arc-length grid are discussed in Appendix A.

3.1. Computing u_n^p

As just mentioned, we will eventually need u_t^p as well as u_n^p. We evaluate both from the exact formula

$$\nabla u^p = \int_D \nabla G\, F(\mathbf{x}', u(\mathbf{x}'))\, d\mathbf{x}', \tag{14}$$

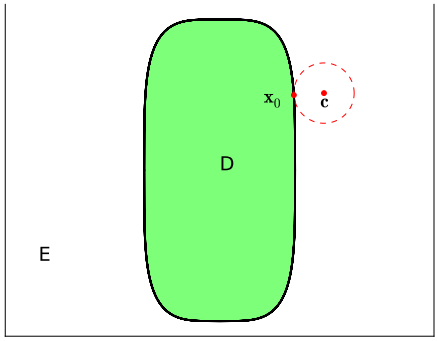

Fig. 1. Domain D and its exterior E. $\mathbf{c}$ is an *expansion center* corresponding to $\mathbf{x}_0$ in the exterior.

where

$$\nabla G(\mathbf{x}, \mathbf{x}') = \frac{1}{2\pi} \frac{\mathbf{x} - \mathbf{x}'}{\|\mathbf{x} - \mathbf{x}'\|^2}. \tag{15}$$

A major advantage of the Green's function formulation is that we have an exact integral representation for the partial derivatives of u^p given by Eq. (14). This formulation, however, comes with two well known computational challenges: first, a robust high order quadrature rule must be used to evaluate the singular integrals; second, the quadrature can be potentially costly in terms of computational time since the integrals are over a two-dimensional domain. Our numerical method addresses both challenges by relying on a variant of the Quadrature By Expansion (QBX) quadrature scheme for the singular integrals [21] accelerated by the Fast Multipole Method (FMM) [22].

Specifically, let us fix our attention to evaluating ∇u_p at $\mathbf{x}_0 \in \partial D$, denoting the outward normal at $\mathbf{x}_0$ by $\mathbf{n}_0$. Let $E = \mathbb{R}^2 \setminus D$ be the exterior of D. We observe that both the components of ∇u_p are harmonic in E and hence define smooth functions there. Using a smooth quadrature rule, we can evaluate ∇u_p and its derivatives accurately at $\mathbf{c} = \mathbf{x}_0 + r\mathbf{n}_0$, with r a constant such that $r = O(h)$ and h is the local spacing of discretization points on the boundary (see Fig. 1). To make this more precise, let $\mathbf{x} = (x, y)$, $\mathbf{x}' = (x', y')$, $\mathbf{c} = (c_x, c_y)$, $z = x + iy$, $z' = x' + iy'$ and $c = c_x + ic_y$. We observe that $w = \partial_x u^p - i\partial_y u^p$ is complex analytic in E, since the real and imaginary parts are harmonic and complex conjugates of each other. Using equation (14), we get,

$$w = \frac{1}{2\pi} \int_D \frac{F(z', u(z'))}{z - z'} \, d\mathbf{x}'. \tag{16}$$

Since w is complex analytic in E, we can form a Taylor expansion about c, to obtain the following representation for w,

$$w = \frac{1}{2\pi} \sum_{j=0}^{\infty} (c - z)^j \int_D \frac{F(z', u(z'))}{(c - z')^{j+1}} \, d\mathbf{x}'. \tag{17}$$

It can be shown that a pth order truncated expansion of the above equation, given by

$$\tilde{w} = \frac{1}{2\pi} \sum_{j=0}^{p} (c - z)^j \int_D \frac{F(z', u(z'))}{(c - z')^{j+1}} \, d\mathbf{x}', \tag{18}$$

is a high order approximation to w, even at $\mathbf{x}_0$. More precisely,

$$\left| \tilde{w}(\mathbf{x}_0) - w(\mathbf{x}_0) \right| \leq C(p, D) \, r^{p+1}. \tag{19}$$

We refer the reader to [21,23] for a detailed discussion. To evaluate $\tilde{w}$, we still need to compute the integrals,

$$\int_D \frac{F(z', u(z'))}{(c - z')^{j+1}} \, d\mathbf{x}', \tag{20}$$

which can be done to high order using a smooth quadrature rule, since the integrand is now smooth. We choose 4th order tensor-product Gauss–Legendre quadrature on each interior grid cell – and 8th order on each boundary cell – in mapped polar-like coordinates (see (30)).

The computational cost of a naive implementation of the above scheme would be $O(N_b \cdot N_v)$, where N_b is the number of points on the boundary where we wish to evaluate ∇u^p and N_v is the number of volume points used to discretize the integrands in equation (20). However, the above computation can be accelerated by the FMM as discussed in [24,25] to reduce the computational cost to $O(N_b + N_v)$, which is of comparable complexity to the other steps in our method. For readers who are unfamiliar with the FMM and interested in the details of the method, we recommend the freely accessible introductory presentation in reference [26].

3.2. Computing U^h

Recall that to evaluate u_n^h, our approach is to compute U^h, the harmonic conjugate of u^h. Since U^h is harmonic by construction, applying Green's second identity to it gives

$$\frac{1}{2}U^h(\mathbf{x}) = \int_{\partial D} \left(G U_n^h(\mathbf{x}') - G_n U^h(\mathbf{x}') \right) dl'. \tag{21}$$

By rearranging and noting that $U_n^h = u_t^h = -u_t^p$ on the boundary, we have

$$\frac{1}{2}U^h(\mathbf{x}) + \int_{\partial D} G_n U^h(\mathbf{x}')\, dl' = -\int_{\partial D} G u_t^p(\mathbf{x}')\, dl' \tag{22}$$

for any $\mathbf{x} \in \partial D$.

Given the computation of ∇u^p according to (14), it is easy to evaluate u_t^p, so that we can regard (22) as a second-kind integral equation for the unknown U^h. We reiterate that it is crucial that this integral equation is of the second kind, in contrast to (10). This integral equation can thus be accurately solved by discretizing each of the integrals on our arc-length grid. It is worth mentioning that the integrand on the left-hand side of (22) is not singular in spite of its appearance. This is because

$$\lim_{\mathbf{x}' \in \partial D \to \mathbf{x}} G_n(\mathbf{x}, \mathbf{x}') = \frac{1}{4\pi}\kappa(\mathbf{x}), \tag{23}$$

where κ is the signed curvature of ∂D. Given a parameterization of the curve $(x(t), y(t))$, it is defined by

$$\kappa := \frac{\ddot{x}\dot{y} - \ddot{y}\dot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}. \tag{24}$$

Thus, G_n restricted to the boundary is a smooth function once we define $G_n(\mathbf{x}, \mathbf{x}) := \kappa(\mathbf{x})/4\pi$. As such, we are free to define

$$\beta_{ij} := \Delta s\, G_n(\mathbf{x}_i, \mathbf{x}_j), \tag{25}$$

where Δs is the arc-length distance between grid points. Thus, the approximation of the integral on the left-hand side of (22) by the trapezoidal rule is written as

$$\int_{\partial D} G_n U^h(\mathbf{x}')\, dl' \approx \sum_j \beta_{ij} U_j^h. \tag{26}$$

The trapezoidal rule suffices since it is well known to converge spectrally for smooth, *periodic* functions [27].

The integrand on the right-hand side of (22), on the other hand, is logarithmically singular. We denote by γ_j the value of the integral at $\mathbf{x} = \mathbf{x}_j$, and again compute it using the standard version of QBX described above.

In the end, the discretized version of (22) is

$$\sum_j \left(\frac{1}{2}\delta_{ij} + \beta_{ij} \right) U_j^h = \gamma_i. \tag{27}$$

This linear system is ill-conditioned because U^h is defined in terms of its derivatives, and thus only well defined up to a constant. We thus expect infinitely many solutions to (27). The problem is solved by imposing the additional constraint that U^h average to zero. That is,

$$\Delta s \sum_j U_j^h = 0. \tag{28}$$

We may impose this constraint without increasing the size of the system by solving

$$\sum_j \left(\frac{1}{2}\delta_{ij} + \beta_{ij} + \Delta s^2 \right) U_j^h = \gamma_i. \tag{29}$$

As shown in [28], this new linear system has a unique solutions which also solves Eq. (22) with probability 1.

The linear system (29) may now be solved using any standard linear algebra package. It bears mentioning that this system is dense, while the stiffness matrix in the FEM formulation is sparse. It may thus seem that solving (29) becomes the rate limiting step in the computation. However, this is not the case. The dimension of the dense system scales with the number of grid points on the boundary, while the stiffness matrix size scales with the number of points in the entire two dimensional domain – roughly the square of the number of boundary points. Thus, the complexity of solving each of the two linear systems is comparable. Moreover, the system (29) need only be solved once, while the finite element system must in general be solved several times since the semi-linear Poisson equation (5) has to be solved iteratively.

4. Implementation details and algorithm summary

As mentioned in Section 2, we choose an isoparametric, bicubic Hermite finite element formulation, which we briefly outline here. The computational domain is represented in (ρ, θ) coordinates, which are defined in terms of the Cartesian coordinates x and y by the following transformation

$$\begin{aligned} x &= x_c + \rho f(\theta)\cos\theta \\ y &= y_c + \rho f(\theta)\sin\theta \end{aligned} \tag{30}$$

for some pre-specified central axis (x_c, y_c). Accordingly, θ is the polar angle, and $f(\theta)$ is represented by a Fourier series, which is computed from given boundary information. When necessary, tangent and normal directions to the boundary may be found by differentiating this Fourier series.

In (ρ, θ) coordinates, the computational domain is $[0,1]\times[0,2\pi)$. This is subdivided into an $N \times N$ grid, to which the FEM is applied using the bicubic Hermite basis in the (ρ, θ) variables. The integrals of the basis functions that must be evaluated are computed using tensor-product Gauss–Legendre quadrature with four nodes in each direction. The solution u of (5) is approximated by iteratively applying this FEM scheme to (7), stopping according to the criterion (8), using infinity norm over the unknowns and $\varepsilon = 5.4 \times 10^{-14}$.

For the purposes of spectral differentiation, it is desirable to have a grid along the boundary that is evenly spaced in arc length. The angles θ_j corresponding to such a grid may be computed using the method described in Appendix A. We construct this grid using $8N$ points, and denote the arc-length grid-points on the boundary by $\mathbf{x}_j$. We proceed through the following steps to find u_x and u_y on the boundary. Throughout, $\mathbf{n}_j = (n_{x,j}, n_{y,j})$ denotes the unit inward normal to the boundary at $\mathbf{x}_j$, and $\mathbf{t}_j$ the unit counterclockwise tangent vector. We present the method in the more complex case in which it is not assumed that $u = 0$ on the boundary, for this general case is required when computing second derivatives.

1. Use FMM accelerated QBX to evaluate
$$\nabla u_j^p = \int_D \nabla G(\mathbf{x}_j, \mathbf{x}') F(\mathbf{x}', u(\mathbf{x}'))\, d\mathbf{x}', \tag{31}$$
as described in Section 3.1.
2. Set $u_{n,j}^p = \mathbf{n}_j \cdot \nabla u^p$ and $u_{t,j}^p = \mathbf{t}_j \cdot \nabla u^p(\mathbf{x}_j)$.
3. Compute
$$\gamma_j = \int_{\partial D} G(\mathbf{x}_j, \mathbf{x}') \left[u_t(\mathbf{x}') - u_t^p(\mathbf{x}')\right] dl' \tag{32}$$
by again using FMM accelerated QBX applied to the computed values of $u_{t,j}^p$. Here, u_t is computed by direct spectral differentiation of the FEM solution.
4. Solve the linear system
$$\sum_j \left(\frac{1}{2}\delta_{ij} + \beta_{ij} + \Delta s^2\right) U_j^h = \gamma_i, \tag{33}$$
where Δs is the spacing of the arc-length grid, as described in Section 3.2. In our code, we simply use MATLAB's "backlash" operator, as we find this is not the rate limiting step in the procedure. More generally, an iterative solver based on the generalized minimal residual method (GMRES) is satisfying since the linear system only needs to be inverted once. The system is well conditioned so will converge quickly, i.e. in $O(1)$ iterations.
5. Compute $U_{t,j}^h$ by spectral differentiation of U_j^h.
6. Set $u_{n,j} = U_{t,j}^h + u_{n,j}^p$, from which it follows that $u_{x,j} = n_{x,j}u_{n,j} + n_{y,j}u_{t,j}$, and $u_{y,j} = n_{y,j}u_{n,j} - n_{x,j}u_{t,j}$.

With this boundary data in hand, we use the same finite element formulation to solve

$$\begin{aligned} \Delta u_x - F_u(\mathbf{x}, u)u_x &= F_x(\mathbf{x}, u) \\ \Delta u_y - F_u(\mathbf{x}, u)u_y &= F_y(\mathbf{x}, u) \end{aligned} \tag{34}$$

for u_x and u_y, respectively.

Computing second derivatives is directly analogous. The procedure above is re-used, but with $u \to u_x$ everywhere and $F(\mathbf{x}, \mathbf{u}) \to F_x(\mathbf{x}, u) + F_u(\mathbf{x}, u)\mathbf{u}_x$ in (31). To be more specific, we evaluate

$$\nabla u_x^p = \int_D \nabla G(\mathbf{x}, \mathbf{x}') \left[F_x(\mathbf{x}', u(\mathbf{x}') + F_u(\mathbf{x}', u(\mathbf{x}'))u_x(\mathbf{x}')\right] d\mathbf{x}' \tag{35}$$

on the arc-length grid, and solve

$$\frac{1}{2}U_x^h(\mathbf{x}) + \int_{\partial D} G_n U_x^h(\mathbf{x}')\, dl' = \int_{\partial D} G(\mathbf{x},\mathbf{x}')\left[u_{xt}(\mathbf{x}') - u_{xt}^p(\mathbf{x}')\right] dl' \tag{36}$$

for U_x^h. Then, $u_{xn} = U_{xt}^h + u_{xn}^p$ and u_{xt} is found by direct spectral differentiation of u_x on the same grid.

Having u_{xn} and u_{xt}, we may find u_{xx} and u_{xy} on the boundary, which are used as boundary data to solve

$$\begin{aligned}\Delta u_{xx} - F_u(\mathbf{x},u)u_{xx} &= F_{xx} + 2F_{xu}u_x + F_{xu}u_x^2 \\ \Delta u_{xy} - F_u(\mathbf{x},u)u_{xy} &= F_{xy} + F_{xu}u_y + F_{yu}u_x + F_{xy}u_x u_y\end{aligned} \tag{37}$$

for u_{xx} and u_{xy} respectively. It is then simple to find u_{yy}, since

$$u_{yy} = F(\mathbf{x},u) - u_{xx} \tag{38}$$

as a consequence of the original PDE (5). Once u and its derivatives are known, it is straightforward to compute ψ and its derivatives to the same accuracy using the formula (4).

5. Numerical results

For numerical tests, we consider two situations in which exact solutions are known. The first situation corresponds to toroidally axisymmetric plasma equilibria with pressure and current profiles chosen so that the right-hand side of the Grad–Shafranov equation does not depend on ψ. Exact solutions are compared with numerical solutions for two magnetic confinement devices, as discussed in detail in section 5.1. In this first situation, both equations (2) and (5) are linear partial differential equations. In order to show that our procedure applies just as well to nonlinear partial differential equations, we consider in section 5.2 an exact solution to the nonlinear Poisson–Boltzmann equation on an ellipse-like domain.

5.1. Grad–Shafranov equation with Solov'ev profiles

In this first example, we solve the Grad–Shafranov equation with a Solov'ev pressure profile [29,30] and no diamagnetic or paramagnetic contribution to the toroidal magnetic field ($dI^2/d\psi = 0$). The simplified Grad–Shafranov equation is given by

$$x\frac{\partial}{\partial x}\left(\frac{1}{x}\frac{\partial \psi}{\partial x}\right) + \frac{\partial^2 \psi}{\partial y^2} = Cx^2. \tag{39}$$

A simple and exact solution to this equation is

$$\psi = \frac{C}{8}x^4 + d_1 + d_2 x^2 + d_3(x^4 - 4x^2y^2) \tag{40}$$

for any constants d_1, d_2, and d_3, where the boundary ∂D corresponding to $\psi = 0$ is simply chosen to be the curve along which the above polynomial vanishes. Of course, not every choice of C, d_1, d_2, and d_3 results in a reasonable plasma boundary. We choose these constants following the approach in [11,30]. Namely, we rewrite them in terms of plasma relevant quantities. These are ϵ, δ, and κ, which are called the inverse aspect ratio, triangulation, and elongation, respectively. By imposing

$$\psi(1+\epsilon, 0) = \psi(1-\epsilon, 0) = \psi(1-\delta\epsilon, \kappa\epsilon) = 0, \tag{41}$$

we may write an invertible linear system that relates $(\epsilon, \delta, \kappa)$ to (d_1, d_2, d_3) [11]. In all our examples, we will use $C = 10$. Its value is not important since it can be scaled out of the problem.

We compute the errors in u and its derivatives using the L^2 norm. That is, the error in any quantity $Q(\mathbf{x})$ is given by

$$\text{Error} = \left(\int_D (Q_{\text{approx}} - Q_{\text{exact}})^2\, d\mathbf{x}\right)^{1/2}. \tag{42}$$

The integral is evaluated using Gauss–Legendre quadrature over the FEM grid in (s,θ) coordinates. In addition, we measure error in the location of the magnetic axis, which is the value of $\mathbf{x}$ at which $\nabla\psi = \mathbf{0}$. This is computed via gradient descent on the $y = 0$ axis, since symmetry tells us the magnetic axis must lie on this line. The numerical result is compared to the exact axis location, given by

$$x^* = 2\sqrt{\frac{-d_2}{C + 8d_3}}. \tag{43}$$

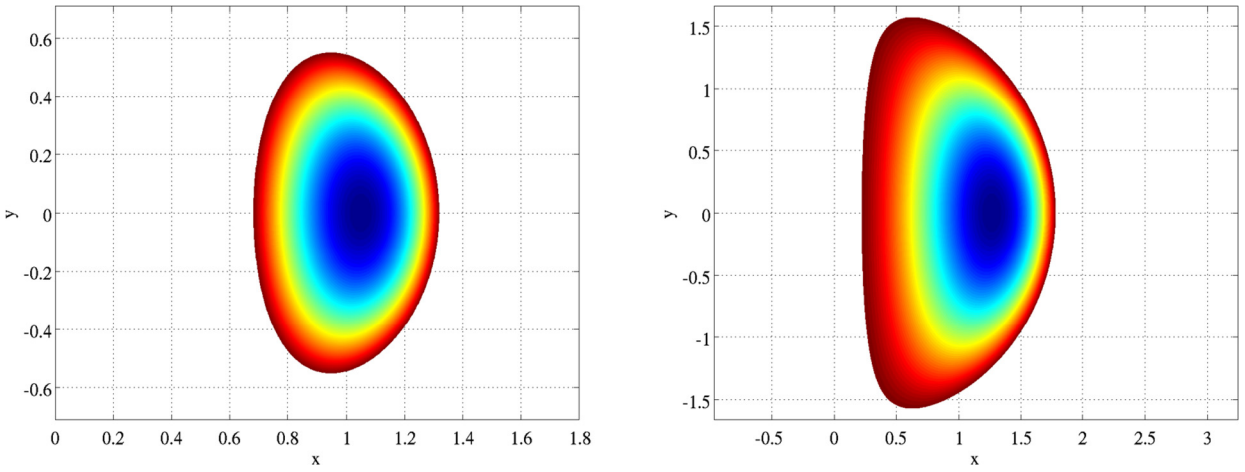

Fig. 2. *Left*: Contours of ψ for the ITER-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.32, 0.33, 1.7)$. *Right*: Contours of ψ for the NSTX-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.78, 0.35, 2)$.

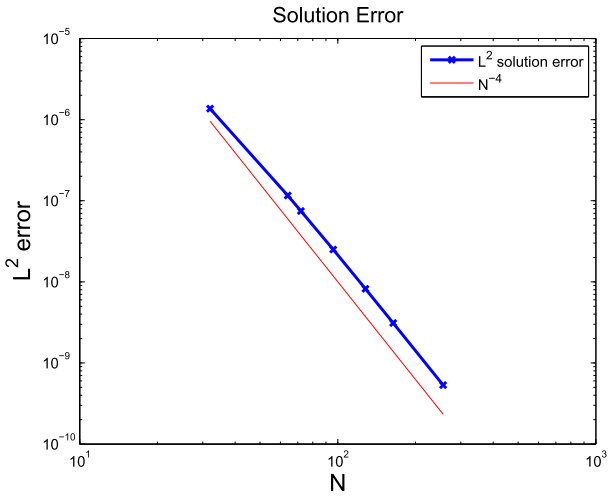

Fig. 3. Error in the solution u for the ITER-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.32, 0.33, 1.7)$, displaying the expected N^{-4} convergence rate.

Note that in order to study convergence in the most general setting and avoid automatic cancellations of numerical errors, we artificially constructed our polar coordinate system so that its center does not coincide with the magnetic axis.

We consider two cases: 1) $(\epsilon, \delta, \kappa) = (0.32, 0.33, 1.7)$, which corresponds to the geometry of the ITER tokamak [31]; 2) $(\epsilon, \delta, \kappa) = (0.78, 0.35, 2)$, which corresponds to the geometry of the spherical tokamak NSTX [32]. For reference, the equilibria are shown in Fig. 2.

5.1.1. ITER example

Let us first focus on the ITER-like equilibrium. In Fig. 3, we plot the error in the solution u. As expected, we see that this error is $O(N^{-4})$ as a result of the FEM scheme used. More interestingly, in Fig. 4 we plot the error in u_x and u_y, as well as the location of the magnetic axis. The dashed lines are computed by reading off u_θ and u_s from the finite element solution, and then converting to rectangular coordinates. As expected, one order of accuracy is lost relative to the solution.

In contrast, the solid lines are the errors using the new method presented in this article. One can observe the improved convergence rate and improved absolute error for any $N > 32$. At $N = 256$, we see an improvement of more than an order of magnitude in each derivative. The error in the magnetic axis is also $O(N^{-4})$. This is to be compared with the results presented in [6], which did not take advantage of our method, and where $O(N^{-3})$ accuracy was observed using the same finite element basis.

Even more significant are the improvements in accuracy of the second derivatives, shown in Fig. 5. For the clarity of the figure, we did not plot the error in u_{yy} obtained with the new method because it is similar to the error in u_{xx} according to equation (38). We did not plot the error in u_{xy} that one obtains with the standard finite element approximation either because it failed to converge – we find an $O(1)$ error independent of N, the reason for which is not known. We note

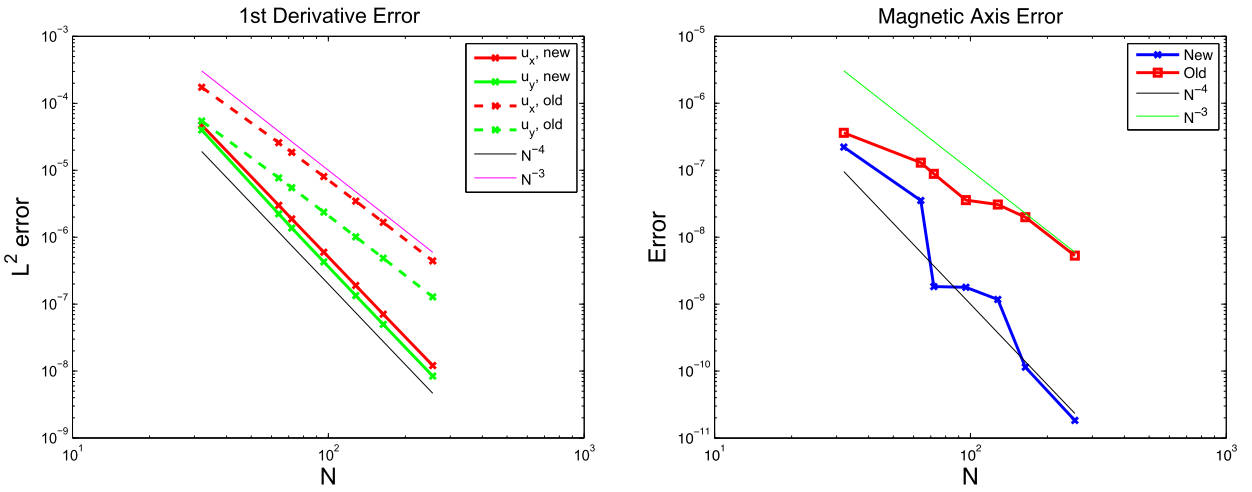

Fig. 4. *Left*: Error in the first partial derivatives for the ITER-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.32, 0.33, 1.7)$. Dashed lines represent the naive computation, while results obtained with the new method presented in this article are shown in solid lines. *Right*: Error in the location of the magnetic axis (in blue), displaying improved convergence relative to results obtained with the direct method (in red) and reported in [6,7]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

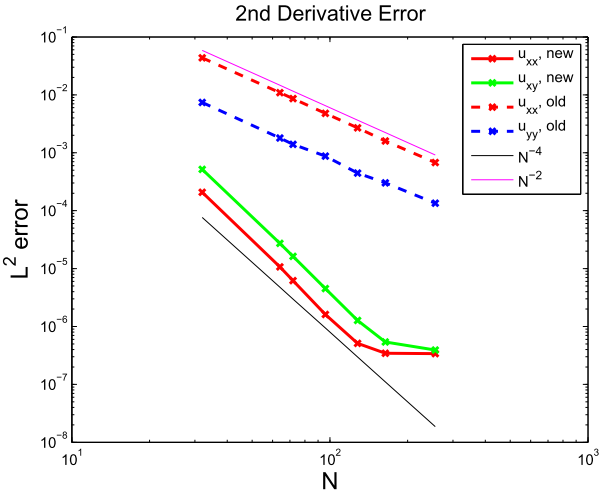

Fig. 5. Error in the second derivatives for the ITER-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.32, 0.33, 1.7)$. Dashed lines represent the naive computation, while results obtained with the new method presented in this article are shown in solid lines.

by looking at Fig. 5 that the improved convergence rate with the new method is observed for moderate N, but the curve flattens as N grows beyond $\approx$128. This is due to round-off error that accumulates from taking numerous spectral derivatives, especially of the Fourier representation of the boundary. This can be easily eliminated when an analytic expression for $f(\theta)$ appearing in (30) is available. However, this is not typically the case. It is possible that a change of variables may be used on a case-by-case basis to minimize the problem, but a more general solution is not readily apparent.

Even with the accumulated round-off error, at the present convergence rate, the standard finite element approximation requires $N \approx 5000$ to reach the accuracy the new method achieves at $N = 256$. That is a reduction in complexity of roughly $(5000/256)^2 \approx 381$, which easily compensates for the additional computation resulting from execution of the methods presented here. An alternate comparison is that for $N = 128$ the new method is more than 1000 times more accurate for u_{xx} than the direct FEM result.

In the event that only 4 digits of accuracy are required in the second derivatives, the new scheme accomplishes this with $N \approx 50$, while the standard method requires $N \approx 315$. The speed improvement in this case is roughly $(315/50)^2 \approx 40$. This again easily outstrips the additional cost devoted to the boundary and the extra finite element solves.

5.1.2. NSTX example

As a second numerical example, we consider the more challenging case of a Solov'ev equilibrium as given by Eq. (39) for the high inverse aspect ratio, highly elongated spherical tokamak NSTX [32], with typical parameters

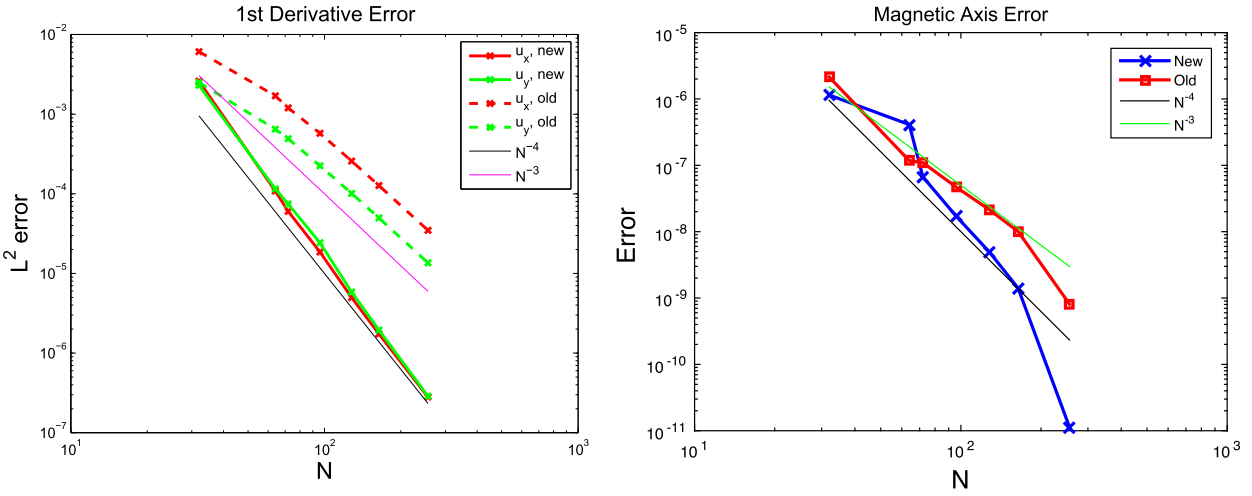

Fig. 6. *Left*: Error in the first partial derivatives for the NSTX-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.78, 0.35, 2)$. Dashed lines represent the naive computation, while results obtained with the new method presented in this article are shown in solid lines. *Right*: Error in the location of the magnetic axis, using the new method presented in this article (blue curve) and the conventional method (red curve). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

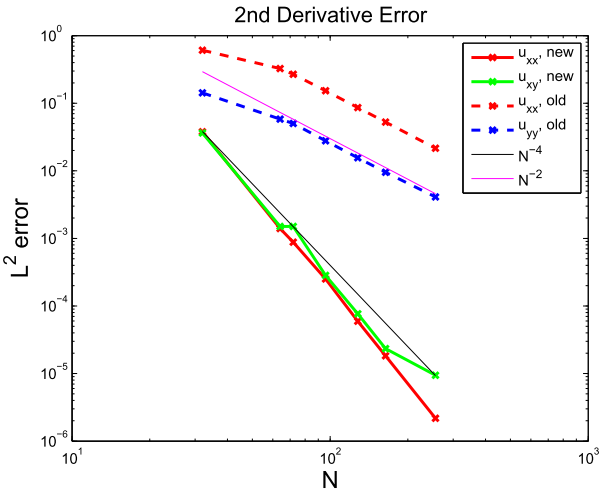

Fig. 7. Error in the second derivatives for the NSTX-like Solov'ev equilibrium given by Eq. (39) with $(\epsilon, \delta, \kappa) = (0.78, 0.35, 2)$. Dashed lines represent the naive computation, while results obtained with the new method presented in this article are shown in solid lines.

$(\epsilon, \delta, \kappa) = (0.78, 0.35, 2)$. The results are shown in Figs. 6 and 7. We did not plot the curves corresponding to u_{yy} with the new method and u_{xy} with the standard method for the same reasons as the ones we gave in section 5.1.1. We observe the same improvement in terms of the order of convergence of the derivatives of the solution to the Grad–Shafranov equation. One does note that for the magnetic axis and second derivatives, the convergence is not as smooth as in the ITER case. We hypothesize that this is due to the fact that the NSTX boundary is more difficult to resolve so that larger N is required to observe the asymptotic behavior.

5.2. Poisson–Boltzmann example

We now verify that the method we describe in this article also performs well for nonlinear cases by considering the nonlinear Poisson problem given by

$$\Delta u = \alpha e^{-u}, \qquad u|_{\partial D} = 0, \tag{44}$$

where ∂D is defined by

$$c_1 \cosh ky - c_2 \cos kx = 1. \tag{45}$$

We pick $\alpha = 2k^2(c_1^2 - c_2^2)$, giving an exact solution,

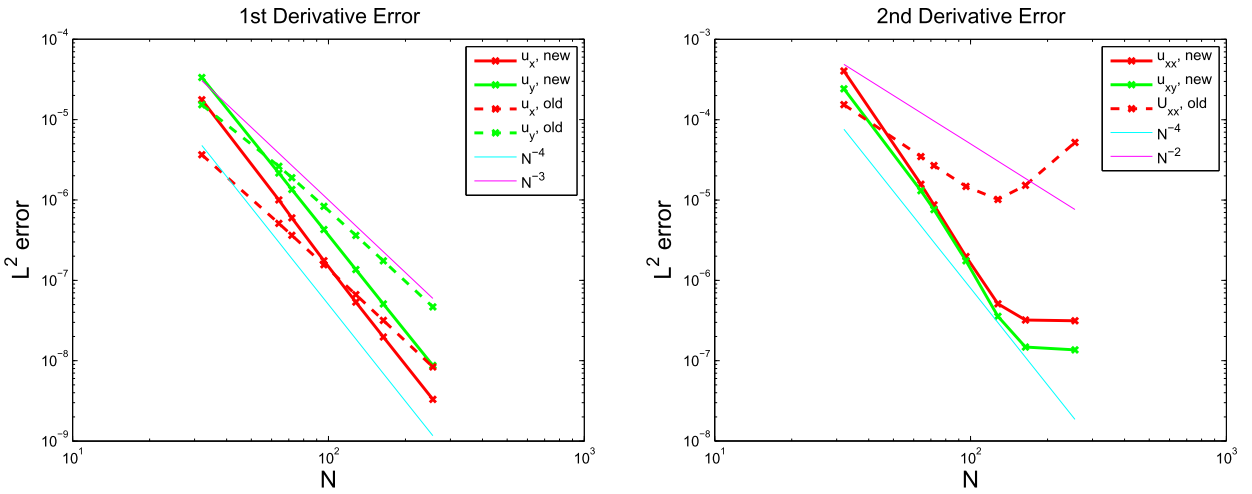

Fig. 8. *Left*: Error in the first partial derivatives for the Poisson–Boltzmann example. Dashed lines represent the naive computation, while results obtained with the new method presented in this article are shown in solid lines. *Right*: Error in the second derivatives, with dashed lines again corresponding to the naive computation and solid lines to the new method.

$$u = 2\log(c_1 \cosh ky - c_2 \cos kx), \tag{46}$$

against which our numerical solutions may be compared. We evaluate the numerical error in the same way as explained in section 5.1. For the free parameters, we choose $k = \pi/5$, $c_1 = 0.0287$ and $c_2 = 0.3301$. This gives an ellipse-like boundary with a 2-to-1 aspect ratio.

Results for the first and second derivatives are plotted in Fig. 8. The method presented here performs as expected. We again observe the flattening of the second derivative curve due to accumulated round-off error from numerous spectral derivatives. Curiously, the finite difference approximation of second derivatives behaves even worse than expected. We've plotted u_{xx}, which initially converges at the expected rate before diverging at large N. However, u_{xy} and u_{yy} fail to converge at all when finite differences are used. This hints that in addition to being more accurate, the new method may prove more robust than the standard approach.

6. Discussion and conclusions

We have presented a method to compute the partial derivatives of the solution u to Poisson's equation or the Grad–Shafranov equation to the same order of accuracy as u itself, while at the same time using the same FEM solver. We do this by differentiating Poisson's equation or the Grad–Shafranov equation analytically, and solving the resulting PDEs for the partial derivatives with this FEM solver. The trace theorem for Sobolev spaces [19] tells us that the error in the solution to these PDEs is bounded from below by the error in the Dirichlet data on the boundary ∂D of the smooth domain $D \in \mathbb{R}^2$. Therefore, the key point for our method to be successful is to calculate this boundary data to the same order of accuracy as u. We do this by relying on an integral equation formulation for the accurate numerical evaluation of the Dirichlet to Neumann map for Poisson's equation. The computational cost of implementing the numerical method we describe in this article is comparable to two Poisson or Grad–Shafranov solves. This additional computational cost is offset by the fact that on small grids, we obtain accuracies for the first and second derivatives that could only be achieved on a much larger grid with the standard method.

It is worth mentioning that while we were testing the isoparametric, bicubic Hermite finite element solver [5–9] we implemented to produce the results we show in this article, we observed unexpected superconvergence for the partial derivatives of u on the boundary ∂D, whose errors decayed as N^{-4} like the error in u. This superconvergence is not observed at points in the interior of D, in agreement with all the results presented in this article. However, this surprising superconvergent behavior – a possible origin of which could be the up-down symmetry of our example problems – suggests that in this special case our strategy could be applied directly without resorting to the calculation of the Dirichlet to Neumann map. While this is true for the cases we considered in this article and the finite element scheme used here, it is unlikely to be true in general, and it is certainly not true for certain classes of finite element schemes, such as the second order scheme described in [3].

In contrast, our numerical method is general and only needs the solution to Poisson's equation or the Grad–Shafranov equation as an input, obtained with any finite element-based solver, which does not need to have superconvergence properties. In the context of plasma physics simulations, a code based on our numerical method can be advantageously combined with existing and commonly used Grad–Shafranov solvers to calculate quantities such as the magnetic field, the parallel current density and the magnetic curvature with much higher accuracy than is possible with the standard methods on the fairly small grids that can usually be afforded. At present, a limitation to our method is that it is only applicable to cases for

which the boundary of the domain is smooth. The quadrature schemes we employ here can handle boundaries with a sharp corner, but the Fourier based method we use to compute spectrally accurate derivatives on the boundary would not yield accurate results for curves that are not smooth. Given the importance of boundaries with corners in magnetic confinement fusion, an improved scheme with this capability, most likely based on local interpolation instead of global interpolation, would be desirable. This is the subject of ongoing work, with progress to be reported at a later date.

Acknowledgements

The authors would like to thank one of the referees for a comment that led us to the discovery of the superconvergence of the FEM solver on the boundary. Thanks are also owed to Prof. Leslie Greengard (NYU CIMS) for many insightful conversations, to Dr. Carlos Borges (NYU CIMS) for his help in implementing the QBX codes in MATLAB, and to Drs. Edmund Highcock (U. Oxford) and Jungpyo Lee (MIT PSFC) for their help in running the solver CHEASE. L.F.R. and A.J.C. were supported by the U.S. Department of Energy, Office of Science, Fusion Energy Sciences under Award Nos. DE-FG02-86ER53223 and DE-SC0012398. M.R. was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics Program under Award DEFGO288ER25053, and by the Office of the Assistant Secretary of Defense for Research and Engineering and AFOSR under NSSEFF Program Award FA9550-10-1-0180.

Appendix A. Arc-length grid computation

We desire a sequence of angles θ_j that correspond to equally spaced points in arc-length, with spacing given by $\Delta s = 8N/L$, where L is the total length of the curve. We first compute L by evaluating

$$L = \int_0^{2\pi} \sqrt{\left(\frac{dx}{d\theta}\right)^2 + \left(\frac{dy}{d\theta}\right)^2}\, d\theta, \tag{A.1}$$

where $dx/d\theta$ and $dy/d\theta$ are computed by differentiating (30), using the Fourier series representation of f to compute its derivative. This integral can be evaluated accurately using any high order quadrature desired. We use Gauss–Legendre with 1000 nodes.

One then sets $\theta_1 = 0$, and solves

$$\frac{8N}{L} = \int_{\theta_{j-1}}^{\theta_j} \sqrt{\left(\frac{dx}{d\theta}\right)^2 + \left(\frac{dy}{d\theta}\right)^2}\, d\theta \tag{A.2}$$

for θ_j given θ_{j-1} using Newton's method. The integral is again evaluated using any desired quadrature. We use Gauss–Legendre with 16 nodes.

References

[1] H. Grad, H. Rubin, Hydromagnetic equilibria and force-free fields, in: Proceedings of the Second United Nations Conference on the Peaceful Uses of Atomic Energy, vol. 31, 1958, p. 190.
[2] V.D. Shafranov, On magnetohydrodynamical equilibrium configurations, Sov. Phys. JETP 6 (1958) 545.
[3] E. Deriaz, B. Després, G. Faccanoni, K. Gostaf, L.-M. Imbert-Gérard, G. Sadaka, R. Sart, Magnetic equations with FreeFem++: the Grad–Shafranov equation & the current hole, in: ESAIM: Proceedings, vol. 32, EDP Sciences, 2011, pp. 76–94.
[4] E.C. Howell, C.R. Sovinec, Solving the Grad–Shafranov equation with spectral elements, Comput. Phys. Commun. 185 (2014) 1415.
[5] G.T.A. Huysmans, J.P. Goedbloed, W. Kerner, Isoparametric bicubic Hermite elements for solution of the Grad–Shafranov equation, Int. J. Mod. Phys. C 2 (1991) 371.
[6] H. Lütjens, A. Bondeson, A. Roy, Axisymmetric mhd equilibrium solver with bicubic Hermite elements, Comput. Phys. Commun. 69 (1992) 287.
[7] H. Lütjens, A. Bondeson, O. Sauter, The CHEASE code for toroidal MHD equilibria, Comput. Phys. Commun. 97 (1996) 219.
[8] J.P. Goedbloed, R. Keppens, S. Poedts, Advanced Magnetohydrodynamics: With Applications to Laboratory and Astrophysical Plasmas, Cambridge University Press, Cambridge, 2010.
[9] A.J.C. Beliën, M.A. Botchev, J.P. Goedbloed, B. van der Holst, R. Keppens, FINESSE: axisymmetric MHD equilibria with flow, J. Comput. Phys. 182 (2002) 91.
[10] K.M. Ling, S.C. Jardin, The Princeton spectral equilibrium code: PSEC, J. Comput. Phys. 58 (1985) 300.
[11] A. Pataki, A.J. Cerfon, J.P. Freidberg, L. Greengard, M. O'Neil, A fast, high-order solver for the Grad–Shafranov equation, J. Comput. Phys. 243 (2013) 28.
[12] J.P. Lee, A.J. Cerfon, ECOM: a fast and accurate solver for toroidal axisymmetric MHD equilibria, Comput. Phys. Commun. 190 (2015) 72.
[13] R. Gruber, F. Troyon, D. Berger, L.C. Bernard, S. Rousset, R. Schreiber, W. Kerner, W. Schneider, K.V. Roberts, ERATO stability code, Comput. Phys. Commun. 21 (1981) 323.
[14] D.H. Liu, A. Bondeson, Improved poloidal convergence of the MARS code for MHD stability analysis, Comput. Phys. Commun. 116 (1999) 55.
[15] S. Jolliet, A. Bottino, P. Angelino, R. Hatzky, T.M. Tran, B.F. Mcmillan, O. Sauter, K. Appert, Y. Idomura, L. Villard, A global collisionless PIC code in magnetic coordinates, Comput. Phys. Commun. 177 (2007) 409.
[16] G. Vlad, S. Briguglio, G. Fogaccia, F. Zonca, Toward a new hybrid MHD gyrokinetic code: progresses and perspectives, in: 11th IAEA Technical Meeting on Energetic Particles in Magnetic Confinement Systems, Kyiv 21–23 Sept. 2009, International Atomic Energy Agency, Vienna, Austria, 2009, P–25.
[17] L.N. Trefethen, D. Bau, Numerical Linear Algebra, SIAM, 1997.
[18] S.C. Jardin, Computational Methods in Plasma Physics, Chapman & Hall/CRC Press, New York, 2010.

[19] L.C. Evans, Partial Differential Equations, American Mathematical Society, 1998.
[20] L.N. Trefethen, Spectral Methods in MATLAB, SIAM, Philadelphia, 2000.
[21] A. Klöckner, A. Barnett, L. Greengard, M. O'Neil, Quadrature by expansion: a new method for the evaluation of layer potentials, J. Comput. Phys. 252 (2013) 332.
[22] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325.
[23] C.L. Epstein, L. Greengard, A. Klöckner, On the convergence of local expansions of layer potentials, SIAM J. Numer. Anal. 51 (2013) 2660.
[24] M. Rachh, A. Klöckner, M. O'Neil, Fast algorithms for 'quadrature by expansion' I: globally valid expansions, in preparation.
[25] Rachh Manas, Integral equation methods for problems in electrostatics, elastostatics and viscous flow, PhD thesis, New York University, 2015.
[26] R.K. Beatson, L. Greengard, A short course on fast multipole methods, in: M. Ainsworth, J. Levesley, W. Light, M. Marletta (Eds.), Wavelets, Multilevel Methods and Elliptic PDEs, Oxford University Press, 1997, pp. 1–37. Freely accessible at http://www.rbfsrus.com/pdfs/beatson+greengard/beatsongreengard.pdf.
[27] L.N. Trefethen, J.A.C. Weideman, The exponentially convergent trapezoidal rule, SIAM Rev. 56 (2014) 385.
[28] J. Sifuentes, Z. Gimbutas, L. Greengard, Randomized methods for rank-deficient linear systems, arXiv:1401.3068, 2014.
[29] L.S. Solov'ev, Sov. Phys. JETP 26 (1968) 400.
[30] A.J. Cerfon, J.P. Freidberg, "One size fits all" analytic solutions to the Grad–Shafranov equation, Phys. Plasmas 17 (2010) 032502.
[31] R. Aymar, P. Barabaschi, Y. Shimomura, The ITER design, Plasma Phys. Control. Fusion 44 (2002) 519.
[32] S.A. Sabbagh, S.M. Kaye, J. Menard, F. Paoletti, M. Bell, R.E. Bell, J.M. Bialek, M. Bitter, E.D. Fredrickson, D.A. Gates, A.H. Glasser, H. Kugel, L.L. Lao, B.P. LeBlanc, R. Maingi, R.J. Maqueda, E. Mazzucato, G.A. Wurden, W. Zhu, NSTX Research Team, Equilibrium properties of spherical torus plasmas in NSTX, Nucl. Fusion 41 (2001) 1601.