Available online at www.sciencedirect.com
SCIENCE DIRECT
Journal of Computational Physics 214 (2006) 657–671
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A finite element Poisson solver for gyrokinetic particle simulations in a global field aligned mesh

Y. Nishimura [a,*], Z. Lin [a], J.L.V. Lewandowski [b], S. Ethier [b]

[a] *Physics and Astronomy, University of California, FRH 4129, Irvine, CA 92697-4575, United States*
[b] *Plasma Physics Laboratory, Princeton University, Princeton, NJ 08543, United States*

Received 12 October 2004; received in revised form 12 May 2005; accepted 10 October 2005
Available online 2 December 2005

Abstract

A new finite element Poisson solver is developed and applied to a global gyrokinetic toroidal code (GTC) which employs the field aligned mesh and thus a logically non-rectangular grid in a general geometry. Employing test cases where the analytical solutions are known, the finite element solver has been verified. The CPU time scaling versus the matrix size employing portable, extensible toolkit for scientific computation (PETSc) to solve the sparse matrix is promising. Taking the ion temperature gradient modes (ITG) as an example, the solution from the new finite element solver has been compared to the solution from the original GTC's iterative solver which is only efficient for adiabatic electrons. Linear and nonlinear simulation results from the two different forms of the gyrokinetic Poisson equation (integral form and the differential form) coincide each other. The new finite element solver enables the implementation of advanced kinetic electron models for global electromagnetic simulations.

PACS: 52.30.Gz; 52.35.Ra; 52.65.Rr

Keywords: Gyrokinetic Poisson equation; Particle simulation; Plasma turbulence; Global field aligned mesh

1. Introduction

In gyrokinetic simulations of turbulence in magnetized plasmas [1], overcoming stringent numerical constraints imposed by the kinetic electrons has been a crucial issue. The split-weight scheme [2] separates the adiabatic and the non-adiabatic electron responses which reduces the statistical noise of the particle simulation. Lin and Chen [3] developed a hybrid electron model which can relax the numerical constraints associated with the fast electron motion in the direction parallel to the magnetic field. Using small parameter expansion based on the square-root of the electron–ion mass ratio, the hybrid model solves the electron adiabatic response in the lowest order and the kinetic response in the higher orders. Both models preserve linear and nonlinear wave–particle interactions. The aim of this paper is to present a newly developed Poisson solver

* Corresponding author. Tel.: +1 949 824 3724; fax: +1 949 824 2174.
E-mail address: nishimuy@uci.edu (Y. Nishimura).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.10.011

which will be the basis for the electromagnetic simulation using both the split-weight scheme [2] and the hybrid model [3].

The global gyrokinetic toroidal code (GTC) [4] currently uses an iterative Poisson solver [5], which is efficient for the adiabatic electron response. The general form of the gyrokinetic Poisson equation is in an integral form [1,5–7]. The iterative solver employs local operations in configuration space to compute the polarization density and automatically takes into account the background profile effects contained in the gyrokinetic Poisson equation [7]. It is useful for global gyrokinetic simulations of toroidal plasmas, where the traditional spectral method is not applicable. However, with the inclusion of the non-adiabatic electron response, using either the split-weight schemes [2,8,9] or the hybrid model [3] for finite-beta plasmas, the resulting gyrokinetic Poisson equation requires a new algorithm. Further, in the electromagnetic split-weight scheme, one needs to solve time derivative of Poisson's equation, as well as Ampere's law for the magnetic perturbation. For all these purposes, a new linear elliptic solver is highly in demand. In its simplest form, the gyrokinetic Poisson equation for electrostatic perturbations is given by

$$\nabla_{\perp}^{2}\Phi = -\sigma, \tag{1}$$

where Φ is the electrostatic potential, σ is the perturbed guiding center charge density averaged over gyromotion, and the subscript $\perp$ denotes the direction perpendicular to the magnetic field.

The global code GTC has a unique grid structure due to the global field aligned mesh employed [10–12]. The field aligned mesh provides the highest computational efficiency without any approximation in geometry to describe the structure of the toroidal drift wave eigenmode. It also keeps the number of particles per cell nearly constant. The computational mesh on a poloidal plane twists along with the sheared magnetic field in the toroidal direction. Consequently, the GTC code employs a logically non-rectangular grid (LNR grid, hereafter) with a number of poloidal grid points which increases radially. This unique mesh structure complicates the Poisson solver. In this work, a finite element method (FEM) [13,14] is introduced for the Poisson solver and successfully implemented into the GTC. The finite element solver developed in this work is also applicable for a general particle in cell code [15].

For the finite element method, one needs to solve a large sparse matrix. To solve the sparse matrix, we employ the state-of-the-art PETSc (portable, extensible toolkit for scientific computation) code [16]. Here, PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the message passing interface (MPI) standard for message-passing communications [16].

The elliptic gyrokinetic Poisson solver in GTC is inherently two-dimensional as a result of perpendicular polarization drift [1]. Taking an international thermonuclear experimental reactor (ITER) size plasma as an example, where the minor radius is on the order of one thousand ion Larmor radii, several million grid points per poloidal plane are used (for a total of typically 32–64 poloidal planes). This requires a fast and efficient way of solving the resulting elliptic equations, the multigrid method, for example. To speed up the matrix solver, the PETSc code is interfaced with *hypre* (high performance preconditioners) [17] which employs algebraic multigrid (AMG) method [18] as a preconditioner. Here, preconditioning means that we change variables to obtain a new equation whose coefficient matrix has a better eigenvalue distribution than that of the original coefficient matrix. In this paper, as a first step, we focus on the finite element method and application of the PETSc code to the gyrokinetic particle simulation.

This paper is organized as follows. In Section 2, the basic model for the gyrokinetic Poisson equation is reviewed. We then discuss the finite element method applied to the GTC grid geometry. Section 3 presents the verification of the finite element solver by employing test problems whose analytical solutions are known. In Section 4 taking the ion temperature gradient (ITG) mode in toroidal geometry as an example, we focus on the benchmark of two solutions from the new FEM solver and the original iterative field solver [5]. We summarize this work in Section 5.

2. Gyrokinetic Poisson equation

In this section, we review the gyrokinetic Poisson equation. We also review the iterative method employed in the original GTC [5] to illustrate the difference between the new finite element solvers and the iterative solver. Following Lee [7], the gyrokinetic Poisson equation is given by

$$\nabla^2\Phi - \frac{\tau}{\lambda_d^2}(\Phi - \tilde{\Phi}) = -4\pi e(\delta\bar{n}_i - \delta n_e), \tag{2}$$

where e is the unit charge, λ_d is the electron Debye length, $\delta\bar{n}_i$, δn_e are the ion and the electron guiding center charge density, and $\tau = T_e/T_i$ is the ratio between the equilibrium electron temperature T_e and the equilibrium ion temperature T_i. Here, $4\pi e(\delta\bar{n}_i - \delta n_e)$ corresponds to σ of Eq. (1). In Eq. (2), the gyro-phase averaged potential $\bar{\Phi}$ and the *second* gyro-phase averaged potential $\tilde{\Phi}$ are given by [5]

$$\bar{\Phi}(\mathbf{R}) = \frac{1}{2\pi}\int \Phi(\mathbf{x})\delta(\mathbf{x} - \mathbf{R} - \boldsymbol{\rho})\mathbf{dx}\, d\phi$$

and

$$\tilde{\Phi}(\mathbf{x}) = \frac{1}{2\pi}\int \bar{\Phi}(\mathbf{R})F_M(\mathbf{R}, \mu, v_\parallel)\delta(\mathbf{R} - \mathbf{x} + \rho)\mathbf{dR}\, d\mu\, dv_\parallel\, d\phi,$$

where $\mathbf{R}$ is the gyro-center position, $\mathbf{x}$ is the position of the particles, $\rho = \mathbf{x} - \mathbf{R}$ corresponds to the Larmor radius, and F_M stands for the Maxwellian distribution. Here, ϕ is the gyro-phase, μ is the magnetic moment, $v_\parallel$ is the parallel velocity. The Debye length is given by $\lambda_d = (T_e/4\pi n_0 e^2)^{1/2}$, where n_0 is the background equilibrium ion and electron densities. In Eq. (2), the first term is the Debye shielding term [7]. The second term is derived by Lee [1] and later by Dubin et al. [6]. Physically, the second term comes from the ion polarization (or the inertia) effects. From the ordering $(\rho_s/\lambda_d)^2 \gg 1$ [6,7], only the second term is retained [5], giving Eq. (2) a form of an integral equation. Here, ρ_s is the ion gyro-radius at the electron temperature. To solve Eq. (2) numerically, an iterative double-gyro-averaging scheme [5] is employed. This iterative method has been one of the key components of the GTC [4].

Unfortunately, the iterative method (under the integral form) cannot be applied for the non-adiabatic kinetic electrons, when they are used in the split-weight schemes [2] and the electromagnetic hybrid model [3]. In the presence of non-adiabatic electrons, the inversion matrix of the iterative method cannot be diagonally dominant. Note this is an inherent nature of the gyrokinetic Poisson equation.

By expansion in the long wavelength limit, the second term of Eq. (2) becomes [1]

$$-\tau\lambda_d^{-2}(\Phi - \tilde{\Phi}) \sim \tau(\omega_{pi}/\Omega_i)^2\nabla_\perp^2\Phi, \tag{3}$$

where ω_{pi} is the ion plasma frequency and Ω_i is the ion cyclotron frequency [note the Debye shielding term in Eq. (2) and the $(\omega_{pi}/\Omega_i)^2\nabla_\perp^2\Phi$ term in Eq. (3) are different]. As noted above, as a result of perpendicular polarization effect [1], the Poisson equation is essentially two-dimensional when the Debye shielding term is neglected.

To calculate the response of the short wavelength mode correctly, Padé approximation [19,20] should be introduced on the right-hand side of the gyrokinetic Poisson equation. Using Eq. (2), together with the approximation of Eq. (3), we obtain

$$\tau(\omega_{pi}/\Omega_i)^2\nabla_\perp^2\Phi = -4\pi e\left(1 - \rho_i^2\nabla_\perp^2\right)(\delta\bar{n}_i - \delta n_e), \tag{4}$$

where ρ_i is the ion Larmor radius. Normalizing Eq. (4) with ρ_s for the length, the background density n_0, and T_e/e for the potential, we obtain (note $\rho_s^2 = \tau\rho_i^2$)

$$\nabla_\perp^2\Phi = -\left(1 - \frac{1}{\tau}\nabla_\perp^2\right)(\delta\bar{n}_i - \delta n_e). \tag{5}$$

Hereafter Φ, $\delta\bar{n}_i$, δn_e, and the $\nabla_\perp$ operator stand for normalized values. Eq. (5) is employed for the simulations in the presence of non-adiabatic kinetic electrons.

With the adiabatic condition $\delta n_e = \Phi$, Eq. (5) reduces to

$$\left(1 - \frac{1+\tau}{\tau}\nabla_\perp^2\right)\Phi = \left(1 - \frac{1}{\tau}\nabla_\perp^2\right)\delta\bar{n}_i \tag{6}$$

with a Helmholtz type operator on the left-hand side. Eq. (6) is employed for the simulations of ITG turbulence where electrons are adiabatic. More complicated form is needed, for example, to simulate low aspect ratio tokamaks.

As we discussed above, the global field aligned coordinate [10–12] gives rise to unique computational grid structures. Fig. 1(a) is the topology of the conventional GTC grid at a particular toroidal angle. The structures of the grid are different at other toroidal angles. Compared to the nominal polar coordinate, the grid points of the azimuthal coordinate increase with respect to the radial coordinate. For example, we have four grid points on the first flux surface ($i_r = 1$) and eight grid points on the second flux surface ($i_r = 2$). Here, the term flux surface (a plasma physics terminology) implies a plane with a constant (r in this simple case with a circular cross-section, the GTC applies to arbitrary cross-sections). The arc length of the segments is kept nearly constant for uniform resolution, $\Delta r = r\Delta\theta$, where Δr and $\Delta\theta$ are the mesh size in the radial (r) and the azimuthal (θ) directions.

One way to solve Eqs. (5) and (6) is to employ the finite difference scheme [22]. When a finite difference scheme is used on our LNR grid, we encounter difficulties. First, one needs to find corresponding "ghost points" (in the radial direction). Second, problems arise near the magnetic axis due to limited number of grid points (see Appendix for ghost points interpolation. Conventionally, GTC simulations [21] exclude center regions since there are not significant turbulence activities near the axis). Third, the application of the geometric multigrid methods becomes extremely difficult.

Instead, we employ the finite element method (FEM) in this work [13]. With the finite element method, the Poisson equation can be solved in the logically non-rectangular grid. Fig. 1(b) demonstrates application of the new FEM grid generator. In general, the FEM is suitable for dealing with complicated geometries, where unstructured meshes are employed. In Fig. 1(b), the poloidal plane is divided into quadrants (this is for the purpose of second domain decomposition in the future) [16,23]. In Fig. 1(b), the boundaries are connected by linear elements but can be smooth enough when sufficient grid points are taken.

2.1. Introducing a finite element method

The application of the finite element method for Eqs. (5) and (6) are described below. Denoting $(x_{i,j,k}, y_{i,j,k})$ as the Cartesian coordinate variables for the vertices ($1 \leqslant i,j,k \leqslant N$), a linear shape function $S^{(e)}$ for Φ is given by

$$S^{(e)} = \frac{1}{4\Delta^{(e)}}[(x_i y_k - x_k y_j) + (y_j - y_k)x + (x_k - x_j)y] = \begin{cases} 1 & \text{on the } i\text{th vertex}, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

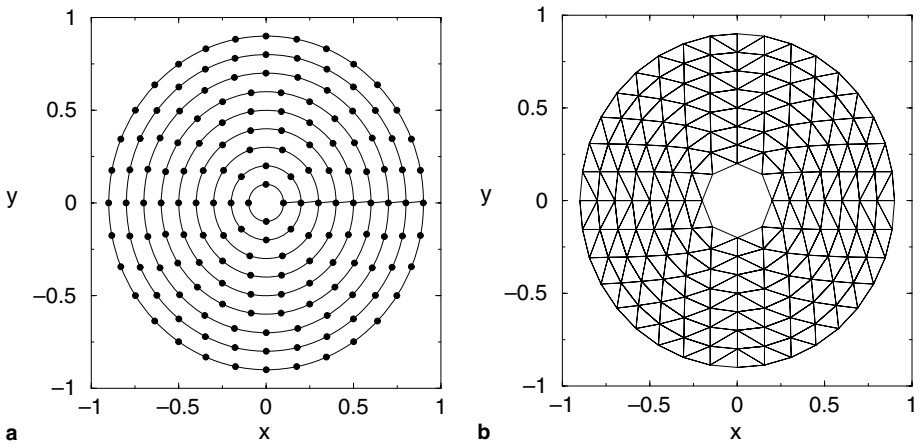


Fig. 1. Adaptation of the FEM generator to the GTC grid. Here, very small numbers of grid points are taken to emphasize the GTC grid topology. (a) A conceptual plot of the logically non-rectangular GTC grid. (b) The GTC grid with an adaptation of triangular finite elements. Note in (b), the poloidal plane is divided into quadrants (this is for the purpose of second domain decomposition). In (b), grid point locations are adjusted; the number of the grid points increases by a constant factor of four in the radial direction.

where

$$\Delta^{(e)} = \tfrac{1}{2}[(x_i - x_k)(y_j - y_k) - (x_k - x_j)(y_k - y_i)] \tag{8}$$

is the area of the triangle elements. Indices i,j,k are labeled counter clock-wise (see Fig. 2(a)). Assembling local 3×3 matrices, Eq. (6) can be written in the form [13,14]

$$\sum_{j=1}^{N} (m_{ij}\Phi_j + k_{ij}\Phi_j) = d_i. \tag{9}$$

In Eq. (9), N is the number of vertices,

$$m_{ij} = \frac{\Delta^{(e)}}{12}(1 + \delta_{ij}) = \begin{cases} \Delta^{(e)}/6, & i = j, \\ \Delta^{(e)}/12, & i \neq j, \end{cases} \tag{10}$$

$$k_{ij} = \frac{1+\tau}{\tau}\frac{1}{4\Delta^{(e)}}[(y_j - y_k)(y_k - y_i) + (x_k - x_j)(x_i - x_k)] \tag{11}$$

and

$$d_i^{(e)} = \frac{\Delta^{(e)}}{12}[2\sigma_i(t) + \sigma_j(t) + \sigma_k(t)], \tag{12}$$

where δ_{ij} is the Kronecker's delta. Here, $\sigma_{i,j,k}$ stand for the particles charges gathered on the vertices i, j and k. The charge gathering scheme stays the same as in the original Poisson solver [5]. In the particle simulations, the charge densities $\sigma_{i,j,k}$ change with time and thus the right-hand side of Eq. (12) needed to be calculated inside the particle code at each time step, while the geometrical information contained in the matrix m_{ij} and k_{ij} stay the same throughout the simulation.

2.2. The matrix equation

To include the boundary condition, let us rewrite Eq. (9) explicitly in a matrix form. In addition, a transformation from the local to the global matrix is demonstrated with an example of three triangle elements with four vertices (see Fig. 2). For the triangle (1) of Fig. 2(b), we have [13]

$$\begin{pmatrix} \Delta^{(1)}/6 + k_{11} & \Delta^{(1)}/12 + k_{12} & \Delta^{(1)}/12 + k_{14} \\ \Delta^{(1)}/12 + k_{21} & \Delta^{(1)}/6 + k_{22} & \Delta^{(1)}/12 + k_{24} \\ \Delta^{(1)}/12 + k_{41} & \Delta^{(1)}/12 + k_{42} & \Delta^{(1)}/6 + k_{44} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} d_1^{(1)} \\ d_2^{(1)} \\ d_4^{(1)} \end{pmatrix}, \tag{13}$$

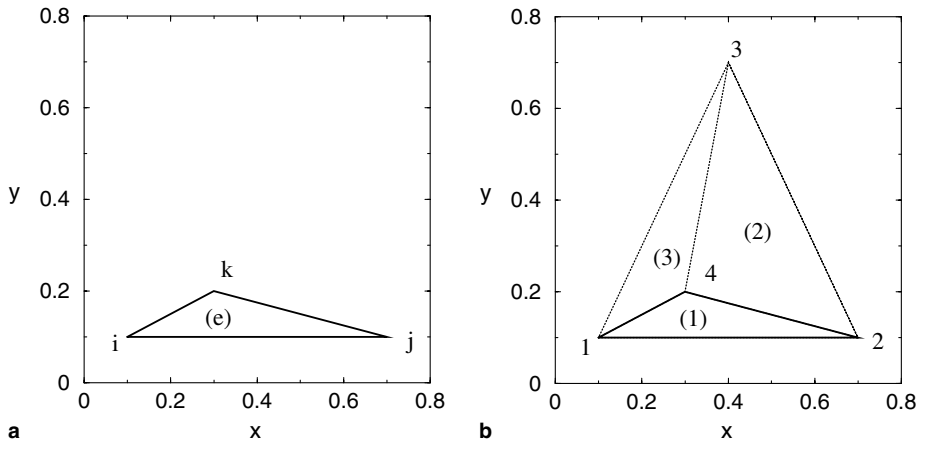


Fig. 2. An example of the finite element labeling scheme with a three-node triangular element. (a) Labeling i, j, k for one element. The labeling is counter-clock wise. (b) Discretization of the solution domain and the system numbering scheme. Cited from Huebner et al. [13].

In the same manner, for the triangle (2)

$$\begin{pmatrix} \Delta^{(2)}/6+k_{22} & \Delta^{(2)}/12+k_{23} & \Delta^{(2)}/12+k_{24} \\ \Delta^{(2)}/12+k_{32} & \Delta^{(2)}/6+k_{33} & \Delta^{(2)}/12+k_{34} \\ \Delta^{(2)}/12+k_{42} & \Delta^{(2)}/12+k_{43} & \Delta^{(2)}/6+k_{44} \end{pmatrix} \begin{pmatrix} \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} d_2^{(2)} \\ d_3^{(2)} \\ d_4^{(2)} \end{pmatrix} \tag{14}$$

and for the triangle (3)

$$\begin{pmatrix} \Delta^{(3)}/6+k_{33} & \Delta^{(3)}/12+k_{31} & \Delta^{(3)}/12+k_{34} \\ \Delta^{(3)}/12+k_{13} & \Delta^{(3)}/6+k_{11} & \Delta^{(3)}/12+k_{14} \\ \Delta^{(3)}/12+k_{43} & \Delta^{(3)}/12+k_{41} & \Delta^{(3)}/6+k_{44} \end{pmatrix} \begin{pmatrix} \Phi_3 \\ \Phi_1 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} d_3^{(3)} \\ d_1^{(3)} \\ d_4^{(3)} \end{pmatrix}. \tag{15}$$

These are called the local matrices [13]. Note that the superscript indices with the brackets (e) are for the elements and the subscripts i, j, k without brackets are for the vertices. Eqs. (13)–(15) are equivalent to writing three 4×4 matrices [13],

$$\begin{pmatrix} \Delta^{(1)}/6+k_{11} & \Delta^{(1)}/12+k_{12} & 0 & \Delta^{(1)}/12+k_{14} \\ \Delta^{(1)}/12+k_{21} & \Delta^{(1)}/6+k_{22} & 0 & \Delta^{(1)}/12+k_{24} \\ 0 & 0 & 0 & 0 \\ \Delta^{(1)}/12+k_{41} & \Delta^{(1)}/12+k_{42} & 0 & \Delta^{(1)}/6+k_{44} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} d_1^{(1)} \\ d_2^{(1)} \\ 0 \\ d_4^{(1)} \end{pmatrix}, \tag{16}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \Delta^{(2)}/6+k_{22} & \Delta^{(2)}/12+k_{23} & \Delta^{(2)}/12+k_{24} \\ 0 & \Delta^{(2)}/12+k_{32} & \Delta^{(2)}/6+k_{33} & \Delta^{(2)}/12+k_{34} \\ 0 & \Delta^{(2)}/12+k_{42} & \Delta^{(2)}/12+k_{43} & \Delta^{(2)}/6+k_{44} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ d_2^{(2)} \\ d_3^{(2)} \\ d_4^{(2)} \end{pmatrix}, \tag{17}$$

$$\begin{pmatrix} \Delta^{(3)}/6+k_{11} & 0 & \Delta^{(3)}/12+k_{13} & \Delta^{(3)}/12+k_{14} \\ 0 & 0 & 0 & 0 \\ \Delta^{(3)}/12+k_{31} & 0 & \Delta^{(3)}/6+k_{33} & \Delta^{(3)}/12+k_{34} \\ \Delta^{(3)}/12+k_{41} & 0 & \Delta^{(3)}/12+k_{43} & \Delta^{(3)}/6+k_{44} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} d_1^{(3)} \\ 0 \\ d_3^{(3)} \\ d_4^{(3)} \end{pmatrix}. \tag{18}$$

Note that the order has been rearranged in the last equation [13]. Adding all together, and denoting $a_{ij} = m_{ij} + k_{ij} = \Delta^{(e)}(1+\delta_{ij})/12 + k_{ij}$, we obtain

$$\begin{pmatrix} a_{11}^{(1)}+a_{11}^{(3)} & a_{12}^{(1)} & a_{13}^{(3)} & a_{14}^{(1)}+a_{14}^{(3)} \\ a_{21}^{(1)} & a_{22}^{(1)}+a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(1)}+a_{24}^{(2)} \\ a_{31}^{(3)} & a_{32}^{(2)} & a_{33}^{(2)}+a_{33}^{(3)} & a_{34}^{(2)}+a_{34}^{(3)} \\ a_{41}^{(1)}+a_{41}^{(3)} & a_{42}^{(2)}+a_{42}^{(2)} & a_{43}^{(2)}+a_{43}^{(3)} & a_{44}^{(1)}+a_{44}^{(2)}+a_{44}^{(3)} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{pmatrix} = \begin{pmatrix} d_1^{(1)}+d_1^{(3)} \\ d_2^{(1)}+d_2^{(2)} \\ d_3^{(2)}+d_3^{(3)} \\ d_4^{(1)}+d_4^{(2)}+d_4^{(3)} \end{pmatrix}. \tag{19}$$

Eq. (19) is called the global matrix [13]. For N vertices in general, Eq. (19) is given by an $N\times N$ matrix

$$\sum^{(e)} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1i} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2i} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{iN} & \cdots & a_{iN} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{Ni} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_i \\ \vdots \\ \Phi_N \end{pmatrix} = \sum^{(e)} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_i \\ \vdots \\ d_N \end{pmatrix}, \tag{20}$$

where $\sum^{(e)}$ denotes contributions from all the triangle elements. In this paper, we employ a homogeneous Dirichlet boundary condition ($\Phi_i = C = 0$) on the inner-most flux surface and the outer-most flux surface. To include the Dirichlet boundary condition, we modify Eq. (20) to

$$\sum^{(e)} \begin{pmatrix} a_{11} & a_{12} & \cdots & 0 & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & 0 & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{N1} & a_{N2} & \cdots & 0 & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_i \\ \vdots \\ \Phi_N \end{pmatrix} = \sum^{(e)} \begin{pmatrix} d_1 - a_{1i}C \\ d_2 - a_{2i}C \\ \vdots \\ C \\ \vdots \\ d_N - a_{Ni}C \end{pmatrix} = \sum^{(e)} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ 0 \\ \vdots \\ d_N \end{pmatrix}. \tag{21}$$

Hereafter, we denote Eq. (21) as $A \cdot \Phi = d$. Note, in Eq. (21), by setting $m_{ij} = 0$, we recover nominal Poisson's equation, Eq. (5), instead of the Helmholtz equation. An example of the *shape* of the matrix [the location of the nonzero components of the matrix in Eq. (21)] is shown in Fig. 3. The maximum number of nonzero components is seven for each row (or column), and we have a sparse positive semi-definite symmetric matrix. The four period we see in Fig. 3 is due to the labeling schemes we took for the vertices (the quadrant by quadrant ordering). Once the matrix is generated, our task is reduced to a routine work of solving $A \cdot \Phi = d$. Rather than developing a matrix solver by our own, we employ the state-of-the-art PETSc code from Angonne National Laboratory [16]. The main task in FEM is the *book-keeping* in relating the labels of the vertices and the labels of the triangle elements. To save the computational memory, in Eq. (21), only (1) the nonzero column index for each rows and (2) the nonzero values of a_{ij} for each rows are passed to the matrix solver.

The generalization of the grid generation to a shaped plasma is straightforward. Fig. 4 shows the application of the grid generator to the geometry of the National Spherical Tokamak Experiment (NSTX) equilibrium profile [24], an ultra-low aspect ratio tokamak. By knowing the number of the grid points on each flux surfaces beforehand (the number of the grid points can be arbitrary), the new grid generator automatically relates the vertex labels and the element labels.

One approach in speeding up the elliptic type solver is to employ the geometric multigrid (GMG) method. The geometric multigrid method is very useful in simple, logically rectangular geometries, such as Cartesian or cylindrical ones where the finite difference method applies. Another approach in speeding up the elliptic solver is to employ the algebraic multigrid (AMG) method, which is a multilevel method where geometry information is not required (as compared to GMG). The algebraic multigrid method is useful for sparse matrices obtained from FEM. We plan to employ AMG [17,18,26] to further accelerate the elliptic solve.

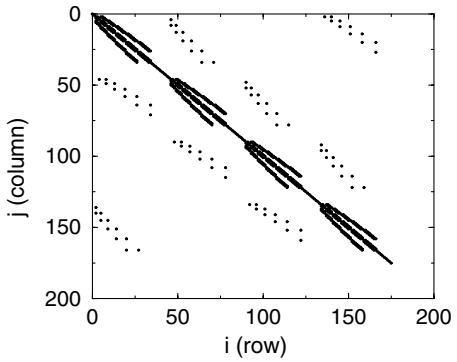


Fig. 3. The shape of a banded sparse matrix for the grid of Fig. 1(b). The boundary condition of Eq. (21) is included. In general, we obtain symmetric semi-positive definite matrix. The number of maximum nonzero components for each rows (columns) is seven. The matrix shape stays the same at larger N.

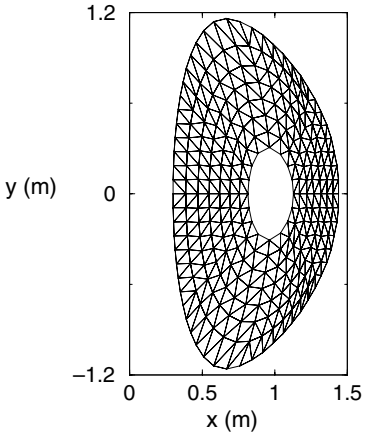


Fig. 4. The application of the grid generator to the shaped plasmas using the National Spherical Tokamak Experiment (NSTX) equilibrium. The grid points are obtained from a Grad–Shafranov equilibrium. By the courtesy of J. Manickam, Princeton Plasma Physics Laboratory.

2.3. Toroidal dependence of elements

Here, we discuss how we include the effects of helically twisted magnetic field lines (q-profiles or the safety factor) into the finite element grid generator. In the GTC, the field aligned coordinate [10–12] is used where the magnetic field is given by

$$\mathbf{B} = \nabla\psi \times \nabla\alpha, \tag{22}$$

where $\alpha = \theta - \zeta/q$ [25]. Here, α is the helical angle, θ is the poloidal angle and ζ is the toroidal angle. Due to the finite value of the safety factor $q(r)$, the grid points on each field lines rotate and move to different locations depending on the toroidal angle. Further, since $q(r)$ in general is not a constant, the flux surfaces do not rotate rigidly. This feature is illustrated in Fig. 5 by *fans*. Each fan represents one-fourth of the poloidal plane. Fig. 5(a) shows the fans at $\zeta = 0$ and $\zeta = \pi$. As a consequence of non-constant $q(r)$ (finite magnetic shear), the triangles are highly distorted at $\zeta = \pi$. For the accuracy of the finite element method, we need to retain each angle of the triangle elements to be larger than certain values [14]. Fig. 5(b) shows how we remedied this

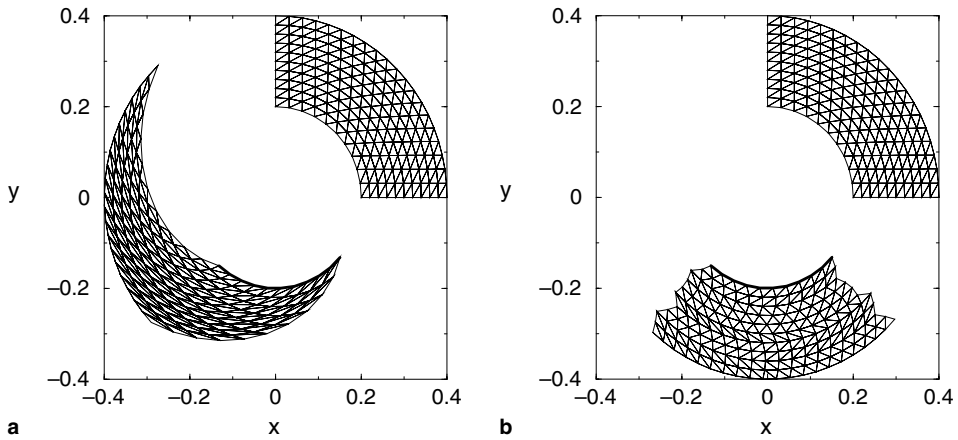


Fig. 5. The 2D FEM Poisson solve is applied to multiple poloidal planes. Each fan represents one-fourth of the poloidal plane. (a) Fans at $\zeta = 0$ and $\zeta = \pi$ (ζ is the toroidal angle). Observe the distortion of the triangle elements. (b) Fans at $\zeta = 0$ and $\zeta = \pi$ with modified triangle elements. By knowing the values of $q(r)$ on each flux surfaces, the elements are forwarded (counter-clock wise) to align closely with that of the inner most flux surface (the solid lines). The vertices still stay on the original GTC grid points. Here, the q-profile is taken as $q(r) = 0.6 + 0.6/(0.358)^2 r^2$ to illustrate the distortion.

problem. Fig. 5(b) shows the fans at $\zeta = 0$ and $\zeta = \pi$, but this time with modified elements. Knowing the value of $q(r)$ on each flux surfaces, the elements are forwarded (counter-clock wise) to align closely with that of the inner-most flux surface suggested by the thick lines. Note the vertices still stay on the original GTC grid points. It is only the labeling of the vertices that has been changed. Consequently, the labeling of the vertices differs depending on poloidal planes and we generate multiple matrices A of Eq. (21). The geometrical information for multiple 32 (or 64) [21] poloidal planes is stored in 32 (or 64) A matrices.

3. Verification and CPU-timing of the new finite element solver

In this section, we generate turbulence like electrostatic potential profile, Φ, whose analytical solution is known. We check the validity of the new FEM solver. The term FEM solver in this paper describes the combination of the grid generator, the matrix generator, and the PETSc code as the matrix solver.

Denoting r as the radial and θ as the azimuthal variable in the polar coordinate, the test analytical profile Φ_{math} is taken as

$$\Phi_{\text{math}}(r,\theta) = \left[\sin(2\pi\xi) + \epsilon \sum_M B_M \sin(2\pi M\xi)\right] \sum_l A_l \cos(l\theta + \Theta_l), \tag{23}$$

where $0 \leqslant \epsilon \leqslant 1$ is a small parameter and $\xi = (r - r_{\min})/(r_{\max} - r_{\min})$ ($r_{\min}$ and $r_{\max}$ are the minimum and the maximum range of r). The Fourier coefficients A_l and B_M are generated by a random number generator and l is taken in the range of $-l_{\max} \leqslant l \leqslant l_{\max}$ (similarly $-M_{\max} \leqslant M \leqslant M_{\max}$). Here, $l_{\max}$ ($M_{\max}$) is the "wave mode number" at the finest resolution that can exist in the azimuthal (radial) direction of the simulating domain. Since we restrict the GTC simulations to circular cross-sections in this work, $x = r\cos\theta$ and $y = r\sin\theta$ apply throughout this paper. The first term in Eq. (23) is an analogy of the zonal flows [4,27] (counter streaming within the range of $r_{\min} \leqslant r \leqslant r_{\max}$) and the second term is an analogy of the drift wave turbulence fluctuations. The phase Θ_l is also given by the random number generator. With Eq. (23), the right-hand side of Eq. (1) is given by

$$\begin{aligned}\sigma(r,\theta) &= -\nabla^2\Phi_{\text{math}}(r,\theta) = -\frac{1}{r}\left(\frac{\partial}{\partial r} r \frac{\partial \Phi_{\text{math}}}{\partial r}\right) - \frac{1}{r^2}\frac{\partial^2 \Phi_{\text{math}}}{\partial \theta^2} \\ &= -\frac{1}{r}\sigma_r(r,\theta)\sum_l A_l \cos(l\theta + \Theta_l) + \left[\sin(2\pi\xi) + \epsilon\sum_M B_M \sin(2\pi M\xi)\right]\sum_l \frac{l^2}{r^2} A_l \cos(l\theta + \Theta_l),\end{aligned}$$

where

$$\sigma_r(r,\theta) = \frac{2\pi}{\Delta r}\cos(2\pi\xi) - \frac{4\pi^2 r}{\Delta r^2}\sin(2\pi\xi) + \epsilon\sum_M B_M\left[\frac{2\pi M}{\Delta r}\cos(2\pi M\xi) - \frac{4\pi^2 M^2 r}{\Delta r^2}\sin(2\pi M\xi)\right].$$

Note Eq. (23) satisfies the homogeneous Dirichlet boundary condition $\Phi = 0$ at $r = r_{\min}$ and $r = r_{\max}$. Fig. 6 compares the analytical profile of Eq. (23) and numerical solutions obtained from the FEM solver (the profiles are extracted at $\theta = 0$). The parameters taken are $l_{\max} = 20$, $M_{\max} = 20$, $\epsilon = 0.4$, $r_{\min} = 0.2$, and $r_{\max} = 0.4$ (the annular simulating domain of $0.2 \leqslant r \leqslant 0.4$). The dots represent the numerical solution while the solid lines represent the mathematical solution. As shown in Fig. 6, the two solutions match. A total of $N = 154{,}560$ grid points are taken in the example of Fig. 6.

Further, we have calculated the L-2 norms of the difference between the analytical and the numerical solutions shown in Fig. 6. The grid size N is varied. Letting $L_2 \equiv (1/N)\sum_N (\Phi - \Phi_{\text{math}})^2$, we obtain $L_2 = 7.705 \times 10^{-4}$ (for $N = 9840$), $L_2 = 5.890 \times 10^{-5}$ (for $N = 38{,}880$), and $L_2 = 8.847 \times 10^{-6}$ (for $N = 154{,}560$). The L_2 value decreases as the number of the grid size increases. This tendency supports the linear shape function Eq. (7) (instead of employing higher order shape functions). In solving our matrix equation (21), PETSc's main solve employs the generalized minimal residual (GMRES) method and the incomplete LU decomposition for the preconditioner [22].

The solid line in Fig. 7(a) is the CPU time (in seconds) versus the number of the grid points "N" (as a reminder the matrix size is $N \times N$). The scaling is obtained using a single processor on IBM SP-3 at National

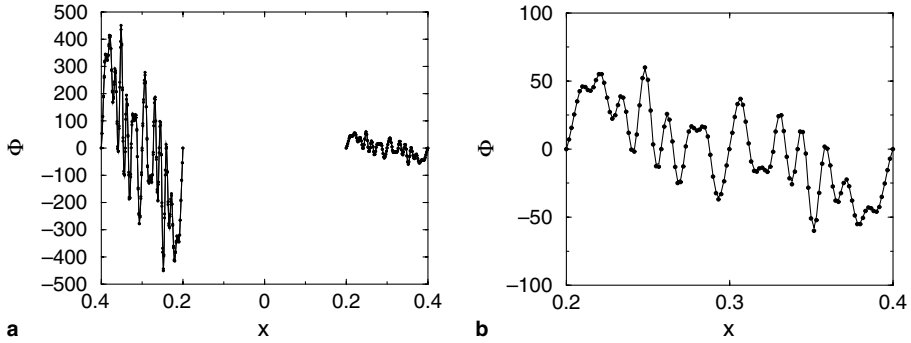


Fig. 6. The analytical profile of Eq. (23) and numerical solutions obtained from the FEM solver. The parameters taken are $l_{\max} = 20$, $M_{\max} = 20$, $\epsilon = 0.4$, $r_{\min} = 0.2$, and $r_{\max} = 0.4$. (a) The radial cut of the Φ profile along the x-axis (the annular simulating domain of $0.2 \leqslant r \leqslant 0.4$ is taken). (b) Expansion of (a) in $0.2 \leqslant x \leqslant 0.4$. The dots represent the numerical solution while the solid lines represent the mathematical profile Eq. (23). The two solutions match closely.

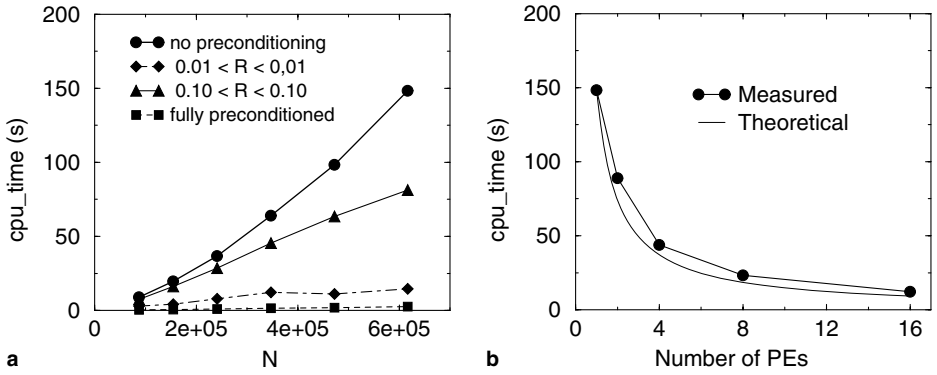


Fig. 7. (a) The CPU time versus the matrix size (number of grid points) in the finite element method using PETSc. (b) The CPU time versus the number of processors. The number of grid points $N = 616{,}320$ is used for the scaling. The timing of PETSc is quite promising.

Energy Research Supercomputing Center (http://www.nersc.gov). Note, the scaling of the thick solid line is obtained from a PETSc solve where the initial guess of Φ is taken as a zero vector.

In our particle simulation, since the numerical time step Δt is much shorter than the timescale of our interest (e.g., eddy turn over time, estimated by the drift wave frequency $\omega_{\star}$) we expect the solution to be similar to $\Phi^{(k-1)}(r,\theta)$, where the superscripts $(k-1)$ signify the previous time step in the simulation. The $\Phi^{(k-1)}(r,\theta)$ values can be used for the initial guess for solving $A \cdot \Phi^{(k)}(r,\theta) = d$. Such preconditioning of the solution $\Phi^{(k)}$ can speed up the computation. This latter timing is shown as dashed line in Fig. 7(a). The timing of the dashed line is trivially time required for one iteration where the solution Φ is employed for the initial guess. To see the effect of time evolution of the charge density in this test model, we manipulate the right-hand side of Eq. (21), d, by multiplying a random number by

$$d_{\mathrm{i}} = [1.0 + R] \times d_{\mathrm{i}}. \tag{24}$$

With this manipulation, we measure the change in CPU timing by varying the amplitude of the random number term R. The long dashed line in Fig. 7(a) is the case when R is taken within the range of $-0.01 \leqslant R \leqslant 0.01$, and a thin solid line is the case with $-0.10 \leqslant R \leqslant 0.10$. In both the cases, we observe a tendency that the preconditioning is effective at large N which is the size of the matrix. On the other hand, Fig. 7(b) shows the CPU time versus the number of processors. The default MPI (in PETSc) speeds up the computation proportional to the number of processors. The number of grid points $N = 616{,}320$ is used for the scaling.

As a reference, the computational timings for the GTC simulations for ITG runs (with the adiabatic approximation for the electrons) [21] are approximately 10 s per step for the particle pusher and approxi-

mately 1 s per step for the Poisson solver using 1024 processors on the IBM SP-3. The time for the field solver is approximately 10% of the total CPU time (due to the use of the original Poisson solver), where the adiabatic electron response is dominant [4]. A rough estimation of the timing with the new FEM Poisson solver is approximately 20 s per step with the same number of grid points, which is about 20 times slower than the original solver. Our goal is to speed up the new solver by a factor of 2, so that we spend equal time for the field solve and for the particle pushing. In the next section, the finite element solver is employed inside the GTC simulation, replacing the original iterative solver [5].

4. Linear and nonlinear simulation of ITG modes

In tokamak core regions, with auxiliary ion heating, the ion temperature gradient exceeds the density gradient and induces an instability known as the ITG mode [29] which is believed to be the drive for the core turbulence. Employing the new finite element solver, we first benchmark the linear growth rate of the ITG mode. We further analyze the nonlinear simulation results. For the ITG analysis, we employ a toroidal geometry but stay in a circular cross-section.

Shown in Fig. 8 is an example of a linear ITG eigenmode structure, a contour plot of Φ at a $\zeta = 0$ poloidal plane. Fig. 8 is obtained from linear runs, which means we select a specific toroidal eigenmode and filter out the others. Fig. 8(a) is obtained from the simulation with the finite element solver and Fig. 8(b) is from the standard GTC field solver. As a reminder, Eq. (6) is solved for Fig. 8(a) ($\tau = T_e/T_i = 1.0$ is taken). These are from two different series of simulations (using two different field solvers). From $t = 0$, we let the linear ITG grow. A linear mode with a toroidal mode number $n = 6$ is chosen, where the safety factor varies within the range of $1.53 \leqslant q \leqslant 3.58$ in the annular simulating domain, $0.2 \leqslant r \leqslant 0.4$. In Fig. 8, the number of grid points in each poloidal plane is $N = 9840$ (160 points on $r = 0.2$ and 320 points on $r = 0.4$). The new FEM solver gives the linear growth rate of $\gamma = 2.39 \times 10^{-4}$, while the original solver gives $\gamma = 2.32 \times 10^{-4}$. The linear growth rate falls within 3% of difference (with the Padé approximation). In Fig. 8(a), the scheme of Section 2.3 is employed (forwarding the triangle elements counter-clockwise). When the number of grid points is changed for the cases with the FEM solver ($N = 38{,}880$, 320 points on $r = 0.2$ and 640 points on $r = 0.4$), the growth rate stays $\gamma = 2.39 \times 10^{-4}$.

Results of nonlinear simulation, where all n modes are kept, are shown in Fig. 9. In the nonlinear regime, due to the mode coupling, multiple modes come into play and the plasma reach a turbulence state. Shown in Fig. 9 is a two-dimensional contour plot of Φ (with the zonal flow components subtracted). In Fig. 9, red represents positive Φ values, while blue fsrepresents negative Φ values. We observe vortex structures elongated in the radial direction. The vortex dynamics along the density and the temperature gradient give rise to net

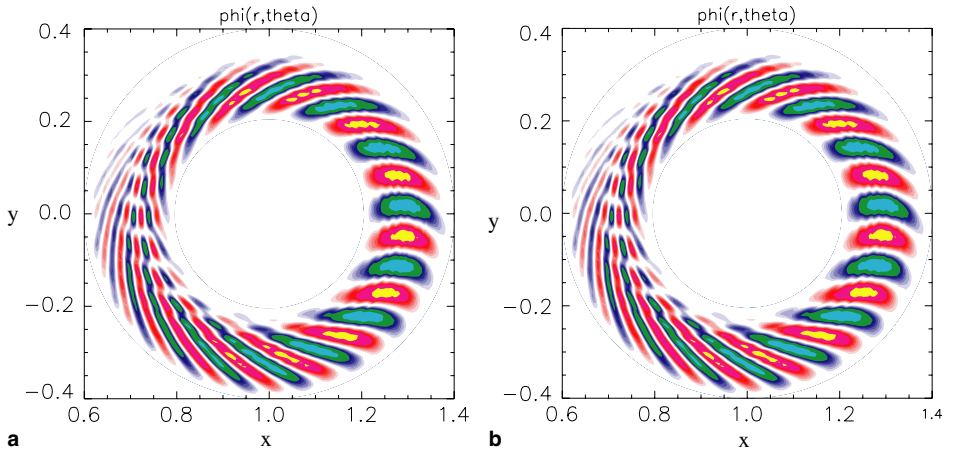
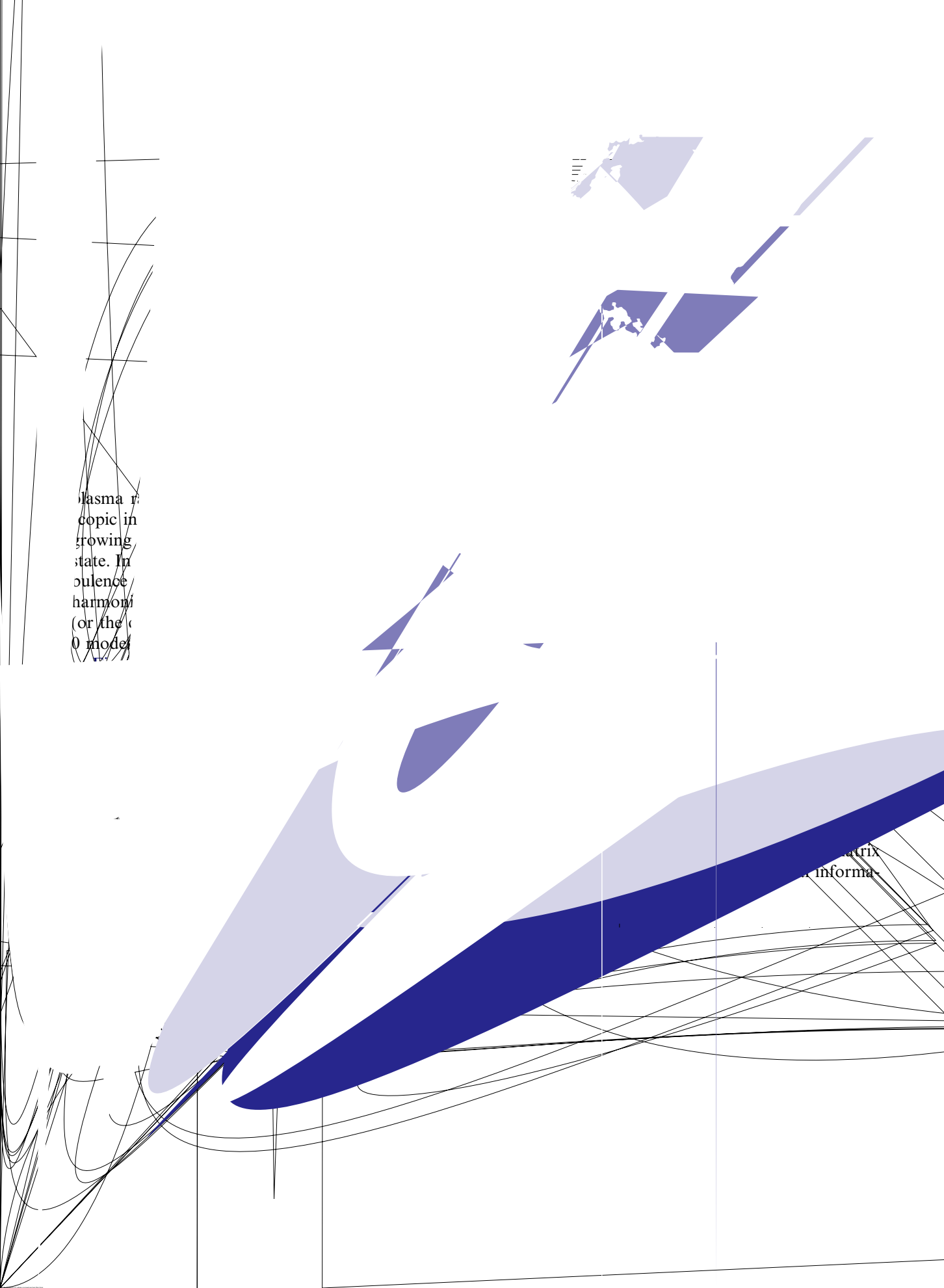


Fig. 8. A linear eigenmode (contour plot) of ITG instability (a) from the new FEM solver, (b) from the original iterative solver. A dominant mode of $m/n = 14/6$ is shown. The safety factor varies in the range of $1.53 < q < 3.58$ within the simulating domain.



tion, but independent of time) and only d is time-dependent (the right-hand side of the Poisson's equation). In the practical GTC runs with the finite element solver, the matrices A for different poloidal planes (32 to 64 in practice) are pre-generated at the beginning of the simulation. The matrices A are given only once at $t = 0$ as one of the input parameters.

As suggested by the analyses of Section 3, the preconditioning of the solution Φ speeds up the computation by employing the solution of the previous time step as an initial guess. In the GTC simulation, we employed $\Phi^{(k-1)}$ from the previous time step for the initial guess of the next time step $\Phi^{(k)}$ (the super-scripts $^{(k)}$ signify the time step in the simulation).

5. Summary and discussion

In this work, a new finite element field solver is developed and successfully implemented into the GTC code [21] which employs a logically non-rectangular grid. For the sparse matrix solve we employed the PETSc code [16]. Test cases for the matrix solver have shown optimistic CPU timing. The CPU timing scaled linearly versus the number of grid points (equivalent to $N \times N$, the size of the matrix). Application of the finite element field solver (together with the algebraic multigrid method) seems quite promising for large scale gyrokinetic turbulence simulations in the presence of kinetic electrons.

The benchmark ITG runs are also discussed. The simulation results using the new solver have been compared favorably with the results from the original field solver. The linear growth rate and the eigenmode structure matched closely, and in the nonlinear ITG simulations, the two gave the same transport level at the turbulence saturation state. The two different forms of the gyrokinetic Poisson equation, that are the integral form and the differential form with the Padé approximation generated similar results both in the linear and the nonlinear simulations.

The goal of our physics research is to investigate electron dynamics and electromagnetic effects [3] on the drift wave type turbulence. At a critical value of plasma pressure, we expect excitations of new branches of electromagnetic modes, that are the Alfvenic ion temperature gradient mode (AITG) and the kinetic ballooning mode (KBM) [31].

The field solver is expected to be more time consuming as the simulation domain (the number of grid points) becomes larger. We expect to further accelerate the computation, by employing AMG as a pre-conditioner to the matrix A [17]. In this work, the Poisson's equation is solved using one processor per poloidal plane. The details on the parallelization strategies will be discussed in our later publications. Second domain decomposition (within a poloidal plane) using MPI will be our future work.

Acknowledgements

The authors thank Dr. Wei-li Lee, for numerous advices and his encouragement throughout the course of this work. The authors also thank Dr. J. Breslau, Dr. J. Chen, Dr. L. Chen, Dr. T. Hahm, Dr. S. Klasky, Dr. J. Manickam, Dr. G. Rewoldt, Dr. W. Tang, and Dr. W. Wang for various useful comments. Y.N. thanks Professor D. Keyes of Columbia University, Dr. M. Knepley, and Dr. B. Smith at Mathematics and Computer Science Division of Argonne National Laboratory, for their kind assistance in the usage of PETSc. This work is supported by Department of Energy (DOE) Grant DE-FG02-03ER54724, Cooperative Agreement No. DE-FC02-04ER54796 (UCI), DOE Contract No. DE-AC02-76CH03073 (PPPL), and in part by SciDAC Center for Gyrokinetic Particle Simulation of Turbulent Transport in Burning Plasmas.

Appendix A. Numerical derivatives in a logically non-rectangular grid

As we discussed in Section 2 for the Padé approximation, we need to operate $\nabla_\perp^2$ on the charge density. Nominally, in a polar coordinate [22] $\nabla_\perp^2$ operator is given by a 5-point scheme

$$\nabla^2\sigma(r,\theta) = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\sigma}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2\sigma}{\partial\theta^2} = \frac{1}{r_i}\frac{1}{\Delta r}\left(r_{i+1/2}\frac{\sigma_{i+1,j}-\sigma_{i,j}}{\Delta r} - r_{i-1/2}\frac{\sigma_{i,j}-\sigma_{i-1,j}}{\Delta r}\right) + \frac{1}{r_i^2}\frac{\sigma_{i,j+1}-2\sigma_{i,j}+\sigma_{i,j-1}}{(\Delta\theta)^2}.$$

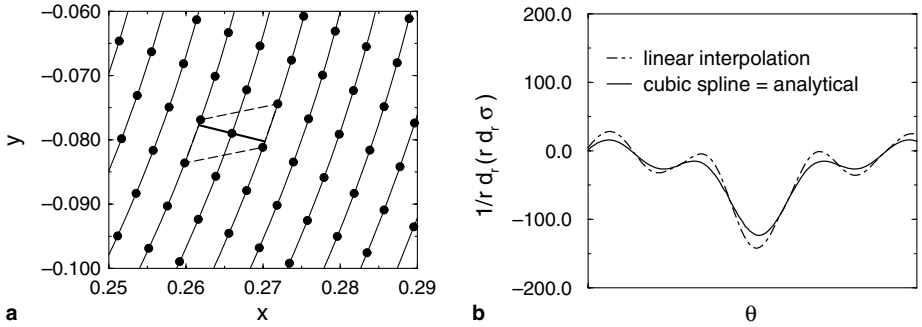


Fig. A.1. (a) The concept of the scheme in finding ghost points for the radial derivatives. The black dots are the grid points and the thin solid lines represent flux surfaces. The parallelogram with the dashed lines signifies the relevant four corners. The thick line connects two ghost points we need for the radial derivatives. (b) Comparison on the value of $1/r\partial_r(r\partial_r\sigma)$ obtained linear interpolation and cubic spline (which matches with the analytical profile).

where $0 \leqslant r \leqslant 1$ and $0 \leqslant \theta \leqslant 2\pi$. Here, i is the label for the radial coordinate r and j is the label for the azimuthal coordinate θ. For the radial derivative in the equation above, $\sigma_{i\pm1,j}$ does not have grid points (while for the θ derivative, $\sigma_{i,j\pm1}$ have corresponding grid points on a magnetic flux surface). To obtain the ghost points $\sigma_{i\pm1,j}$, we need an interpolation scheme (see Fig. A.1(a)). For the interpolation scheme, cubic spline is indispensable for accuracy.

Fig. A.1(b) compares the second derivative values from (1) a linear interpolation (2) the cubic spline and (3) the analytical profile. The values are plotted along the azimuthal coordinate. Here the analytical model of Eq. (23) is taken for σ with $\epsilon = 0.2$, $l_{\max} = 20$, $M_{\max} = 0$. The accuracy in the cubic spline reveals superiority over the linear interpolation.

References

[1] W.W. Lee, Phys. Fluids 26 (1983) 556.
[2] I. Manuilskiy, W.W. Lee, Phys. Plasmas 7 (2000) 1381.
[3] Z. Lin, L. Chen, Phys. Plasmas 8 (2001) 1447.
[4] Z. Lin, T.S. Hahm, W.W. Lee, W.M. Tang, R.B. White, Science 281 (1998) 1835.
[5] Z. Lin, W.W. Lee, Phys. Rev. E 52 (1995) 5646.
[6] D.H.E. Dubin, J.A. Krommes, C. Oberman, W.W. Lee, Phys Fluids 26 (1983) 3524.
[7] W.W. Lee, J. Comput. Phys. 72 (1987) 243.
[8] W.W. Lee et al., Phys. Plasmas 8 (2001) 4435.
[9] J.L.V. Lewandowski, Phys. Plasmas 10 (2003) 3204.
[10] A.M. Dimits, Phys. Rev. E 48 (1993) 4070.
[11] B. Scott, Phys. Plasmas 6 (1998) 2334.
[12] Z. Lin, T.S. Hahm, Phys. Plasmas 11 (2004) 1099.
[13] K.H. Huebner, D.L. Dewhirst, D.E. Smith, T.G. Byrom, The Finite Element Method for Engineers, fourth ed., Wiley, New York, 2001, p. 43.
[14] N. Tosaka, K. Ohnishi, Numerical Simulation of Partial Differential Equations, University of Tokyo Press, Tokyo, 1991, p. 102 (in Japanese).
[15] J.M. Dawson, Phys. Fluids 5 (1962) 445.
[16] PETSc: Portable, Extensible Toolkit for Scientific Computation, Mathematics and Computer Science Division, Argonne National Laboratory. Available from: <http://www-unix.mcs.anl.gov/petsc/petsc-2>.
[17] *hypre*: high performance preconditioners; Scalable Algorithms Group, Center for Applied Scientific Computing. Available from: <http://www.llnl.gov/CASC/hypre>.
[18] V.E. Henson, U.M. Yang, Appl. Numer. Math. 41 (2002) 155.
[19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in Fortran, second ed., Cambridge University Press, London, 1992, p. 194.
[20] T.S. Hahm, L. Chen, Phys. Fluids 28 (1985) 3061.
[21] Z. Lin, T.S. Hahm, W.W. Lee, W.M. Tang, R.B. White, Phys. Plasmas 7 (2000) 1857.
[22] J.C. Strikwerda, Finite Differencing Schemes and Partial Differential Equations, Chapman & Hall, New York, 1989, p. 291.
[23] H.R. Strauss, D.W. Longcope, J. Comput. Phys. 147 (1998) 318;
J. Breslaru, J. Chen, private communication, 2003.

[24] J. Manickam, private communication, 2003.
[25] Note in the nominal Clebsch coordinate, conventionally $\mathbf{B} = \nabla\alpha \times \nabla\beta$. See W.D. D'haeseleer, W.N.G. Hitchon, J.D. Callen, J.L. Shohet, Flux Coordinates and Magnetic Field Structure, Springer, Berlin, 1991, p. 100, for example, β representing the helical angle. In the GTC frame work, we denote α for the helical angle (instead of β) to avoid confusion with the plasma pressure β.
[26] W. Hackbusch, Multi-Grid Methods and Applications, Springer, New York, 1985, p. 93;
J.W. Ruge, K. Stuben, Algebraic multigrid, in: S.F. McCormick (Ed.), Multigrid Methods, SIAM, Philadelphia, PA, 1987.
[27] S. Chandrasekhar, Hydrodynamic and Hydromagnetic Stability, Clarendon Press, Oxford, 1961, p. 507.
[29] B. Coppi, Phys. Fluids 10 (1967) 582.
[30] A. Hasegawa, M. Wakatani, Phys. Rev. Lett. 59 (1987) 1581.
[31] G. Zhao, L. Chen, Phys. Plasmas 9 (2002) 861.