Neural network assisted electrostatic global gyrokinetic toroidal code using cylindrical coordinates

Jaya Kumar Alageshan[1,†], Joydeep Das[1], Tajinder Singh[1], Sarveshwar Sharma[2,3], Manjunatha Valmiki[4], Ashish Ranjan[4], Sandeep Agrawal[4], Sanjay Wandhekar[4], Animesh Kuley[1,*]

[1]Department of Physics, Indian Institute of Science, Bangalore 560012, India
[2]Institute for Plasma Research, Bhat, Gandhinagar-382428, India
[3]Homi Bhabha National Institute, Anushaktinagar, Mumbai, Maharashtra 400094, India
[4]Centre for Development of Advanced Computing, Pune-411005, India

E-mail: [†]jayaka@iisc.ac.in,[*]akuley@iisc.ac.in

June 2026

Abstract.
Gyrokinetic simulation codes are used to understand the microturbulence in the linear and nonlinear regimes of the tokamak and stellarator core. The codes that use flux coordinates to reduce computational complexities introduced by the anisotropy due to the presence of confinement magnetic fields encounter a mathematical singularity of the metric on the magnetic separatrix surface and at the X-point. To overcome this constraint, we develop a neural network-assisted Global Gyrokinetic Code using Cylindrical Coordinates (G2C3) to study the electrostatic microturbulence in realistic tokamak geometries. In particular, G2C3 uses a cylindrical coordinate system for particle dynamics, which allows particle motion in arbitrarily shaped flux surfaces, including the magnetic separatrix of the tokamak. We use a particle locating scheme, which uses a box scheme or a neural network and iterative local search algorithm, for the charge deposition and field interpolation. G2C3 uses the field lines estimated by numerical integration to train the neural network in universal function approximator mode to speed up the subroutines related to gathering and scattering operations of gyrokinetic simulation. Finally, as verification of the capability of the new code, we present results from self-consistent simulations of linear ion temperature gradient modes in the core region of the DIII-D tokamak.

Keywords: Tokamak, PIC, Gyrokinetic, ITG, Neural Network, G2C3

1. Introduction

Plasma turbulence in the scrape-off-layer (SOL) driven by microinstabilities will play a crucial role in the plasma confinement and heat load to the tokamak wall. Also, understanding the parasitic absorption of radio frequency waves in the SOL region is still an open problem. It is believed that parametric decay instabilities will be a plausible cause for such absorption

[1, 2, 3, 4]. Microinstabilities, like ion-temperature-gradient (ITG) and trapped electron modes (TEM) are unstable due to the gradients in plasma temperature and density, and are known to drive robust turbulence activity[5]. It is a critical challenge to capture both edge and core regions in an integrated global simulation due to the shape complexity arising in the SOL region by the divertor and X point. Presently, simulations within the SOL are primarily performed with fluid and gyro-fluid codes based on Braginskii equations. Fluid codes, such as SOLPS [6] and BOUT++[7], are widely used for SOL and divertor modelling. The significant advantage of a fluid code is that it needs less computational effort than kinetic approaches. Due to the plasma-wall interaction, the plasma is much colder in the SOL region than the core and edge regions. Therefore, collisions play an essential role in the SOL and influence turbulent transport. So, the fluid treatments based on Braginskii's approach provides valuable insights into SOL turbulence. These codes keep only a few moments and, therefore, cannot fully capture kinetic effects such as trapped particles, nonlinear wave-particle interactions and suprathermal tail particles. Several discrepancies have been reported between the experimental observations and fluid-based simulations [8, 9]. Unlike fluid approaches, the gyrokinetic simulation uses formulations that apply to a wide range of collisionality regimes even though the collisional mean free path is not small compared to the parallel scale length. Therefore, the fluid-based transport code will remain important for the boundary plasma model. In the fluid simulations, the flux coordinate independent approach (FCI) has been considered an efficient numerical method to study turbulence in the SOL region of realistic X-point geometries[10]. The global gyrokinetic simulation results are also expected to provide a better understanding of boundary plasma simulation and validate the experimental observations. Another promising code Gkeyll [11], based on the discontinuous Galerkin algorithm, has been recently applied to study the curvature-driven turbulence in the open field line region and plasma wall interactions. Also, it is extended to the nonlinear electromagnetic simulations in a helical open field line system using National Spherical Torus Experiment (NSTX) parameters [12].

In the last three decades, our understanding of the microturbulence in the tokamak core region has vastly improved, thanks to the development of several gyrokinetic simulation codes such as GTC [13, 14, 15, 16], GYRO [17], C-GYRO [18], ORB5 [19], GENE [20], etc.

These codes use flux coordinates, which at the magnetic separatrix surface and at the X-point (B_{pol} goes to zero) encounter a mathematical singularity in the metric. To circumvent this problem, we have developed a new simulation code called G2C3 (Global Gyrokinetic Code using Cylindrical Coordinates), similar in spirit to the XGC-1 [22], GTC-X[23] and TRIMEG [24]. XGC, TRIMEG and G2C3 are particle-in-cell (PIC) codes based on the Lagrangian approach and the fundamental cylindrical coordinate system. Avoiding the flux coordinate system allows G2C3, XGC, and TRIMEG to perform the gyrokinetic simulations in arbitrarily shaped flux surfaces, including separatrix and X point in the tokamak. XGC and G2C3 use field-aligned gather–scatter operations to achieve the efficiency of field-aligned meshes. Both G2C3 and XGC employ field-aligned meshes constructed in cylindrical coordinates, which are essential for accurately representing particularly in complex geometries. However, the required field-line resolution and particle number can become

substantially large depending on the device configuration. Consequently, performing charge scattering and field gathering for all particles during every PIC cycle in a realistic tokamak becomes computationally demanding in cylindrical coordinates aligned with field lines. To address this challenge, G2C3 explores neural-network-based representations of the projection operator, together with other optimization strategies.

In the present implementation, the neural network is used to construct the projection operator, $\mathscr{N}_{\parallel}$, which projects particles from the three-dimensional bulk plasma onto two-dimensional poloidal planes along magnetic field lines. This operation enables the magnetic anisotropy to be captured accurately even when a relatively sparse toroidal grid (i.e., a limited number of poloidal planes) is employed. In contrast, XGC mitigates projection-related numerical diffusion errors by using a substantially larger number of poloidal cross-sections together with particle–mesh spline interpolation.

Its also important to note that the numerical requirements differ among gyrokinetic codes. For example, GTC employs straight-field-line coordinates, where the projection reduces to a comparatively simple linear operation. TRIMEG uses a Fourier representation with a single poloidal plane and a fixed toroidal mode number, eliminating the need for the projection operator considered here. Recently, the GENE code was also updated to GENE-X to incorporate the SOL region based on a locally field-aligned coordinate system following flux coordinate. Continuum codes such as GENE-X do not involve particles and therefore avoid particle projection entirely. Furthermore, the FCI methodology adopted in GENE-X and GRILLIX [10] is based on finite-difference discretization on local cartesian grids and is designed to avoid reliance on magnetic flux surfaces, making it particularly suitable for complex three-dimensional magnetic geometries and stochastic magnetic-field regions. The main focus of the development of G2C3 is to couple the core and SOL regions to understand electromagnetic turbulence using the guiding center particle dynamics.

Neural network/machine learning methods have been employed in fusion research to predict and control disruptions in discharges using experimental data[26, 27], as a diagnostic tool to infer physical quantities using data obtained from simulations [28], to replace the computationally expensive kinetic PIC simulations with reduced or surrogate models [29], and to model the collision operators [30, 31]. For the first time, G2C3 incorporates machine learning techniques within the global PIC simulation. G2C3 uses a supervised multi-layered neural network to perform interpolation along the magnetic field lines, with training data obtained via numerical integration. This helps to perform scatter and gather operations. This constitutes one of the primary focuses of G2C3, highlighting the use of neural networks for efficient gather and scatter operations along the parallel direction. Furthermore, in G2C3, we have also extended this neural network module on the poloidal plane for searching the particle in addition to the standard box scheme.

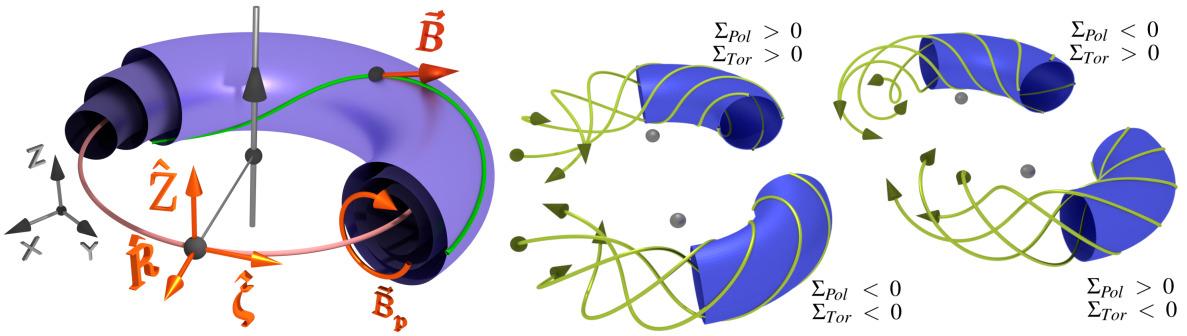
Some salient features of our approach include: (a) The neural-network-based projection operator enables G2C3 to capture magnetic anisotropy without requiring coordinate transformation; (b) The projection allows all poloidal planes to share an identical grid and mesh structure, thereby preserving the toroidal symmetry of the tokamak; (c) Since each poloidal plane is treated equivalently, toroidal periodicity is incorporated naturally; (d) By

utilizing the projection operator, the dependence on the safety factor is eliminated, enabling G2C3 to perform global simulations encompassing both the magnetic axis and the X-point; (e) The sparse toroidal grid results in a larger number of particles per toroidal domain, thereby reducing particle-induced noise during charge deposition.

The work related to the ITG mode for the core region of DIII-D is presented here is electrostatic and provides only a first step towards achieving a desirable whole plasma device simulation from the magnetic axis to the wall (scrape-off layer) including X-point. G2C3 reads the equilibrium fitting (EFIT) [32] and IPREQ [33] data files generated from experimental discharges. While the plasma ions are simulated with gyrokinetic marker particles, the electron response is assumed to be adiabatic in this study. Other microturbulence modes in the core and edge regions, such as TEM, kinetic ballooning modes, pressure-driven magneto-hydrodynamic modes, and others, will be added in future work.

In this article, we organize the material in the following form: after setting the conventions and details of the equilibrium data, we briefly describe the recipe to construct the simulation grids and the triangular mesh in Sec. 2; the neural network-based gather-scatter module and the triangle locator module are presented in Sec. 3, with error-analysis and convergence; the particle module in Sec. 4 describes the particle initialization and their governing dynamics; the finite element-based gyrokinetic Poisson solver to estimate the electric field is presented in Sec. 5. Finally, in Sec. 6 we describe how the modules are used within a PIC cycle to simulate and analyze ion temperature gradient-driven linear instability mode, with adiabatic electrons. The conclusions are presented in Sec. 7.

2. Equilibrium magnetic field, coordinates and conventions



(a) (b)

Figure 1. (a) Schematic of constant flux-function surfaces and a magnetic field line in different coordinate systems (Grey: Cartesian coordinate; Yellow: Cylindrical coordinate); (b) Configurations of magnetic field lines on a flux surface for different choices of Σ_{Pol} and Σ_{Tor}.

For an axisymmetric toroidal system, the poloidal flux function is symmetric in the toroidal direction, i.e., $\psi(R,\zeta,Z) = \psi(R,Z)$ such that ψ is minimum at the magnetic axis, and $\nabla \cdot \vec{\mathbf{B}} = 0$ implies the equilibrium magnetic field can be written in an orthonormal coordinate

system [see Fig. 1] as:

$$\vec{\mathbf{B}} = \vec{\mathbf{B}}_{Pol} + \vec{\mathbf{B}}_{Tor} = \Sigma_{Pol}\ (\nabla\psi(R,Z) \times \nabla\zeta) + \Sigma_{Tor}\ \left(F_{Pol}(\psi)\,\vec{\mathbf{e}}^{\zeta}\right), \tag{1}$$

where $F_{Pol}(\psi)$ is the poloidal current function, such that $F_{Pol}(\psi) > 0$; $\Sigma_{Pol} = sign\left(\vec{\mathbf{B}}_{Pol} \cdot (\nabla\psi \times \nabla\zeta)\right)$, $\Sigma_{Tor} = sign\left(\vec{\mathbf{B}}_{Tor} \cdot \vec{\mathbf{e}}^{\zeta}\right)$. Let the position vector in Euclidean space be

$$\vec{\mathbf{X}} = \{x, y, z\} = \{R\cos\zeta, R\sin\zeta, Z\} \tag{2}$$

$$\sigma^1 = R,\ \sigma^2 = \zeta, \sigma^3 = Z. \tag{3}$$

Therefore, the coordinate tangent vectors and the metric tensors are

$$\vec{\mathbf{e}}_i = \frac{\partial\vec{\mathbf{X}}}{\partial\sigma^i},\ g_{ij} = \frac{\partial\vec{\mathbf{X}}}{\partial\sigma^i} \cdot \frac{\partial\vec{\mathbf{X}}}{\partial\sigma^j} \tag{4}$$

By defining contravariant basis vectors $\vec{\mathbf{e}}^R = \nabla R$, $\vec{\mathbf{e}}^{\zeta} = \nabla\zeta$, $\vec{\mathbf{e}}^Z = \nabla Z$, and $\vec{\mathbf{e}}^i = g^{ij}\,\vec{\mathbf{e}}_j$, the metric component for this orthogonal basis, velocity, and magnetic field can be written as

$$g_{RR} = 1,\ g_{\zeta\zeta} = R^2,\ g_{ZZ} = 1,\ \text{and}\ g_{ij} = 0,\ \text{if}\ i \neq j \tag{5}$$

$$\vec{\mathbf{v}} = v^R\vec{\mathbf{e}}_R + v^{\zeta}\vec{\mathbf{e}}_{\zeta} + v^Z\vec{\mathbf{e}}_Z = v_R\vec{\mathbf{e}}^R + v_{\zeta}\vec{\mathbf{e}}^{\zeta} + v_Z\vec{\mathbf{e}}^Z \tag{6}$$

$$\vec{\mathbf{B}} = \tilde{B}^R\,\vec{\mathbf{e}}_R + \tilde{B}^Z\,\vec{\mathbf{e}}_Z + \tilde{B}^{\zeta}\,\vec{\mathbf{e}}_{\zeta} = \tilde{B}_R\,\vec{\mathbf{e}}^R + \tilde{B}_Z\,\vec{\mathbf{e}}^Z + (R\,\tilde{B}_{\zeta})\,\vec{\mathbf{e}}^{\zeta} \tag{7}$$

Equations (1) and (7) provide the components of the magnetic field in cylindrical coordinates as

$$\tilde{B}_R = -\frac{\Sigma_{Pol}}{\mathscr{J}}\frac{\partial\psi}{\partial Z};\ \tilde{B}_Z = \frac{\Sigma_{Pol}}{\mathscr{J}}\frac{\partial\psi}{\partial R};\ \tilde{B}_{\zeta} = \Sigma_{Tor}\frac{F(\psi)}{R}, \tag{8}$$

where $\mathscr{J}$ is the Jacobian of the transformation, given by $\mathscr{J} = \sqrt{Det|g_{ij}|}$. The magnitude of the magnetic field is,

$$B = \sqrt{g^{ij}B_iB_j} = \sqrt{\tilde{B}_R^2 + \tilde{B}_Z^2 + \tilde{B}_{\zeta}^2} \tag{9}$$

Units and Normalization

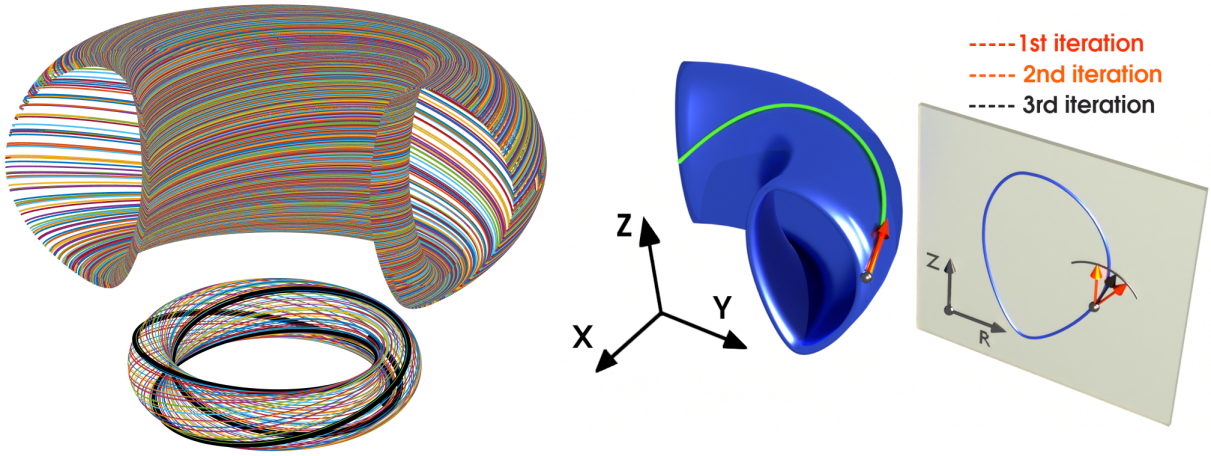
The basic units and normalization used in G2C3 are summarized as follows:

- Mass: proton mass, m_p
- Charge: proton charge, e
- Magnetic field: on axis, B_a
- Length: Tokamak major radius, R_0
- Density: on axis electron density, n_{e0}
- Temperature: on axis electron temperature, T_{e0}
- Time: Inverse on axis cyclotron frequency of proton, $\omega_p^{-1} = m_pc/eB_a$

2.1. Parameterizing the magnetic flux lines

In the presence of an external magnetic field, the plasma has a high degree of anisotropy, with vastly different length and time scales along the magnetic field lines and perpendicular to it. So to maintain the separation of scales and simplify calculations, it is convenient to define a coordinate system that encodes the field line geometry. We build the simulation grids which consider the anisotropy to enable efficient computation. The fields typically vary slowly in the $\|$-direction compared to the perpendicular directions. The magnetic field's global structure can be derived by integrating the differential equation of a curve which follows a magnetic field line. The local tangent to the magnetic field $d\mathscr{C}_{\mathbf{B}}(s)/ds$, considering $\mathscr{C}_{\mathbf{B}}(s)$ to be the trajectory along a magnetic field, can be written as:

$$\left.\frac{d\mathscr{C}_{\mathbf{B}}(s)}{ds}\right|_{\text{along }\vec{\mathbf{B}}} = \frac{\vec{\mathbf{B}}}{B} = \hat{b} \tag{10}$$



(a) (b)

Figure 2. (a) A sectional (top) and complete (bottom) view of flux lines lying on the same flux surface, $\psi = 0.28$; (b) a schematic of how the numerically estimated field lines are constrained to lie on the flux surface and the iterative scheme we use to project the iteration points onto the flux surface, given by Eq.(18).

We parametrize the flux line/curve using its natural length s, such that $ds^2 = dx^2 + dy^2 + dz^2$. Then the flux-line is given by: $\mathscr{C}_{\mathbf{B}}(s) = \{x(s),\, y(s),\, z(s)\}$. If $\vec{\mathbf{B}}(\mathbf{X})$ is the magnetic field in Euclidean 3d space, i.e. $\mathbf{X} \in \mathbb{R}^3$, then the magnetic flux line is the integral curve corresponding to $\hat{b}$ and this curve through a point $\mathbf{X}_0$ is given by:

$$\mathbf{X}(s) = \mathbf{X}_0 + \int_0^s \hat{b}\,(\mathbf{X}(\tilde{s}))\; d\tilde{s} \tag{11}$$

The equation above for the flux line can be solved numerically by discretizing s and using Euler's method as,

$$\mathbf{X}(s_i) = \mathbf{X}(s_{i-1}) + \Delta s\;\; \hat{b}\,(\mathbf{X}(s_{i-1}))$$

where $\Delta s = (s_i - s_{i-1})$. Figure 2(a) shows the numerically estimated trajectories for the DIII-D flux profiles using the Euler scheme (and the ψ-invariance scheme described in Sec. 2.2),

starting from all grid points of a given flux surface, $\psi = 0.28$ at $\zeta = 0$. To numerically study the plasma in a tokamak, the computational space is divided into a discrete set of poloidal planes, specified by $\zeta_i = i\,\Delta\zeta$, where $i \in \{1, 2, ...N_p\}$, $\Delta\zeta = 2\pi/N_p$, and N_p is the number of poloidal planes. Any computation on points in between these poloidal planes, the point is projected onto the nearest poloidal planes along the flux line given by Eq. (11). But the estimation of s on the ζ_i's is non-trivial. If $d\zeta/ds \neq 0$ at all points along the flux line, we can simplify the estimation of projected points by parametrizing the curve using ζ. Therefore,

$$\frac{d\mathbf{X}}{d\zeta} = \frac{ds}{d\zeta}\,\frac{d\mathbf{X}}{ds} = \frac{ds}{d\zeta}\,\hat{b}, \tag{12}$$

so the Euler scheme based linear projection operator that projects coordinates onto ζ_i poloidal plane, $\mathscr{P}_{\zeta_i}$ is given by

$$\begin{aligned}\mathscr{P}_{\zeta_i}\mathbf{X} &= \mathbf{X} + \int_{\zeta}^{\zeta_i} \frac{d\mathbf{X}}{d\zeta}\,d\zeta, \\ &\approx \mathbf{X} + (\zeta_i - \zeta)\,\left.\frac{d\mathbf{X}}{d\zeta}\right|_{\zeta}\end{aligned} \tag{13}$$

Also,

$$\begin{aligned}\hat{b} &= \frac{d\mathbf{X}}{ds} = \{\, B_x/B\,,\, B_y/B\,,\, B_z/B\,\} \\ &= \{\, \frac{dR}{ds}\,\cos\zeta - R\sin\zeta\,\frac{d\zeta}{ds}\,,\, \frac{dR}{ds}\,\sin\zeta + R\cos\zeta\,\frac{d\zeta}{ds}\,,\, \frac{dZ}{ds}\,\}\end{aligned} \tag{14}$$

Comparing the first two components we get,

$$\left.\begin{aligned}\frac{d\zeta}{ds} &= (B_y\,\cos\zeta - B_x\,\sin\zeta)/(RB) \;=\; (\hat{\mathbf{e}}_\zeta.\vec{\mathbf{B}})/(RB) \;=\; \frac{\tilde{B}_\zeta}{RB}, \\ \frac{dR}{ds} &= (B_x\,\cos\zeta + B_y\,\sin\zeta)/B \;=\; (\hat{\mathbf{e}}_R.\vec{\mathbf{B}})/B \;=\; \frac{\tilde{B}_R}{B}, \\ \frac{dZ}{ds} &= \frac{\tilde{B}_Z}{B}.\end{aligned}\right\} \tag{15}$$

Assuming $d\zeta/ds \neq 0$ (i.e. magnetic field is never purely poloidal), we can write $ds/d\zeta = (d\zeta/ds)^{-1}$ and $\Delta s \approx (R\,B/B_\zeta)\Delta\zeta$. Therefore,

$$\left.\begin{aligned}R \;&\rightarrow\; R + \Delta s\,\frac{dR}{ds} = R + \Delta\zeta\,\frac{R\,B_R}{\tilde{B}_\zeta}, \\ Z \;&\rightarrow\; Z + \Delta s\,\frac{dZ}{ds} = Z + \Delta\zeta\,\frac{R\,B_Z}{\tilde{B}_\zeta}, \\ \zeta \;&\rightarrow\; \zeta + \Delta s\,\frac{d\zeta}{ds} = \zeta + \Delta\zeta, \\ s \;&\rightarrow\; s + \Delta\zeta\,\frac{R\,B}{\tilde{B}_\zeta}.\end{aligned}\right\} \tag{16}$$

2.2. Invariance of ψ along the field lines

As indicated in the schematic of the flux-surface in Fig. 2(a), the field line should lie on the same flux surface, *i.e.* $\nabla_{\parallel}\psi = 0$. Conversely, the flux-surface constrains the magnetic field lines. So we need to ensure that the newly estimated point on the discrete flux-line has the same ψ value. We impose this constraint at each update step in Eq. (16) by an iterative projection onto the flux-surface as shown in Fig. 2(b). We modify the update equation such that if

$$\alpha = \arctan(\Delta Z/\Delta R)\ , \ \text{ and } \ r = \sqrt{\Delta R^2 + \Delta Z^2}, \tag{17}$$

then

$$\left.\begin{aligned} \Delta R &\rightarrow r\,\cos(\alpha), \\ \Delta Z &\rightarrow r\,\sin(\alpha), \\ \Delta\psi &\rightarrow \left[\psi(R+\Delta R, Z+\Delta Z) - \psi(R,Z)\right], \\ \Delta\alpha &\rightarrow sign(\Delta\psi)\,\frac{|\Delta\alpha|}{2}, \\ \alpha &\rightarrow \alpha + \Delta\alpha. \end{aligned}\right\} \tag{18}$$

The above iteration is performed with initial $\Delta\alpha = \pi/2$, until $|\Delta\psi|$ is less than a predefined cut-off value. This ensures that the field lines do not deviate by more than the cut-off from the starting ψ at all iterations.

2.3. Grids for the core region

We normalize and redefine ψ generated using IPREQ [33] and EFIT [32] such that $\psi(R,Z) \geq 0$ and $\psi(R_0,Z_0) = 0$, where (R_0,Z_0) is the position of the magnetic axis. Let $\psi_\times$ be the ψ-value along the flux surface through the X-point. Then we consider the ~~annular region with $0 < \psi_1(R,Z) < \psi_{m_\psi}(R,Z) < \psi_\times$ as the~~ core region, i.e. $0 \leq \psi(R,Z) < \psi_\times$. Here we describe the scheme to generate a flux surface following grid. First, we create a grid on the outer mid-plane starting from grid points at $(R_0 + n\,\Delta R, Z_0)$, where $n \in \{1,2,\cdots,m_\psi\}$, and $\Delta r = (R_{m_\psi} - R_0)/m_\psi$, such that $\psi(R_{m_\psi},Z_0) = \psi_{m_\psi}$, with m_ψ number of flux surfaces. Now starting with initial position (R_i,Z_0) we estimate the grid points on the $\psi = \psi(R_i,Z_0)$ flux surface using the scheme described in Sec. 2.1 and Sec. 2.2. Let (R_{ij},Z_{ij}) be the j-th grid point along s on the i-th flux surface and Δs_{sim} be the preferred grid point spacing along s, i.e.

$$\Delta s_{sim} = \sqrt{(R_{ij} - R_{i(j+1)})^2 + (Z_{ij} - Z_{i(j+1)})^2}.$$

However, various flux surfaces have different closed-loop lengths and could lead to unevenly spaced first and last grid points on the flux surface. To ensure that the grids are uniform on the flux surface, we redefine the preferred grid spacing using the following scheme: we first generate a high-resolution grid along s with $\Delta\tilde{s} = (\Delta s_{sim}/\mathbb{F}_s)$, where $\mathbb{F}_s > 1$ (see Fig.3(a)). We find the total number of grid points for a single loop of the i-th flux surface, $\tilde{m}_{\theta i}$ and estimate the required number of grid points as,

$$m_{\theta i} = \left\lfloor \frac{\tilde{m}_{\theta i}}{\mathbb{F}_s} \right\rceil, \tag{19}$$

where $\lfloor\cdot\rceil$ stands for the nearest integer. Then the preferred grid points are $(R_{ij},Z_{ij}) = (R_{i\tilde{j}},Z_{i\tilde{j}})$, where $\tilde{j} = \lfloor (j-1)(\tilde{m}_{\theta i}/m_{\theta i})\rfloor + 1$, and $\lfloor\cdot\rfloor$ is the floor function. Then the distance between the first grid point and the last grid point of the flux surface is

$$\Delta\tilde{s}\left(\tilde{m}_{\theta i} - \left\lfloor (m_{\theta i}-1)\frac{\tilde{m}_{\theta i}}{m_{\theta i}}\right\rfloor - 1\right) = \Delta\tilde{s}\left(\left\lfloor\frac{\tilde{m}_{\theta i}}{m_{\theta i}}\right\rfloor - 1\right)$$

and from Eq. (19), we have

$$\left(\frac{\tilde{m}_{\theta i}}{\mathbb{F}_s} - 1\right) \leq m_{\theta i} \leq \left(\frac{\tilde{m}_{\theta i}}{\mathbb{F}_s} + 1\right).$$

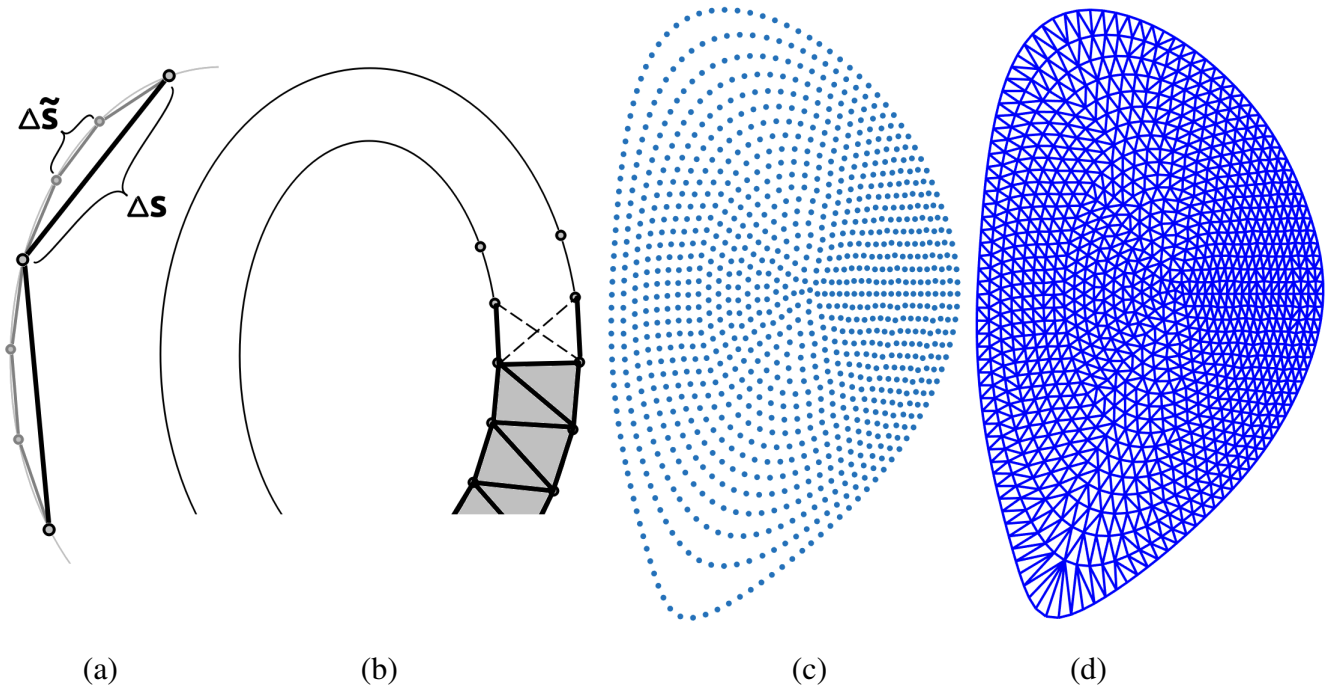


Figure 3. (a) Schematic of re-sampling based grid construction process; (b) Construction of triangles between two consecutive constant flux surfaces. Note that at every iteration there are two possible triangles. The option for which the sum of the lengths of its sides is smaller is chosen; (c) Grid generated in the core region ($0 \leq \psi \leq 0.99\psi_\times$), where $\psi_\times$ is the flux-function at the X-point, for the DIII-D configuration and the corresponding mesh is shown in (d).

Therefore, the error in the spacing between consecutive grid points is $\sim \Delta\tilde{s}$. Note that in R_{ij} and Z_{ij}, the dimension of the j index depends on i and hence is inconvenient to store as matrices. Instead, we define an array $(R_\alpha,Z_\alpha) = (R_{ij},Z_{ij})$, where, $\alpha = g_i + j$ and $g_i = \sum_{k=1}^{i-1} m_{\psi k}$.

2.4. Mesh construction for the core

In the case of closed flux surfaces, as in the core, we triangulate the annular region between the two neighbouring flux surfaces. For each annular region, we start from an edge that

connects the two different flux surfaces along the outer midplane. Then moving in the counter-clockwise direction we have two different choices for the triangle as shown in Fig. 3. We choose the one with a minimal perimeter and continue this process until all the triangles in the annular region are exhausted. The triangles are stored as a triplet of indices (α,β,γ) which are the labels/indices corresponding to the 3 vertices that constitute the triangle. Now for each annulus between flux surface indexed i and $(i+1)$, the number of triangles that this approach will construct is $(m_{\theta i}+m_{\theta(i+1)})$. Therefore, the total number of triangles that will be formed at the end of the process will be $n_\Delta=(m_\psi-1)(m_{\theta i}+m_{\theta(i+1)})$.

3. Field-aligned interpolation (gather/scatter)

The presence of a strong applied magnetic field renders the electromagnetic system anisotropic, and the gyro-centers of charged particles predominantly move along the magnetic field lines. Furthermore, the typical modes we expect to analyze using the G2C3 have $k_{||}\ll k_\perp$. We use this anisotropy to reduce computational complexity by using a fine grid in the poloidal plane, $\perp\!\!\!\perp$, and a coarse grid along the $||$-direction. To perform the first principle, self-consistent simulation we need the grid structure to calculate the fields using the Poisson solver and transfer data to (gather) and from (scatter) the particles using the following processes:

(i) $||$-*projection*: Move along the magnetic field lines and project the point on the neighbouring poloidal planes (Fig. 4(a));

(ii) *Triangle locator*: Localize the projected points to within a triangle from the 2d mesh;

(iii) $\perp\!\!\!\perp$-*interpolation*: Use area coordinates to perform 2d poloidal plane interpolation (Sec. 3.5, Fig. 4(b)).

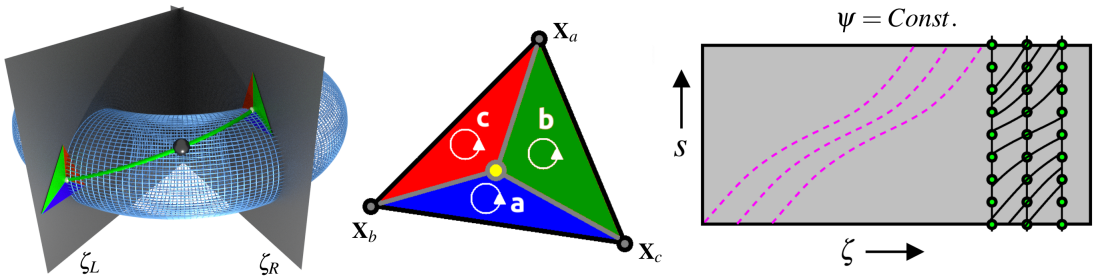


Figure 4. (a) A schematic of $||$-projection along the magnetic field lines on to the neighboring left- ($\zeta=\zeta_L$) and right- ($\zeta=\zeta_R$) poloidal mesh. The magnetic flux surface is plotted as a mesh only for the visual guide; (b) Area-coordinates visualization of a projected point relative to a triangle used for 2d ($\perp\!\!\!\perp$) interpolation in Sec. 3.5; (c) A schematic that shows the mesh structure in the $||$-direction on a flux surface, where the broken magenta curves represent the magnetic field lines.

Within the core region, where every point is uniquely specified by the flux surface and an angle in the poloidal plane, conventionally (i) is performed via transformation to Boozer coordinates

to reduce the computational cost. But the Boozer coordinates encounter singularity at the separatrix and fail to extend to the open field line region. We can circumvent this problem using numerical integration of Eq.(16), but it is iterative in nature and hence time consuming. Instead, in G2C3 we use a supervised neural network as a single step proxy integrator, with training data obtained from numerical integration. Similarly, we adopt the neural network to find the triangle in (ii). Before we detail the procedures to perform the above operations, first we present the operation of a neural network as a universal function approximator in Sec. 3.1. And then in Sec. 3.2 and Sec. 3.3 reformulate (i) and (ii) as a function approximation problem.

3.1. Neural network as universal function approximator

If y is a multivariate continuous function, then we use a fully-connected vanilla neural network [34] to approximate the function in the form,

$$\tilde{y}_i(\mathbf{x}) := \sum_{j=1}^{N_H} \left[W2_{ji}\ \boldsymbol{\sigma}\left(\sum_{k=1}^{N_I} W1_{kj}\, x_k + B1_j \right) \right] + B2_i \tag{20}$$

where σ is a non-linear function (we use tanh), referred to as *activation-function*. N_I is the dimension of the multi-variable input $\mathbf{x}$, N_H is the number of nodes in the hidden layer. $W1$ is a parameter matrix of dimension $N_I \times N_H$, and $W2$ of dimension $N_H \times N_O$, where N_O is the dimension of the output function $\tilde{y}$. And $B1$ is of size $N_H \times 1$ and $B2$ of size $N_O \times 1$. In general, the neural network acts as a map,

$$\mathscr{N} : \mathbf{x} \to \mathbf{y}$$

In the mathematical theory of neural networks and approximation theory, the *Universal Approximation Theorem* [35] establishes that such an approximation exists for large enough N_H, and is related to the *Kolmogorov–Arnold representation theorem* [36, 37]. Generically, the above function is diagrammatically represented in Fig. 5. The network structure, referred to as the network architecture, is such that it has three layers, namely: (i) input-layer with N_I nodes, (ii) hidden-layer with N_H nodes, and (iii) output-layer with N_O nodes. The network is fully connected, as each node of one layer is connected to all the nodes in the next layer. In general, we can have multiple numbers of hidden layers. The total number of unknowns in the approximating function is $\#_{parameters} = (N_H N_I) + (N_O N_H) + N_H + N_O$. Here, N_I and N_O are determined by the input and output dimensions, respectively, and N_H is chosen via convergence study.

Given the network architecture, we need to solve for the unknown parameters $\{W1, W2, B1, B2\}$. Typically, the number of unknowns is large, rendering the problem ill-defined. We will regularize the problem and formulate the process of determining the parameters as an optimization problem similar to the curve-fitting problem described in Sec. 3.1.2, referred to as *training*. But before training the network, we process the input and output data for the best fit.

3.1.1. Pre-processing the data: The function y to be approximated is evaluated at finitely many points on the input domain, such that it densely spans the domain. Each input, x_i, is

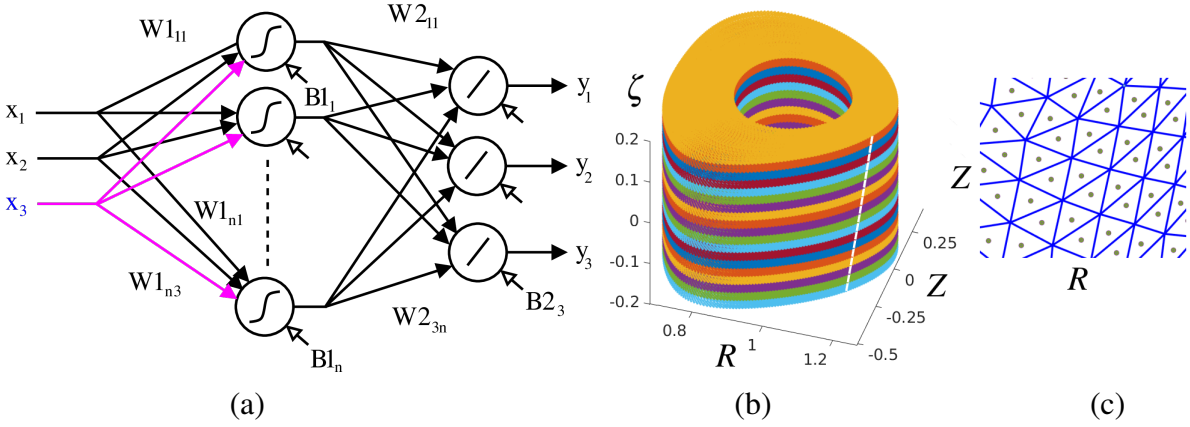


Figure 5. (a) Diagrammatic representation of Eq. 20. Here, $N_I = 3(2)$, $N_H = n$, $N_O = 3$. The curves within the nodes indicate the form of the activation function, such as the hidden layer has tanh, and the output layer has a linear activation functions. The plot in (b) shows the data points generated by solving the ODE in Eq. 16 for the field lines, starting from the grid points. The different colored points indicate discrete planes of ζ, and the white curve shows a schematic of a field line. (c) Shows the training data for the triangle locator problem, which takes triangle centroid coordinates as input, and the corresponding triangle label is the output.

normalized, such that their numerical values lie in $[0,1]$. Furthermore, the range of output is also standardized, such that

$$x_k \to \frac{x_k - x_k^{min}}{x_k^{max} - x_k^{min}} \quad ; \quad y_i \to \frac{y_i - y_i^{min}}{y_i^{max} - y_i^{min}}$$

3.1.2. Training: We formulate the problem of estimating the parameters in Eq. (20) as an optimization problem, where we need to find the best values of unknown parameters for which the function

$$\mathscr{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \sum_{i=1}^{N_O} \left[\tilde{y}_i(\bar{\mathbf{x}}) - y_i(\bar{\mathbf{x}})\right]^2,$$

referred to as *loss-function* is minimum. Now the minimization is performed numerically, in an iterative form, using the gradient descent method. Also, notice that, unlike a typical minimization problem, as in regression analysis, not all the data points are used at a time. The minimization will be performed one data point at a time, chosen at random, and is called *stochastic gradient-descent*. In general, the parameters W and B's are updated using the gradient descent scheme:

$$W_{ij} \to W_{ij} - \eta \frac{\partial \mathscr{L}}{\partial W_{ij}}, \quad B_i \to B_i - \eta \frac{\partial \mathscr{L}}{\partial B_i},$$

where $\eta \in [0,1)$ is the learning rate. In particular,

$$\left.\begin{aligned}
\frac{\partial \mathscr{L}}{\partial B1_i} &= \left(\sum_{j=1}^{N_O} W2_{ji}\frac{\partial \mathscr{L}}{\partial B2_j}\right)\ \mathrm{sech}^2\left(\sum_{k=1}^{N_H} W1_{ik}\,\bar{x}_k + B1_i\right), \\
\frac{\partial \mathscr{L}}{\partial B2_i} &= 2\ (\tilde{y}_i(\bar{\mathbf{x}}) - y_i(\bar{\mathbf{x}}))\,, \\
\frac{\partial \mathscr{L}}{\partial W1_{ij}} &= \frac{\partial \mathscr{L}}{\partial B1_i}\,\bar{x}_j, \\
\frac{\partial \mathscr{L}}{\partial W2_{ij}} &= \frac{\partial \mathscr{L}}{\partial B2_i}\ \tanh\left(\sum_{k=1}^{N_H} W1_{jk}\,\bar{x}_k + B1_j\right).
\end{aligned}\right\} \tag{21}$$

As per the stochastic gradient descent method, the training data are picked in random sequence, and the above update process is performed. An epoch corresponds to the operation where all the training data has been used once in the updating process. The training process is terminated when the gradients fall below a 0.5% error cut-off threshold.

3.1.3. Prediction mode: Once the training is complete, the weights W and B's are fixed. Now Eq. (20) gives the approximating function in the analytic form. The same equation can be used to predict the output for any input data in the trained domain and need not be from the training dataset.

The neural network training is part of setting up the simulation and does not contribute to the computational time of the main PIC loop. Furthermore, for a given magnetic equilibrium and simulation parameters/resolutions the neural network is trained once and the trained weights are saved and reused without retraining the network for the consequent runs. We algorithmically detect if the magnetic equilibrium and simulation parameters/resolutions have changed, by monitoring the prediction error (use a predetermined cut-off) to retrain the neural network.

3.2. $||$-Projection using neural network

We use a vanilla neural network, as described in Sec. 3.1, to perform the $||$-projection. Given the invariance of the $||$-projection in ζ in a tokamak, the neural network estimates $\delta\vec{\mathbf{X}}$, for a given $\delta\zeta$, as a map

$$\mathscr{N}_{||} : (R, Z, \delta\zeta) \longrightarrow (\delta R, \delta Z, \delta s)\ ,$$

where R, Z belong to the simulation domain and $\delta\zeta \in [-\Delta\zeta, \Delta\zeta]$.

To generate the training dataset, we estimate the points on the field lines using the method described in Sec. 2.1 thus constructing the map $\mathscr{N}(R_i, Z_i, n\,\delta\zeta)$, for $n \in \mathbb{Z}$ (set of integers), and (R_i, Z_i) belonging to the simulation grid points on a poloidal plane.

The input layer has 3 nodes, corresponding to R, Z, and $\delta\zeta$. We find by search with multiple of 10 nodes that 30 nodes in the hidden layer is optimal, and the output layer has 3 nodes corresponding to δR, δZ, and δs. Here we use the learning rate (η) of 0.001 and perform training for 1000 epochs [cf. Fig. 7(c)].

Next, we describe how to use a neural network to predict the triangle that encloses a point in 2d and present the performance of the current neural network in Sec. 3.7.1.

3.3. Triangle locator ~~using Neural network~~

To perform the 2D poloidal interpolation for a given a point (R,Z), first we need to find the triangle it belongs to. In G2C3, we use the box scheme [24] to identify the triangle. Alternatively, we augment this scheme by replacing the box look-up table by a neural network-based map.

3.3.1. Box scheme: To reduce the search space, we locate the particle within a rectangular grid in Cartesian (R,Z) space using the box method as,

$$i_{Box} = \left\lfloor \frac{R - R_{min}}{dR_{Box}\,(R_{max} - R_{min})} \right\rfloor, \quad j_{Box} = \left\lfloor \frac{Z - Z_{min}}{dZ_{Box}\,(Z_{max} - Z_{min})} \right\rfloor.$$

Here, (R_{min}, R_{max}) and (Z_{min}, Z_{max}) refers to the range of R and Z values of the triangular mesh. The dR_{Box} and dZ_{Box} are parameters that are chosen to be $\sim \sqrt{2\bar{A}_\Delta}$, where $\bar{A}_\Delta$ is the average area of the triangles.
Now, we construct a look-up table/array

$$\mathscr{B} : (i_{Box}, j_{Box}) \to T$$

that takes the given box-index to one of the triangles that encloses it. 3.4 ~~to find the correct triangle.~~ If the box is within only one triangle, we have located the triangle and T is taken as positive. When a box belongs to more than one triangle $\mathscr{B}$ maps to one of the overlaping triangle label with negative value. Then when $\mathscr{B}$ maps to negative label we use the iterative scheme in Sec. 3.4 to find the correct triangle.

3.3.2. Neural network-based scheme: Here our goal is to replace the look-up table $\mathscr{B}$ with a neural network map. Now, this map is enabled by partial ordering of the mesh system, namely:

- The grid points lie on constant flux functions;
- The mesh is such that all the grid points on a flux surface are connected;
- The triangles are labelled such that we start from the triangle along the outer mid-plane on the innermost flux surface and increases along the counter-clockwise direction (thus acting as a measure of flux-surface length);
- When the labelling of a flux surface is complete we move on to the next (outer) flux surface and continue the labelling scheme.

The partial order of the mesh enables us to map the mesh into the (ψ,θ), where the θ acts as a proxy for the length of the flux surface and also assists with the periodicity. Since we know the number of triangles in each flux surface annulus, we construct 2D indexing, (i_ψ, i_θ), where $1 \le i_\psi < (m_\psi - 1)$ refers to the label of the constant flux contour, with m_ψ number of

flux surface, and i_θ refers to the index of the triangle within each annulus. To find the triangle in which a given point in the poloidal plane lies, we need to find a map

$$\mathscr{N}_\Delta : (R,Z) \to (i_\psi, i_\theta)$$

But this map is not smooth, as the index i_θ has a jump across the outer mid-plane line because of 2π periodicity. To eliminate the jump we reparametrize the label i_θ into two variables $i_{c\theta} = \cos(2\pi i_\theta/n_\psi)$, $i_{s\theta} = \sin(2\pi i_\theta/n_\psi)$, where n_ψ is the number of triangles in the annular region. Hence, our goal is to find a smooth function

$$\tilde{\mathscr{N}}_\Delta : (R,Z) \to (\tilde{i}_\psi, \tilde{i}_{c\theta}, \tilde{i}_{s\theta})$$

such that

$$i_\psi := \lfloor \tilde{i}_\psi \rceil \quad \text{and} \quad i_\theta := \left\lfloor \frac{n_\psi}{2\pi} \arctan(\tilde{i}_{s\theta}, \tilde{i}_{c\theta}) \right\rceil ,$$

where $\lfloor . \rceil$ refers to the roundoff function. Then the triangle index is, $T = \left(\sum_{i<i_\psi} n_\psi[i] \; + i_\theta \right)$.

We use the neural network to estimate the function $\tilde{\mathscr{N}}_\Delta$. The input layer has two nodes, corresponding to R and Z, and the output layer has three nodes which correspond to $(\tilde{i}_\psi, \tilde{i}_{c\theta}, \tilde{i}_{s\theta})$. We use ten nodes in the hidden layer with a learning rate of $\eta = 0.01$, and perform training for ~ 1000 epochs. The training data is obtained by considering the triangle and the coordinates of the corresponding centroids and tested with randomly distributed points in the mesh. Both $\mathscr{B}$ and $\mathscr{N}_\Delta$ do not always map the particle location to the correct triangle (See Sec. 3.7.1 for error analysis). So, the next section describes an iterative algorithm to correctly locate the triangle.

3.4. Iterative triangle local search scheme

The mesh we use in the simulations is partially structured, so we incorporate an iterative local search scheme to track down the exact triangle in which the particle lies.

As shown in Fig. 4(b), if the point falls within a triangle, then all the area coordinates are positive. If the point lies outside the triangle, then at least one area coordinate will be negative. Thus we can build a locator algorithm that uses the area coordinates to check if the particle is within a triangular element, and use the signs to find a criterion to move to a neighbouring triangle in the mesh [38].

Note that the sign of the area coordinates has the information about on which side of the triangle the particle lies. For example, in Fig. 6 for the case in which the point lies outside the triangle, $c < 0$, implying that $\mathbf{X}$ is farthest from point $\mathbf{X}_c$. So we can shift to the neighbouring triangle element that shares the corresponding edge. This is accomplished using the triangle label map

$$\mathscr{T} : T_i \to (T_a, T_b, T_c)$$

Given a point (R,Z) and an initial guess for the triangle T_i, if the corresponding area coordinates are positive, then the point is within the triangle. Else, if for example $c < a$ and $c < b$, then the triangle label is updated to T_c using $\mathscr{T}$. The iteration is complete when all the area-coordinates are positive and the particle's triangle is correctly identified. The number

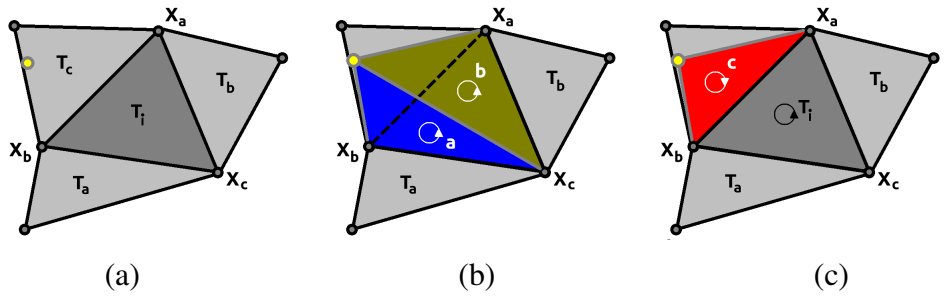


Figure 6. Scheme for locating a particle (yellow) w.r.t. the triangle T_i. Here (b) and (c) show the area coordinates when the particle is outside the triangle (T_i) and the arrows indicate the corresponding signs (anti-clockwise is positive). As shown in (b), for this case a-, b-coordinates are positive and the c-coordinate is negative, as in (c).

of iteration steps required depends on the initial accuracy of the triangle predicted by the box or the neural network scheme, as discussed in Sec. 3.7.1. Once the triangle is identified, we can proceed with the 2d interpolation scheme detailed in the next section.

3.5. $\perp\!\!\!\perp$-interpolation using area coordinates

The 2D triangular grids constructed in Sec. 2.3 form the poloidal planes ($\zeta = Const.$) and the mesh sizes are refined to resolve the $k_\perp$. Now, for any point within a triangle, we can linearly interpolate the quantities from the triangle vertices to the point using the area coordinates, as shown in Fig. 4(b). Here given a point (R,Z) in the poloidal plane and a triangle from the poloidal mesh with coordinates $\mathbf{X}_a$, $\mathbf{X}_b$, and $\mathbf{X}_c$, which are counter-clockwise in direction, then the corresponding signed areas a, b, c are:

$$a = \frac{1}{2A}\left[(R_bZ_c - R_cZ_b) + R(Z_b - Z_c) + Z(R_c - R_b)\right],$$
$$b = \frac{1}{2A}\left[(R_cZ_a - R_aZ_c) + R(Z_c - Z_a) + Z(R_a - R_c)\right],$$
$$c = \frac{1}{2A}\left[(R_aZ_b - R_bZ_a) + R(Z_a - Z_b) + Z(R_b - R_a)\right],$$

where, $A = (1/2)\left[(R_a - R_b)(Z_a - Z_c) - (R_a - R_c)(Z_a - Z_b)\right]$ is the area of the triangle, such that $a+b+c = 1$. If the point $\mathbf{X}$ lies within the triangle, then all three areas are positive. But if $\mathbf{X}$ is outside the triangle, then at least one of the triangle areas will be negative.

By definition, $\{R,Z\} = a\,\{R_a,Z_a\} + b\,\{R_b,Z_b\} + c\,\{R_c,Z_c\}$, and in general for any field F defined on the mesh, $F(R,Z) = a\,F_a + b\,F_b + c\,F_c$.

3.6. Field gathering (Grid field $\rightarrow$ Particle field)

Finally, we can put together the various processes described in this section to transfer data from the grid to the particle. For a particle located at (R,ζ,Z) with φ_L and φ_R being the data specified on the neighboring poloidal grid, first we $||$-project the points to $\mathbf{X}_L = (R_L,\zeta_L,Z_L)$ and $\mathbf{X}_R = (R_R,\zeta_R,Z_R)$. The projecting neural network also estimates the corresponding

arc-lengths s_L and s_R. Lastly, we locate the projected points and find the area coordinates (a_L, b_L, c_L) and (a_R, b_R, c_R). Then the value at the $||$-projected points are

$$\varphi_* = a_* \, \varphi_{a*} + b_* \, \varphi_{b*} + c_* \, \varphi_{c*},$$

where $* \in \{L, R\}$, such that,

$$\varphi = \frac{s_R \, \varphi_L + s_L \, \varphi_R}{s_L + s_R}. \tag{22}$$

The gathering operation helps us transform any 2D fields, which are defined on the grid points of poloidal planes, into the 3D field in the bulk of the tokamak compuatational region.

3.7. Scattering (Particle field → Grid field):

Let the particle position be (R, ζ, Z). We find the two nearest poloidal planes, $\zeta = \zeta_L$ and $\zeta = \zeta_R$, such that $\zeta_L \le \zeta \le \zeta_R$, and estimate $(\mathbf{X}_L, \mathbf{X}_R)$ via $||$-projection. Then the scattering of the field φ onto these points are given by

$$\varphi_L := \frac{s_R}{s_L + s_R} \, \varphi(\mathbf{X}); \;\; \text{and} \;\; \varphi_R := \frac{s_L}{s_L + s_R} \, \varphi(\mathbf{X}) \tag{23}$$

where s_L and s_R are the lengths along the flux-line from $\mathbf{X}_L$ and $\mathbf{X}_R$, respectively. Now, we use the triangle locator and area-coordinates to scatter the field value onto the nearest grid points in the poloidal plane as:

$$\varphi_{a*} = a \, \varphi_*, \;\; \varphi_{b*} = b \, \varphi_*, \;\; \varphi_{c*} = c \, \varphi_*,$$

where, $* \in \{L, R\}$. In particular, Sec. 5 uses scattering operation to estimate the charge density at the grid from the particle weights.

3.7.1. Neural network performance analysis: We visualize the performance of the trained neural network by generating the plot of the predicted value versus the expected value in Fig. 7(a) and (b) corresponding to the three outputs of $\mathscr{N}_{||}$ and $\tilde{\mathscr{N}_\Delta}$. Figure 7(c) shows the convergence behaviour of the neural networks using the plot of loss function evolution. ~~In particular, we find that the network can locate the triangle with a maximum error of three to four triangle distances from the correct triangle.~~

To analyze the efficacy of $\mathscr{N}_{||}$ in the gather-scatter operation, we initialize electric potential ϕ on each poloidal plane as (see Fig. 8(a))

$$\phi \;:=\; \sin(m\,\theta_B - n\,\zeta) \, \exp\left(-\frac{(q-q_0)^2}{\sigma_q^2} \right), \tag{24}$$

with $m = 6$, $n = 3$, $q_0 = 2$, and $\sigma_q = 0.15$ on each poloidal planes, where θ_B is the Boozer coordinate defined in Sec. 6.1. The corresponding error in interpolating the field onto the particle (gather) and then scattering it back onto the poloidal plane, which is effectively an identity function, is $\sim 3\%$ (see Fig. 8(b)). Figure 8(c) shows the variation of *rms* error in interpolating the field to intermediate poloidal planes, which on average is $\sim 0.6\%$.

The plots in Fig. 7(a) show the estimation error we encounter in the trained $\mathscr{N}_\Delta$ map for 50 flux surfaces and ~ 500 grid points in each flux surface. The maximum error of $\pm 2\%$

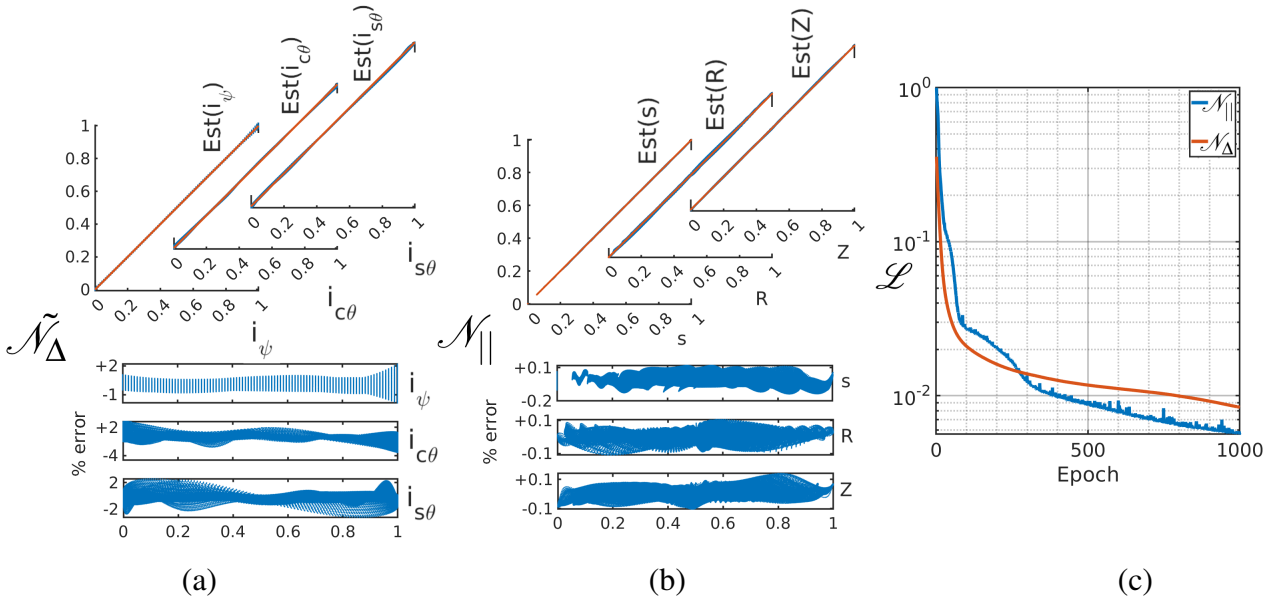


Figure 7. The expected Vs estimated output plots for the three outputs are shown in: (a) triangle locator, $\tilde{\mathscr{N}}_{\Delta}$ (see Sec. 3.3); (b) $||$-projector, $\mathscr{N}_{||}$ (see Sec. 3.2), where the best performance corresponds to the 45^o line. The corresponding %-errors are shown in the bottom rows. The plot in (c) shows the evolution of the loss function during training in the log scale for $\mathscr{N}_{||}$ and $\tilde{\mathscr{N}}_{\Delta}$.

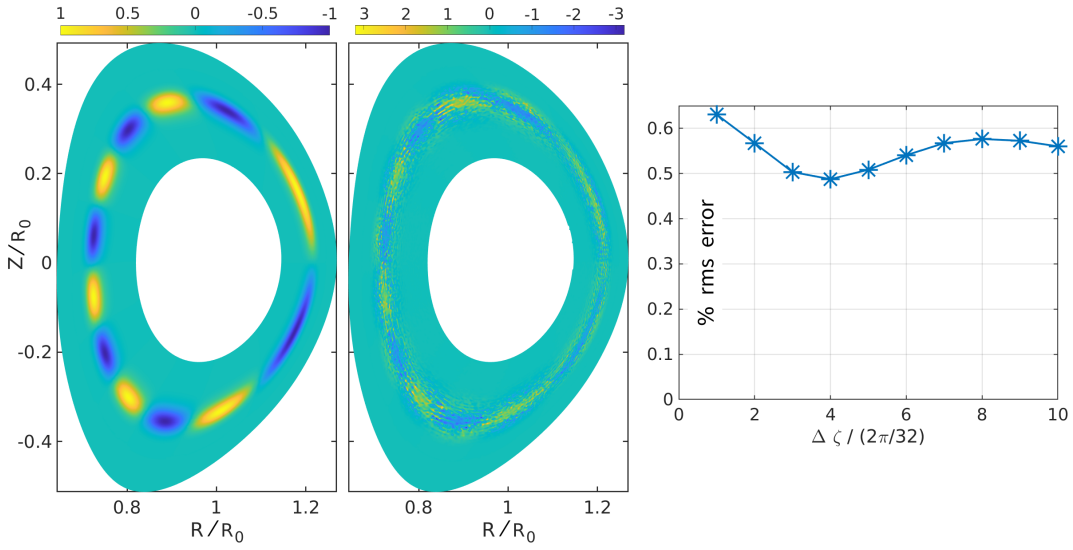


Figure 8. The contour plot in (a) shows the initialized analytic mode structure; Percentage error estimation for the gather-scatter operation using the neural network is shown in (b); The corresponding percentage rms error at various poloidal planes are shown in (c).

implies the estimated flux surface always lies in the immediate neighbourhood. Furthermore, the errors in $i_{c\theta}$ and $i_{s\theta}$ shown in Fig. 7(a) leads to the triangle being shifted on average by 3.35 triangles in comparison to 1.22 triangles for the box scheme, when erroneously classified.

And in terms of computational efficiency, the box scheme is ~ 6 times faster than the NN-approach when compared using 10^5 particles.

The training time and the resource requirement depend on the size of the machine and the simulation resolutions. For example, for the DIII-D case with 50 flux surfaces, $\sim 1.5\times10^4$ grid points, and $\sim 3.5\times10^4$ (N_Δ) triangles the locator uses N_Δ number of training data (corresponding to triangle centroid) and on a CPU (Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz) takes about 10 mins to train, with max. 10^3 epochs. The integrator uses $N_{Grid}\times 20$ ($N_{Grid}\sim 3.5\times10^4$) number of training data (with 20 data points along the $\|$-direction for each grid point) and takes about 40 mins to train, with max. 2×10^3 epochs.

The box scheme is both accurate and computationally efficient for locating the containing triangle. Furthermore, the box scheme extends to the SOL region naturally, whereas the NN-based method needs modification. We emphasize that the motivation for developing the NN-based triangle locator was not to improve upon the performance of the box scheme. Rather, it was intended to extend the existing neural network framework used in the projection operation and to investigate the feasibility and future extensibility of a unified NN-based approach for multiple particle-tracking operations.

4. Particle module (PDE $\rightarrow$ ODEs)

In this paper, we focus entirely on the electrostatic-collisionless processes with adiabatic electrons to observe the ITG modes. Appropriately, the dynamics of the ions are described by the Vlasov-Maxwell equations

$$\frac{\partial f_i}{\partial t}+\dot{\mathbf{x}}\cdot\nabla f_i+\dot{\mathbf{v}}\cdot\frac{\partial f_i}{\partial \mathbf{v}}=0 \quad \text{and} \quad \nabla\cdot\mathbf{E}=4\pi\rho\ , \tag{25}$$

where $\mathbf{v}=\dot{\mathbf{x}}$, and $f_i(\mathbf{x},\mathbf{v})$ is the ion distribution function in the 6D phase space. The above set of equations are solved self-consistently using

$$\dot{\mathbf{v}}=\left(\frac{q}{m}\mathbf{E}+\frac{q}{mc}\mathbf{v}\times\mathbf{B}\right) \quad \text{and} \quad \rho=q\int d^3v\,(f_i-f_e)\ , \tag{26}$$

where $\mathbf{B}$ is the applied external magnetic field and f_e corresponds to the Maxwellian distribution of the adiabatic electron.

We study micro-turbulence phenomena with time scales that are much larger than the gyro-motion time scales of ions and electrons, which enables us to integrate out the gyro-motion. We thus obtain an effective, computationally feasible, 5D gyro-kinetic phase space, $(R,\zeta,Z,v_{||},\mu)$, and the corresponding evolution equation is given by [39]

$$\frac{d}{dt}f_i=\frac{\partial f_i}{\partial t}+\dot{\bar{\mathbf{X}}}\cdot\nabla f_i+\dot{v}_{||}\frac{\partial f_i}{\partial v_{||}}=0\ , \tag{27}$$

where f_i now is the guiding centre distribution function.

In a gyro-kinetic PIC simulation, instead of evolving the 5D partial differential equation (PDE) for f_i we evolve a distribution of N_i marker particles, thus reducing the problem to $5N_i$ ordinary differential equations (ODE). The next section describes the ODE's and the corresponding initial conditions.

4.1. Particle push/dynamics

The evolution of f_i is captured by the following guiding centre equations of motion of N_i particles in 5D phase space as: [40, 41]

$$\dot{\vec{\mathbf{X}}} = v_{\parallel}\frac{\vec{\mathbf{B}}}{B^*_{\parallel}} + \vec{\mathbf{v}}_E + \vec{\mathbf{v}}_d, \qquad \dot{v}_{\parallel} = -\frac{1}{m_i}\frac{\vec{\mathbf{B}}^*}{B^*_{\parallel}}\cdot(\mu\nabla B + Z_i\nabla\phi), \tag{28}$$

and $\vec{\mathbf{B}}^* = \vec{\mathbf{B}} + Bv_{\parallel}/\omega_{ci}(\nabla\times\hat{b})$ is the equilibrium magnetic field at the guiding center position, $B^*_{\parallel} = \hat{b}\cdot\vec{\mathbf{B}}^*$, $\vec{\mathbf{v}}_E$ is the $\vec{E}\times\vec{B}$ drift velocity, and $\vec{\mathbf{v}}_d$ is the magnetic drift velocity due to curvature and gradient in magnetic field, which are given by

$$\vec{\mathbf{v}}_E = \frac{\hat{b}\times\nabla\phi}{B} \quad \text{and} \quad \vec{\mathbf{v}}_d = \vec{\mathbf{v}}_c + \vec{\mathbf{v}}_g = \frac{v_{\parallel}^2}{\omega_{ci}}\nabla\times\hat{b} + \frac{\mu}{m_i\omega_{ci}}\hat{b}\times\nabla B. \tag{29}$$

Now, if the equilibrium current is suppressed, then

$$\vec{\mathbf{v}}_d = \left(\frac{m_i\, v_{\parallel}^2 + \mu\, B}{m_i\, \omega_{ci}}\right)\left(\frac{\hat{b}\times\nabla B}{B}\right) \tag{30}$$

For an axisymmetric system, components of velocity in cylindrical coordinates are as follows:

$$\dot{R} = v_{||}\frac{B_R}{B^*_{||}} + \frac{c}{B^*_{||}}\left(\frac{B_\zeta}{B}\frac{\partial\phi}{\partial Z} - \frac{B_Z}{RB}\frac{\partial\phi}{\partial\zeta}\right) - \frac{Bv_{||}^2}{B^*_{||}\omega_{ci}}\frac{1}{\mathscr{J}}\frac{\partial}{\partial Z}\left(\frac{RB_\zeta}{B}\right) + \frac{\mu}{m_i\omega_{ci}}\frac{B_\zeta}{B}\frac{\partial B}{\partial Z}, \tag{31}$$

$$\dot{Z} = v_{||}\frac{B_Z}{B^*_{||}} - \frac{c}{B^*_{||}}\left(\frac{B_\zeta}{B}\frac{\partial\phi}{\partial R} - \frac{B_R}{RB}\frac{\partial\phi}{\partial\zeta}\right) + \frac{Bv_{||}^2}{B^*_{||}\omega_{ci}}\frac{1}{\mathscr{J}}\frac{\partial}{\partial R}\left(\frac{RB_\zeta}{B}\right) - \frac{\mu}{m_i\omega_{ci}}\left(\frac{B_\zeta}{B}\frac{\partial B}{\partial R}\right), \tag{32}$$

$$\begin{aligned}\dot{\zeta} = v_{||}\frac{B_\zeta}{B^*_{||}}\frac{1}{R} + \frac{c}{B^*_{||}\mathscr{J}}\left(\frac{B_Z}{B}\frac{\partial\phi}{\partial R} - \frac{B_R}{B}\frac{\partial\phi}{\partial Z}\right) - \frac{\mu}{m_i\omega_{ci}\mathscr{J}}\left(\frac{B_R}{B^*_{||}}\frac{\partial B}{\partial Z} - \frac{B_Z}{B^*_{||}}\frac{\partial B}{\partial R}\right)\\ + \frac{Bv_{||}^2}{B^*_{||}\omega_{ci}}\frac{1}{\mathscr{J}}\left[\frac{\partial}{\partial Z}\left(\frac{B_R}{B}\right) - \frac{\partial}{\partial R}\left(\frac{B_Z}{B}\right)\right],\end{aligned} \tag{33}$$

$$\begin{aligned}\dot{v}_{||} = &-\frac{\mu}{m_i}\left[\frac{B_R}{B^*_{||}}\frac{\partial B}{\partial R} + \frac{B_Z}{B^*_{||}}\frac{\partial B}{\partial Z}\right] - \frac{Z_i}{m_i}\left[\frac{B_R}{B^*_{||}}\frac{\partial\phi}{\partial R} + \frac{B_\zeta}{RB^*_{||}}\frac{\partial\phi}{\partial\zeta} + \frac{B_Z}{B^*_{||}}\frac{\partial\phi}{\partial Z}\right]\\ &-\frac{\mu v_{||}}{m_i\omega_{ci}}\frac{B}{B^*_{||}}\frac{1}{\mathscr{J}}\left[\frac{\partial}{\partial R}\left(\frac{RB_\zeta}{B}\right) - R\frac{\partial}{\partial Z}\left(\frac{B_\zeta}{B}\right)\right]\\ &-\frac{v_{||}Z_i}{m_i\omega_{ci}}\frac{B}{B^*_{||}}\frac{1}{\mathscr{J}}\left[\frac{\partial\phi}{\partial Z}\frac{\partial}{\partial R}\left(\frac{RB_\zeta}{B}\right) + \frac{\partial\phi}{\partial\zeta}\left(\frac{\partial}{\partial Z}\left(\frac{B_Z}{B}\right) - \frac{\partial}{\partial R}\left(\frac{B_Z}{B}\right)\right) - R\frac{\partial\phi}{\partial R}\frac{\partial}{\partial Z}\left(\frac{B_\zeta}{B}\right)\right].\end{aligned} \tag{34}$$

If f_{i0} is the equilibrium distribution and δf_i is the perturbation ($\delta f_i \ll f_{i0}$), such that $f_i = f_{i0} + \delta f_i$, and the particle weight is defined as $w_i = \delta f_i/f_i$, then

$$\frac{d}{dt}w_i = \frac{1}{f_{i0}}(1-w_i)\frac{d}{dt}\delta f_i \tag{35}$$

$$\frac{d}{dt}\delta f_i = -\underbrace{\vec{\mathbf{v}}_E\cdot\nabla f_{i0}|_{v_\perp}}_{\text{Drive term}} - \underbrace{\frac{Z_i}{T_i}\left(\hat{b}\cdot\nabla\phi\right)v_\parallel f_{i0}}_{\text{Parallel term}} - \underbrace{\frac{Z_i}{T_i}\left(\vec{\mathbf{v}}_d\cdot\nabla\phi\right)f_{i0}}_{\text{Drift term}} \tag{36}$$

$$\begin{aligned}
\text{Drive term} :=& -\vec{\mathbf{v}}_E\cdot\nabla f_{i0}|_{v_\perp} = -\vec{\mathbf{v}}_E\cdot\nabla f_{i0} - \frac{\mu f_{i0}}{T_i}\ \vec{\mathbf{v}}_E\cdot\nabla B \\
=& -f_{i0}\ \vec{\mathbf{v}}_E\cdot\left(\frac{\nabla n_i}{n_i} + A\frac{\nabla T_i}{T_i}\right) \\
=& -\frac{1}{RB^2}\left[\left(RB_\zeta\frac{\partial\phi}{\partial Z} - B_Z\frac{\partial\phi}{\partial\zeta}\right)\left(\frac{1}{n_i}\frac{\partial n_i}{\partial R} + \frac{A}{T_i}\frac{\partial T_i}{\partial R}\right) + \right. \\
& \left.\left(B_R\frac{\partial\phi}{\partial\zeta} - RB_\zeta\frac{\partial\phi}{\partial R}\right)\left(\frac{1}{n_i}\frac{\partial n_i}{\partial Z} + \frac{A}{T_i}\frac{\partial T_i}{\partial Z}\right)\right] f_{i0},
\end{aligned} \tag{37}$$

where

$$A = \left(\frac{2\mu\ B + m_i\ v_\parallel^2}{2T_i} - \frac{3}{2}\right).$$

$$\begin{aligned}
\text{Parallel term} :=& -\frac{Z_i}{T_i}\left(\hat{b}\cdot\nabla\phi\right)v_\parallel f_{i0} \\
=& -\frac{Z_i}{B\ T_i}\left(B_R\frac{\partial\phi}{\partial R} + B_Z\frac{\partial\phi}{\partial Z} + \frac{B_\zeta}{R}\frac{\partial\phi}{\partial\zeta}\right)v_\parallel f_{i0}
\end{aligned} \tag{38}$$

$$\begin{aligned}
\text{Drift term} :=& -\frac{Z_i}{T_i}\left[\frac{m_i\ v_\parallel^2 + \mu\ B}{m_i\ \omega_{ci}}\right]\left(\hat{b}\times\frac{\nabla B}{B}\right)\cdot\nabla\phi f_{i0} \\
=& -\frac{Z_i}{T_i}\left[\frac{m_i\ v_\parallel^2 + \mu\ B}{m_i\ \omega_{ci}\ B^2}\right]\left[\left(B_Z\frac{\partial B}{\partial R} - B_R\frac{\partial B}{\partial Z}\right)\frac{1}{R}\frac{\partial\phi}{\partial\zeta} + B_\zeta\frac{\partial B}{\partial Z}\frac{\partial\phi}{\partial R} - B_\zeta\frac{\partial B}{\partial R}\frac{\partial\phi}{\partial Z}\right] f_{i0}
\end{aligned} \tag{39}$$

In the present manuscript we are concerned with the linear simulation of ITG mode, and the particle trajectory is characterised only by the equilibrium magnetic field. We use 2nd order Runge Kutta (RK2) method to update the dynamical quantities in G2C3. Figure9(a) shows the different banana and passing particle orbits obtained using the above set of equations for a typical ADITYA-U tokamak discharge.

4.2. Particle loading/initialization

4.2.1. (R,Z,ζ) *initialization:* We distribute the particles uniformly in the spatial domain, by generating a uniform distribution of particles in a 3D Cartesian box and choosing particles which fall within the computational domain, as shown in Fig. 9(b). The particles are assigned to different cores of a Message Passing Interface (MPI) processes so that the computations in Sec. 4.1 can be performed in parallel (See Fig.9(c)). If we have N_{MPI} number of processors, then each processor is assigned $\Delta\zeta = 2\pi/N_{MPI}$ section of the computational domain with N_i/N_{MPI} number of particles each.

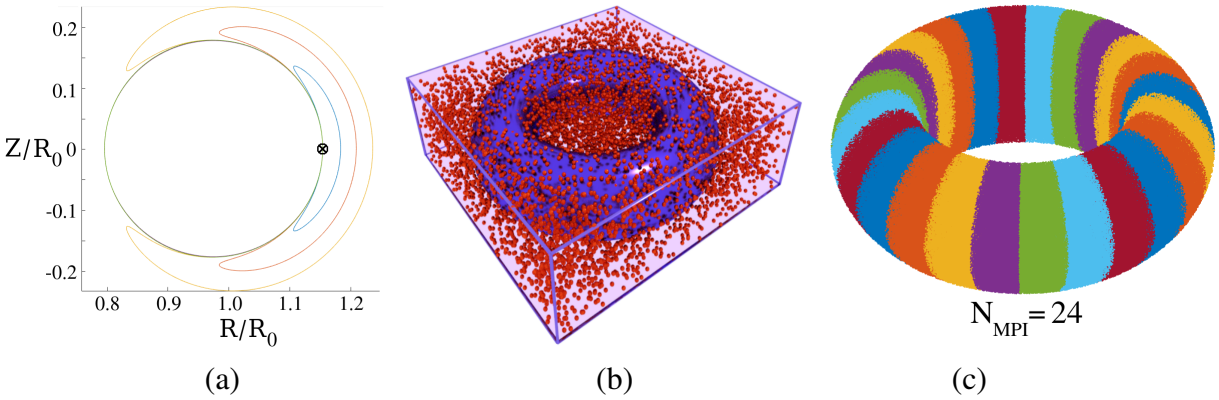


Figure 9. Particle trajectory with different $v_{||}$ starting from the same initial position with a transition from banana to passing orbits (ADITYA-U Shot # 36628) are shown in (a). The schematic in (b) shows the uniform distribution of particles in a box encompassing the toroidal computation domain. The particles belonging to the computational domain and the different MPI's are shown with different colours in (c).

4.2.2. $(v_{||}, \mu, w)$ *initialization:* The equilibrium phase space guiding centre distribution for ions is Maxwellian and is given by

$$f_{i0}(|\mathbf{v}|) = f_{i0}(v_{||}, \mu) = \frac{n_i(\psi)}{\left(\frac{2\pi T_i(\psi)}{m_i}\right)^{3/2}} exp\left[-\frac{2\mu B + m_i v_{||}^2}{2T_i(\psi)}\right] \tag{40}$$

where n_i and T_i are the equilibrium radial ion density and temperature profile, respectively. We use the particle loading scheme in which the ion-temperature and ion-density profiles are taken to be constant. In contrast, the gradients in the ion density and temperature profile are retained to drive the instabilities [42], as shown in Fig. 10. For this study, we consider the realistic DIII-D geometry shot #158103 at 3050 ms. This discharge is used for resonant magnetic perturbation (RMP) edge localized mode (ELM) [43, 44] suppression in the DIII-D pedestal. However, this work uses an experimental equilibrium such that RMPs are not applied and a toroidally symmetric equilibrium is used with the cyclone base case plasma profiles [45] to benchmark the ITG mode in the core region. Furthermore, the weights w are initialized to a uniform random distribution in the range of $(-\varepsilon, \varepsilon)$, where we choose $\varepsilon = 0.01$.

4.3. Transfer of particles between MPI's (SHIFTi)

During the evolution process if a particle crosses from one MPI domain to the other, the particle's information is transferred to the corresponding MPI processes. Figure 11(a) shows the trajectory of a set of particles as they move through different toroidal domains, where the colours indicate the label of the toroidal domain the particle is assigned to. The MPI partitioning is done using the ζ-values, with a time step such that the particles can only hop into neighbouring MPI's. The shift operation involves four send-receive operations between 3 neighbour MPI's: two to share the transfer-array dimension and two to transfer the array

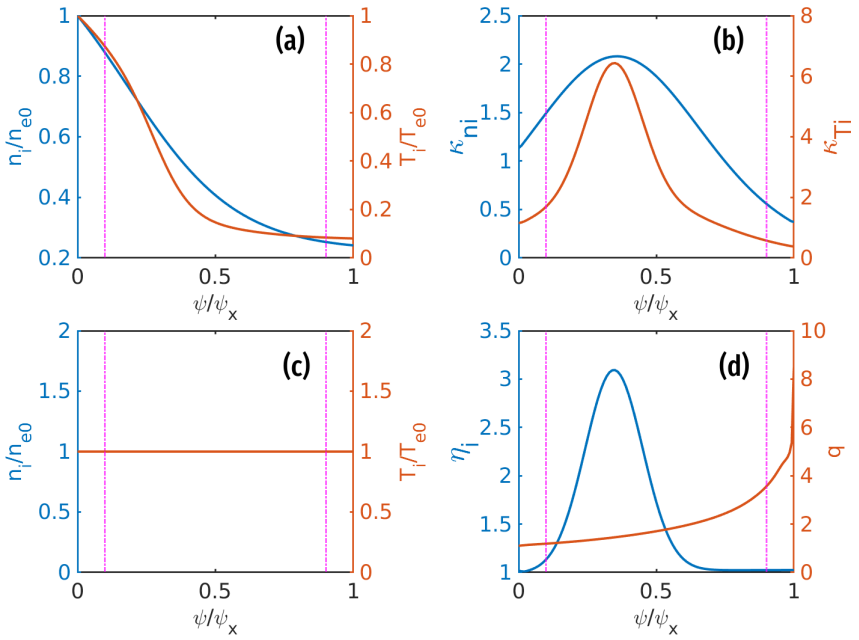


Figure 10. In (a) we plot the ion temperature and number density data as a function of normalized flux function. Subplot (c) shows the uniform profile of ion temperature and number density we use in the distribution function $f(\psi, v_{||}, \mu)$; But we retain the profile gradients in $\kappa_{Ti} := -\partial(\ln T_i)/\partial\psi$ and $\kappa_{ni} := -\partial(\ln n_i)/\partial\psi$, as shown in (b). The plots in (d) show the profile of safety factor and the $\eta = \kappa_{Ti}/\kappa_{ni}$. The dotted lines in the plot refer to the range of $\psi/\psi_X \in [0.1, 0.9]$ that belongs to the G2C3 computational domain.

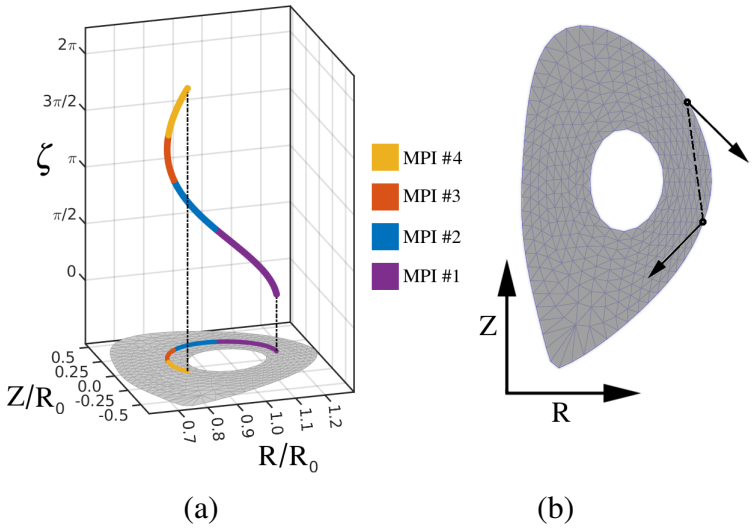


Figure 11. (a) Shows the passing particle trajectory shift across different MPIs ($N_{MPI} = 4$) during the time evolution; (b) When a particle is lost due to it exiting the simulation region it is reintroduced into the domain, such that energy is conserved. The vectors indicate the poloidal component of velocities.

elements. The algorithm is schematically demonstrated in Fig. 12 that entails the following steps:

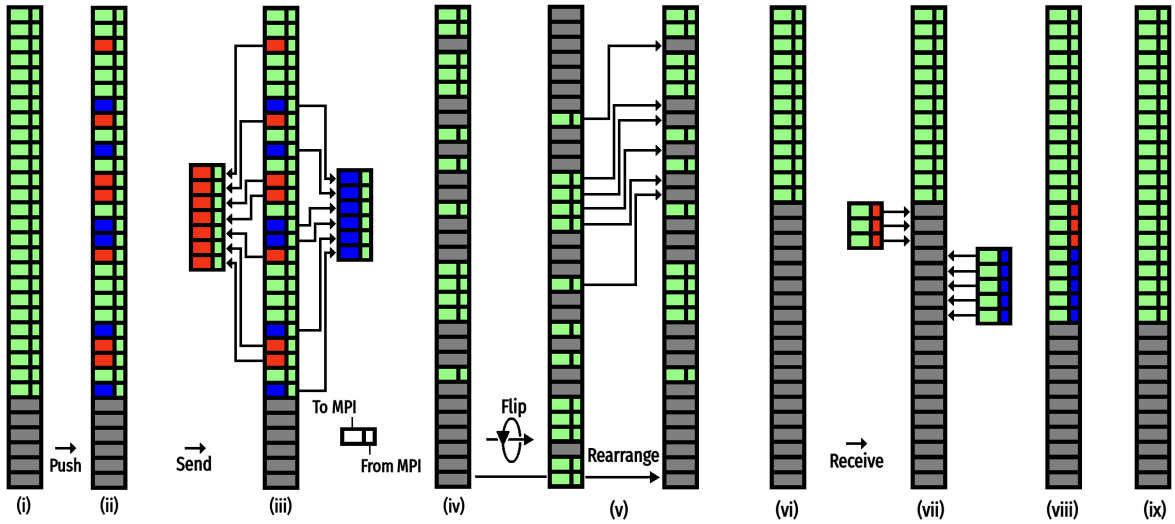


Figure 12. The schematic demonstrates how the shift operation is carried out between particle arrays that belong to different MPI's

(i) Particle array before Push;

(ii) After the particle push subroutine, the particles in the green MPI can be pushed to either of red or blue MPI's (ζ domains). The "From MPI" is determined by ζ before push and "To MPI" after push. The blocks in gray are the unused array elements;

(iii) Using the "To MPI" the data is extracted into different arrays;

(iv) The data with "To MPI" that don't match the current MPI are labelled as "unused";

(v) The gaps in the array are filled in from the reverse order;

(vi) The array has been rearranged;

(vii) Through MPI communication, the data from the neighbouring MPI's that have the current MPI as the "To MPI" labels are plugged into the unused array elements sequentially, first with red "From MPI", then blue "From MPI";

(viii) Particle array after the shift/receive operation;

(ix) Re-labelling of particle's "From MPI" label.

At the end of above loop we push the particles are repeat the sequence. This ensures that the particles belonging to different toroidal domains are handled by different MPI's, even after the particles cross-over into other toroidal domain.

4.4. Particle conservation and boundary conditions

As shown in Fig. 11(b), when a particle exits the computational domain, it is reintroduced again such that the $v_{||in} = v_{||out}$ and the re-entry point is determined such that the magnetic field strength is conserved, i.e., $B_{in} = B_{out}$. This ensures that the particle number and the total energy are conserved.

5. Field Calculations

To complete the PIC cycle for the electrostatic gyrokinetic simulation, we need to solve the gyrokinetic Poisson's equation. In this section, we describe the finite element method (FEM) to solve the canonical Poisson's equation in cylindrical coordinates. Later, we adopt this method to the gyrokinetic case. First, the particle weight, w_i, is scattered to the grid using the method described in Sec. 3.7 to obtain the density δn at the grid points. Then we solve for the electric potential as described below.

5.1. Poisson solver using FEM

In the cylindrical coordinate system, the Poisson's equation is given by

$$\nabla^2\phi = \left(\frac{1}{R}\frac{\partial}{\partial R}\left(R\frac{\partial\phi}{\partial R}\right) + \frac{\partial^2\phi}{\partial Z^2} + \frac{1}{R^2}\frac{\partial^2\phi}{\partial\zeta^2}\right) = -\delta n \tag{41}$$

where δn is the density perturbation from quasi-neutral state and ϕ is the corresponding electric potential.

Now, working with a large aspect ratio, such that $\left|\nabla^2\phi\right| \gg \left|\partial_R\phi/R\right|$ and safety factor, $q > 1$, we have $\partial_{||}\phi \approx \partial_\zeta\phi/R$, and by construction, $\left|\nabla\phi\right| / \left|\nabla_{||}\phi\right| \sim \left(k_\perp/k_{||}\right)$. Therefore,

$$\nabla^2\phi \approx \nabla^2_\perp\phi \approx \left(\frac{\partial^2\phi}{\partial R^2} + \frac{\partial^2\phi}{\partial Z^2}\right) := \underline{\nabla^2_{||}}\,\phi \tag{42}$$

which approximates the 3D Poisson's equation to an effective 2D equation on the poloidal planes. Note that in the effective Poisson's equation, (R,Z) act as Euclidean coordinates, and there is no coupling between the different poloidal planes. This enables us to solve for ϕ in each poloidal plane on different MPI's with minimal communication.

We use the FEM to solve for the weak form of Poisson's equation, given by

$$\int_{\mathscr{A}} \chi_\alpha(\mathbf{x})\ \underline{\nabla^2_{||}}\,\phi(\mathbf{x})\ d^2\mathbf{x} = -\int_{\mathscr{A}} \chi_\alpha(\mathbf{x})\ \delta n(\mathbf{x})\ d^2\mathbf{x}, \tag{43}$$

where $\mathscr{A}$ is the 2D interior domain of the poloidal grid; $\partial\mathscr{A}$ is the boundary of $\mathscr{A}$; $\chi_\alpha(\mathbf{x})$ is a test function, $\alpha = \{1,2,...N_G\}$, with N_G number of grid points, such that $\chi_\alpha(\mathbf{x})|_{\partial\mathscr{A}} = 0$, and $\sum_\alpha \chi_\alpha(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathscr{A}$ and 0 otherwise. Now integrating the LHS by parts we get,

$$\text{LHS} = \oint_{\partial\mathscr{A}} \chi_\alpha(\mathbf{x})\left(\hat{\mathbf{n}}\cdot\underline{\nabla_{||}}\,\phi(\mathbf{x})\right) ds - \int_{\mathscr{A}} \underline{\nabla_{||}}\,\chi_\alpha(\mathbf{x})\cdot\underline{\nabla_{||}}\,\phi(\mathbf{x})\ d^2\mathbf{x},$$

where $\hat{\mathbf{n}}$ is the outward normal at the boundary. Here we consider only the Dirichlet boundary condition, and by the definition of χ, the first term above is zero.

Now, we choose the test functions that are localized to the triangles connected to a vertex and zero outside, i.e. $\chi_\alpha(\mathbf{x}_i) = \delta_{\alpha i}$, where δ is the Kroneker delta, for the i^{th} grid point. Thus the number of test functions is equal to the number of grid points, and each of them gives rise to a relation in Eq.(43). The α^{th} integral equation is restricted to the triangles connected to α^{th} grid point as

$$\int_{\mathscr{A}}(.) \rightarrow \sum_{T_{\alpha jk}\in\{T\}}^{j,k} \int_{T_{\alpha jk}}(.) \tag{44}$$

where $T_{\alpha jk}$ is the triangle with vertices (α, j, k); $\{T\}$ is the set of all triangles in the mesh and the domain of integration is reduced to a triangle. Now, working in the area coordinates (see Sec. 3.5), without loss of generality we choose $(\alpha, j, k) \rightarrow (a, b, c)$; then $\chi_\alpha(\mathbf{x}) = a(\mathbf{x})$ and the RHS of Eq.(43) has,

$$\begin{aligned}
&-\int_{T_{\alpha jk}} \chi_\alpha(\mathbf{x})\ \delta n(\mathbf{x})\ d^2\mathbf{x} = -\int_{T_{abc}} a\ (a\ \delta n_a + b\ \delta n_b + c\ \delta n_c)\sqrt{g}\ da\ db \\
&= -\int_{a=0}^{1}\int_{b=0}^{1-a} a\ (a\ \delta n_a + b\ \delta n_b + (1-a-b)\ \delta n_c)\ \sqrt{g}\ da\ db \\
&= -A_{abc}\left(\frac{2\ \delta n_a + \delta n_b + \delta n_c}{12}\right)
\end{aligned} \tag{45}$$

and the LHS has,

$$\begin{aligned}
&-\int_{T_{\alpha jk}} \nabla_{\perp\!\!\!\perp}\chi_\alpha(\mathbf{x}) \cdot \nabla_{\perp\!\!\!\perp}\phi(\mathbf{x})\ d^2\mathbf{x} = -\int_{T_{abc}} g^{ef}\ \partial_e a\ \partial_f(a\ \phi_a + b\ \phi_b + c\ \phi_c)\ \sqrt{g}\ da\ db \\
&= -\int_{a=0}^{1}\int_{b=0}^{1-a} (g^{aa}(\phi_a - \phi_c) + g^{ab}(\phi_b - \phi_c))\ \sqrt{g}\ da\ db \\
&= -\left(\frac{R_{bc}^2 + Z_{bc}^2}{4\,A_{abc}}\right)\phi_a - \left(\frac{R_{ac}R_{cb} + Z_{ac}Z_{cb}}{4\,A_{abc}}\right)\phi_b - \left(\frac{R_{ab}R_{bc} + Z_{ab}Z_{bc}}{4\,A_{acb}}\right)\phi_c
\end{aligned} \tag{46}$$

where, $R_{ij} := (R_i - R_j)$, $Z_{ij} := (Z_i - Z_j)$; A_{ijk} is the (signed) area of triangle T_{ijk} (see Sec. 3.5). The metric tensor is given by,

$$g_{ij} = \begin{bmatrix} \mathbf{X}_{ab}\cdot\mathbf{X}_{ab} & \mathbf{X}_{ab}\cdot\mathbf{X}_{ac} \\ \mathbf{X}_{ab}\cdot\mathbf{X}_{ac} & \mathbf{X}_{ac}\cdot\mathbf{X}_{ac} \end{bmatrix} = \begin{bmatrix} R_{ab}^2 + Z_{ab}^2 & R_{ab}\,R_{ac} + Z_{ab}\,Z_{ac} \\ R_{ab}\,R_{ac} + Z_{ab}\,Z_{ac} & R_{ac}^2 + Z_{ac}^2 \end{bmatrix}$$

and $\sqrt{g} = \sqrt{det(g_{ij})} = 2\,A_{abc}$; Furthermore, g^{ij} is the matrix inverse of g_{ij}.
Then upon integration over the whole domain $\mathscr{A}$, the contribution at the i^{th} grid point can be written in a matrix form (with Einstien summation convention)

$$\begin{aligned}
&\mathbb{K}_{ij}\,\phi_j \ = \ -\mathbb{A}_{ik}\,\delta n_k \ = \ d_i \\
\Rightarrow\quad &\phi_j \ = \ -\mathbb{K}_{ij}^{-1}\mathbb{A}_{ik}\,\delta n_k \ = \ \mathbb{K}_{ij}^{-1}\,d_i
\end{aligned} \tag{47}$$

where,

$$\begin{aligned}
\mathbb{K}_{ii} &= -\sum_{T_{ijk}\in\{T\}}^{j,k}\left(\frac{R_{jk}^2 + Z_{jk}^2}{4\,A_{ijk}}\right), \qquad \mathbb{A}_{ii} = \sum_{T_{ijk}\in\{T\}}^{j,k}\frac{A_{ijk}}{6}, \\
\mathbb{K}_{ij} &= -\sum_{T_{ijk}\in\{T\}}^{k}\left(\frac{R_{ik}R_{kj} + Z_{ik}Z_{kj}}{4\,A_{ijk}}\right), \quad \mathbb{A}_{ij} = \sum_{T_{ijk}\in\{T\}}^{k}\frac{A_{ijk}}{12},
\end{aligned} \tag{48}$$

and for all the boundary grid points, i.e. $i \in \partial\mathscr{A}$

$$\begin{aligned}
&\mathbb{K}_{ii} = 1,\ \ \mathbb{K}_{ij} = 0,\ \text{ if }\ j \neq i \\
&\text{and }\ d_i = \phi_{0i}
\end{aligned}$$

which implies $\phi_i = \phi_{0i}$, where ϕ_{0i} is the boundary value, as per the Dirichlet boundary condition.

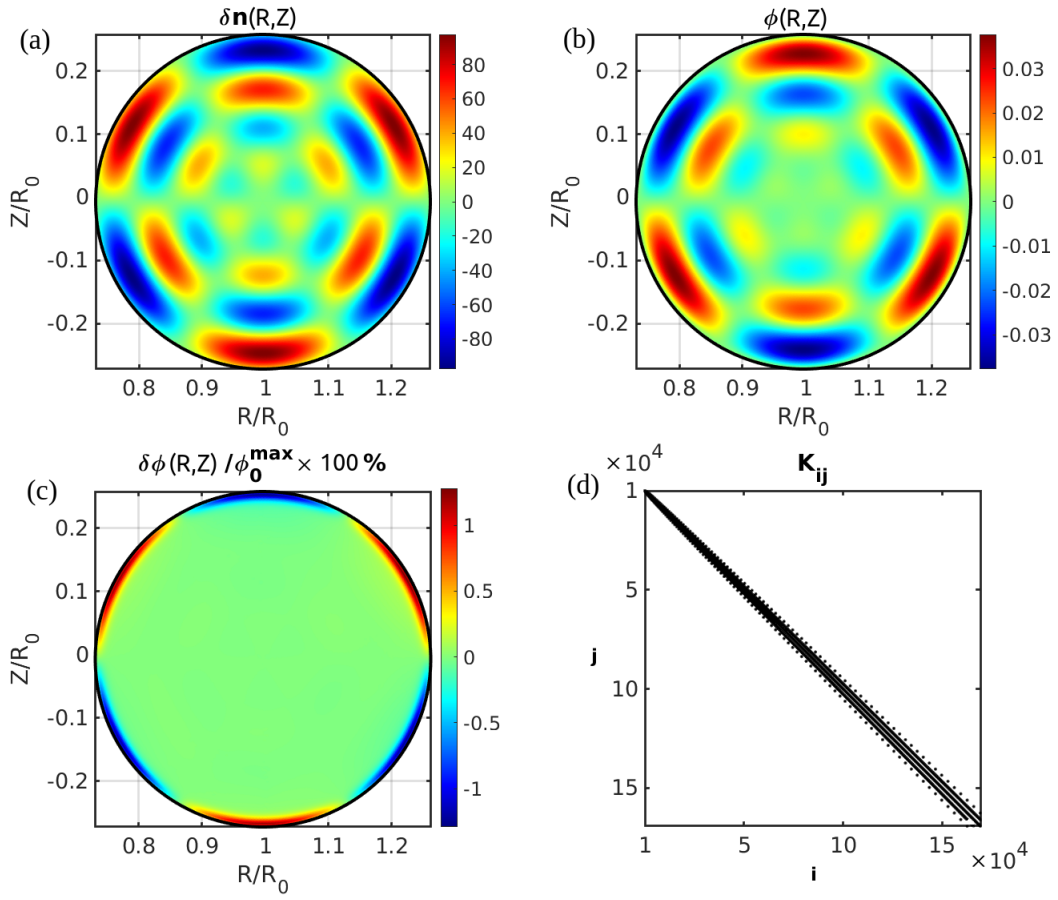


Figure 13. Plot (a) shows the charge density, δn, on a circular mesh centered around the magnetic axis with outer radius r_{max} and (b) is the estimated electric potential, ϕ, using the FEM. The difference between the estimated ϕ and the analytic solution is given in (c). We get up to 2% accuracy for 50 flux surfaces with 500 grid points in the angular direction. The binary plot in (d) is the visualization of non-zero elements of the sparse matrix $\mathbb{K}_{ij}$. Notice that the $\mathbb{K}_{ij}$ matrix gets wider for larger grid indices i as the grid points form a spiral starting from the inner circle and have an increasing number of grid points in consecutive flux surfaces.

Notice that a typical $\mathbb{K}_{ij}$ is sparse, as $\mathbb{K}_{ij} = 0$ if $T_{ijk} \notin \{T\}$ for all k (see Fig. 13(d)). This enables us to use the MPI-based PETSc package [46] to compute the resulting sparse-matrix inverse. To test the Poisson solver, we construct $\mathscr{A}$ with triangular mesh in the form of a circular disc including the magnetic axis (see Fig. 13). To ensure that the test function is well-defined at the magnetic axis and the 2nd derivative exists, we choose $\phi(r,\theta) = (1 - e^{(-r^2/\sigma^2)}) \sin(2\pi n_r \bar{r}/\Delta r) \sin(n_\theta \theta)$, and the corresponding density can be calculated as $\delta n(r,\theta) = -\nabla_\perp^2 \phi(r,\theta)$, where $r = \sqrt{(R/R_0 - 1)^2 + (Z/R_0)^2}$, $\theta = \arctan(Z/(R - R_0))$, $\Delta r = (r_{max} - r_{mag})$, $\bar{r} = (r - r_{mag})$; and n_r, n_θ are the mode numbers in the r, θ directions, respectively. Figure 13(a)-(c) shows the corresponding plots and the estimation error, for the case $n_r = 2$, $n_\theta = 3$. Also, when we include the magnetic axis in the simulation we only need to specify the boundary conditions on the outer most flux surface grids.

5.2. Gyrokinetic Poisson's equation

Now, we consider the gyrokinetic case [47, 48],

$$\frac{Z_i^2 e n_{i0}}{T_{i0}}(\phi - \tilde{\phi}) = Z_i \delta\bar{n}_i - \delta n_e, \tag{49}$$

where $\delta\bar{n}_i$ and δn_e are the guiding centre densities of ion and electron, respectively; n_{i0} and T_{i0} are the equilibrium density and temperature profiles of ions; $\tilde{\phi}$ is the second gyro-averaged perturbed potential, defined as

$$\tilde{\phi}(\vec{\mathbf{x}}) = \frac{1}{2\pi}\int d^3\vec{\mathbf{v}} \int d^3\vec{\mathbf{X}}\ f_{i0}(\vec{\mathbf{X}})\ \bar{\phi}(\vec{\mathbf{X}})\ \delta(\vec{\mathbf{X}} + \vec{\rho} - \vec{\mathbf{x}}) \tag{50}$$

where, $\vec{\mathbf{X}}$ and $\vec{\mathbf{x}}$ are the position coordinates of the guiding centre and the particle, respectively and $\vec{\rho}$ is the gyroradius vector. If α is the gyro angle, then the first gyro-averaged potential $\bar{\phi}(\vec{\mathbf{x}})$ is defined as

$$\bar{\phi}(\vec{\mathbf{X}}) = \int d^3\vec{\mathbf{x}} \int \frac{d\alpha}{2\pi}\ \phi(\vec{\mathbf{x}})\ \delta(\vec{\mathbf{x}} - \vec{\mathbf{X}} - \vec{\rho}) \tag{51}$$

The gyro-averaged physical quantities are calculated in real space using the four-point method [49]. Here, each particle is partitioned into four evenly spaced points on the gyro-ring in a plane perpendicular to the magnetic field. The gyro-averaging procedure is executed on poloidal planes instead of gyro-planes since we consider the large aspect ratio cross-section. The weight of the particle is uniformly distributed to the 4 gyro-particles and is projected onto the poloidal planes along the field lines to generate the scattered gyro-charge and hence the gyro-averaged density. Similarly, the potential estimated using the gyro-kinetic Poisson solver is gathered onto the 4 gyro-particles and the average potential is assigned to the particle. However, for spherical tokamaks such as MAST, ST-40, and NSTX an accurate treatment for the gyro-averaging needs to be carried out by taking into account the finite ratio of the poloidal to the total magnetic field B_{Pol}/B, which separates the poloidal plane from the gyro-plane, as discussed in [51].

For an adiabatic electron model, $\delta n_e = n_{e0} e\phi/T_{e0}$, the above equation can be rewritten as [50],

$$\underline{\nabla_{\perp}^2}\,\phi = -\frac{\omega_{ci}^2}{\omega_{pi}^2} 4\pi e \left(1 - \rho_i^2 \underline{\nabla_{\perp}^2}\right)(\delta\bar{n}_i - \delta n_e) \tag{52}$$

$\phi = e\,\phi/T_e$, length normalization by ρ_s, and $\rho_s = \tau\,\rho_i$. Then,

$$\underline{\nabla_{\perp}^2}\,\phi = -\left(1 - \frac{1}{\tau}\underline{\nabla_{\perp}^2}\right)(\delta\bar{n}_i - \delta n_e) \tag{53}$$

For adiabatic electron, $\delta n_e = \phi$. So,

$$\left(1 - \frac{1+\tau}{\tau}\underline{\nabla_{\perp}^2}\right)\phi = \left(1 - \frac{1}{\tau}\underline{\nabla_{\perp}^2}\right)\delta\bar{n}_i \tag{54}$$

The corresponding weak form can be written as

$$\tilde{\mathbb{K}}_{ij}\phi_j = \tilde{\mathbb{A}}_{ij}\ \delta\tilde{n}_j = \tilde{d}_i$$

where,

$$\tilde{\mathbb{K}}_{ij} = \left(\mathbb{A}_{ij} - \frac{1+\tau}{\tau}\mathbb{K}_{ij}\right); \quad \tilde{\mathbb{A}}_{ij} = \left(\mathbb{A}_{ij} - \frac{\mathbb{K}_{ij}}{\tau}\right) \tag{55}$$

and for all the boundary grid points, i.e. $i \in \partial\mathscr{A}$

$$\tilde{\mathbb{K}}_{ii} = 1, \;\; \tilde{\mathbb{K}}_{ij} = 0, \;\; \text{if} \;\; j \neq i \;\; \text{and} \;\; \tilde{d}_i = \phi_{0i}$$

where the Dirichlet boundary conditions are implemented in the FEM Poisson solver, by setting the ϕ_{0i} values at the boundary grids. Furthermore, the boundary conditions are enforced using a mask function of the form, $f_{mask}(\psi) := \tanh(\frac{\psi-\psi_{in}}{\sigma_\psi})\tanh(\frac{\psi-\psi_{out}}{\sigma_\psi})$, where σ_ψ controls the width of the buffer zone, which can be applied to both fields as well as gradients, to make it zero at the boundaries. In the present paper, We choose σ_ψ such that $f_{mask} \lesssim 0.9$ for 3 flux surfaces from the boundary.

5.3. Gradient calculation

To evaluate the particle dynamics, we need to estimate the gradients of various fields defined on the grids. We use a FEM based weak formulation of the gradient for $\nabla_{\perp\!\!\!\perp}$, where the $\perp\!\!\!\perp$ refers to the (R,Z) poloidal plane. We evaluate $\nabla_{||}$ using the finite difference along the field lines and can be estimated with higher accuracy than the toroidal component. This can be estimated as,

$$\nabla_{\parallel} = \hat{b}\cdot\nabla = \frac{1}{B}\left(\mathbf{B}\cdot\nabla_{\perp\!\!\!\perp} + \frac{B_\zeta}{R}\,\partial_\zeta\right) \;\Rightarrow\; \partial_\zeta = \frac{R}{B_\zeta}(B\,\nabla_{\parallel} - \mathbf{B}\cdot\nabla_{\perp\!\!\!\perp})$$

5.3.1. $\nabla_{\perp\!\!\!\perp}$ *- component* : The electrostatic potential ϕ obtained using the first-order FEM described in Sec. 5.2 is a piece-wise linear and continuous function in (R,Z), given by $\phi(\mathbf{x}) = a(\mathbf{x})\phi_a + b(\mathbf{x})\phi_b + c(\mathbf{x})\phi_c$. So the naive estimate of the electric field,

$$\mathbf{E}_{\perp\!\!\!\perp} = -\nabla_{\perp\!\!\!\perp}\phi \quad \Rightarrow \; E_\alpha = -\partial_\alpha\phi, \;\; \alpha \in \{R,Z\}$$

is constant within each triangle and is discontinuous across the edges and vertices (see Fig. 14). We overcome this problem using the weak formulation for the gradient, to ensure that the fields do not have a jump when particles shift across triangles (See Fig. 14). We have

$$(a\,E_{\alpha a} + b\,E_{\alpha b} + c\,E_{\alpha c}) \neq -(\phi_a\,\partial_\alpha a + \phi_b\,\partial_\alpha b + \phi_c\,\partial_\alpha c),$$

but imposing weak equivalence at the i^{th} grid point we get,

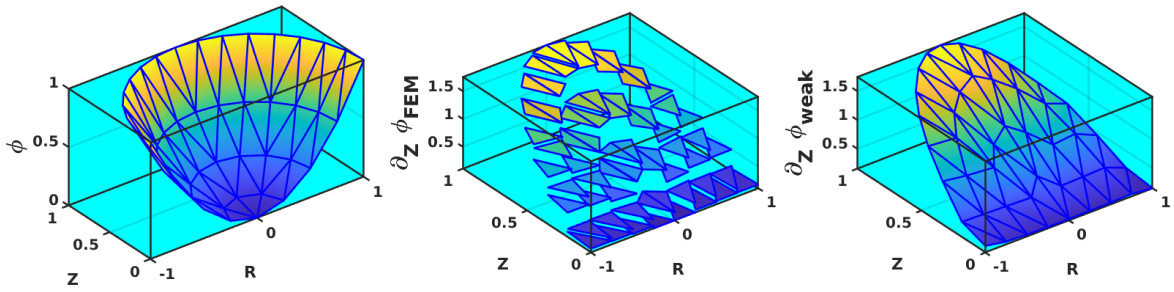
$$\int_{T_{abc}} \chi_i\,(aE_{\alpha a} + bE_{\alpha b} + cE_{\alpha c})\sqrt{g}\,da\,db = -\int_{T_{abc}} \chi_i\,(\phi_a\partial_\alpha a + \phi_b\partial_\alpha b + \phi_c\partial_\alpha c)\sqrt{g}\,da\,db$$

where χ_i is the test function, as defined in Sec. 5.1. Again, choosing $\chi_i = a$, we get the matrix form as: $\mathbb{A}_{ij}E_{\alpha j} = -\mathbb{B}_{\alpha ij}\phi_j$, where $\mathbb{A}$ is defined in Eq.(48) and

$$\mathbb{B}_{Rii} = \sum_{T_{ijk}\in\{T\}}^{j,k} \frac{Z_{jk}}{6}, \qquad \mathbb{B}_{Rij} = \sum_{T_{ijk}\in\{T\}}^{k} sign(A_{ijk})\,\frac{Z_{ki}}{6}$$

$$\mathbb{B}_{Zii} = \sum_{T_{ijk}\in\{T\}}^{j,k} \frac{R_{jk}}{6}, \qquad \mathbb{B}_{Zij} = \sum_{T_{ijk}\in\{T\}}^{k} sign(A_{ijk})\,\frac{R_{ki}}{6}$$

Observe that the $\mathbb{A}$ and $\mathbb{B}$ matrices are sparse and will have similar structures as in Fig. 13(d).



(a) (b) (c)

Figure 14. Schematic shows (a) the ϕ-field which we choose to have a parabolic form for demonstration. The gradient calculated by direct FEM (which is discontinuous) and the weak form are shown in (b) and (c), respectively.

5.3.2. $\nabla_{\parallel}$ *- component* : We calculate the parallel component of the electric field using the finite difference of potentials from the two neighbouring poloidal planes, along the field lines. But since this field is a function of two points on different poloidal planes, the resulting field is located at the mid-poloidal plane. Let (R_i, ζ_{mid}, Z_i) be the i^{th} grid point in the mid-poloidal plane, with the corresponding $\parallel$-projected points in the neighbouring poloidal plane given by $\mathbf{X}_L = (R_L, \zeta_L, Z_L)$ and $\mathbf{X}_R = (R_R, \zeta_R, Z_R)$. Then $\zeta_{mid} = (\zeta_L + \zeta_R)/2$, and if s_L, s_R are the corresponding arc-lengths, then

$$E_{\parallel}(R_i, Z_i)|_{\zeta_{mid}} \;=\; sign\left(B_{\zeta}(\zeta_R - \zeta_L)\right)\left(\frac{\phi(\mathbf{X}_R) - \phi(\mathbf{X}_L)}{s_R + s_L}\right) \tag{56}$$

6. Simulation of the linear ITG mode

We have so far described all the modules needed to perform a gyrokinetic electrostatic PIC simulation. In this section, we proceed to benchmark our implementation by studying the linear Ion Temperature Gradient (ITG) driven instability modes. For this study, we use the equilibrium magnetic field obtained from the DIII-D shot #158103, from a g-file generated using EFIT [32], with cyclone profile.

We follow the below-mentioned setup sequence and parameters to initialize the simulation:

- Grid generation (Sec. 2): We perform the simulation in an annular region ($\psi_1/\psi_\times$=0.1 and $\psi_{m_\psi}/\psi_\times = 0.9$) to minimize computations, given that turbulence activity near the magnetic axis is minimal. We use 50 flux surfaces (m_ψ) with 32 poloidal planes (m_ζ), such that the radial and angular resolution in the poloidal planes are $\Delta r/R_0 \sim 3.8 \times 10^{-3}$ and $\Delta s/R_0 \sim 3 \times 10^{-3}$, respectively. Note that we have chosen the grid sizes such that $\Delta r/\rho_i \approx 3$, to resolve the poloidal components.
- Train the neural network for $\parallel$-projection and triangle-locator (Sec. 3): We use the 2 layered neural network to find the maps $\mathcal{N}_{\parallel}$ and $\mathcal{N}_{\Delta}$ which are trained with a dataset of

size 7.4×10^5 and 7.4×10^4, respectively. We use a stochastic gradient descent-based optimizer, with a quadratic loss function. Upon training, for the $\mathscr{N}_{\parallel}$ neural network we observe that the loss converges to $\sim 1\%$ and for the $\mathscr{B}$ or the $\mathscr{N}_{\Delta}$ network the predicted triangle falls within the next nearest neighbour on average (which is corrected using the iterative scheme in Sec. 3.4). The loss evolutions of the networks are given in Fig. 7.

- Load profiles (Sec. 4.2.1): We load the profile data shown in Fig. 10 onto the simulation grid. The ion and electron temperatures at the magnetic axis are 1.690 keV. We consider hydrogen ions ($Z_i = 1$), and hence by the quasi-neutrality requirement we have $n_i = n_e$, equal to $3.28 \times 10^{13} cm^{-3}$ at the magnetic axis. The purpose of our study is to benchmark the capabilities of G2C3 to handle realistic geometry with minimum technical complexity and compare with GTC results. The fully self-consistent treatment of the density/temperature profile and the equilibrium will be addressed in subsequent work.
- Load particles (Sec. 4.2.1): We load 32×10^6 number of particles, uniformly into the simulation domain, with Maxwellian velocity distribution. We initialize the particle weights with uniform distribution, such that $w \in [-0.005, 0.005]$.
- Initialize the FEM matrices (Sec. 5): We initialize the $\mathbb{K}_{ij}$, $\mathbb{A}_{ij}$, $\mathbb{B}_{Rij}$, and $\mathbb{B}_{Zij}$ as sparse matrices to estimate the electric field. We apply the Dirichlet boundary condition and enforce the field quantities to vanish at the boundaries. The sparse matrices are partitioned across multiple MPIs and handled by the PETSc package [46]. We use a combination of a Krylov subspace method (KSP solver) [53] with a "Block Jacobi" preconditioner [46] to perform iterative numerical inversion to find the solution of the linear system, with a tolerance level of $\sim 10^{-7}$. This preconditioner partitions the matrix into blocks, each assigned to a different process and is well suited for parallel computing. Each block is then preconditioned independently using Incomplete LU factorization.

We perform a linear adiabatic-electron electrostatic gyrokinetic simulation, with a time step size of $0.04\, R_0/C_s$, where R_0 is the major radius and C_s is the ion sound speed. Now, the system is run through the main PIC loop, namely: scatter particle weights to the grid (Sec. 3); estimate the potential on the grid (Sec. 5); estimate the electric field (Sec. 5.3); gather the electric field to the particle (Sec. 3); push the particle (Sec. 4); transfer particles to their respective MPI (Sec. 4.3). The above sequence of operations is repeated iteratively to evolve the mode structure growth.

We simulate for $60\, R_0/C_s$ time to find an ITG mode becomes unstable due to the ion temperature gradient in the core. To verify the linear ITG mode implementation in G2C3, we adopt the realistic DIII-D tokamak geometry with analytical plasma profiles. Simulations are performed using G2C3 and the results are benchmarked against those from the well-established gyrokinetic code GTC, using identical geometry and plasma parameters. This cross-comparison serves as a rigorous verification of G2C3's linear physics capabilities.

The mode structure of the electrostatic potential on the poloidal plane is shown in Fig. 15(a), where the dominant ITG mode is found to have mode numbers (m,n) = (79,56), with a growth rate of $\gamma = 0.43\, C_s/R_0$, localized at $\psi/\psi_{\times} = 0.31$. The linear eigenmode exhibits a typical ballooning structure, concentrated on the outer mid-plane where the

magnetic curvature is unfavorable, consistent with the drive mechanism of the ITG instability. The G2C3 results are found to be in good agreement with those from GTC for the same plasma profiles, geometry, and input parameters. It should be noted that this comparison is approximate, as the two codes differ in their physical models, numerical schemes, and coordinate systems. The extraction of mode numbers requires a transformation to Boozer coordinates, which is detailed in the following section.

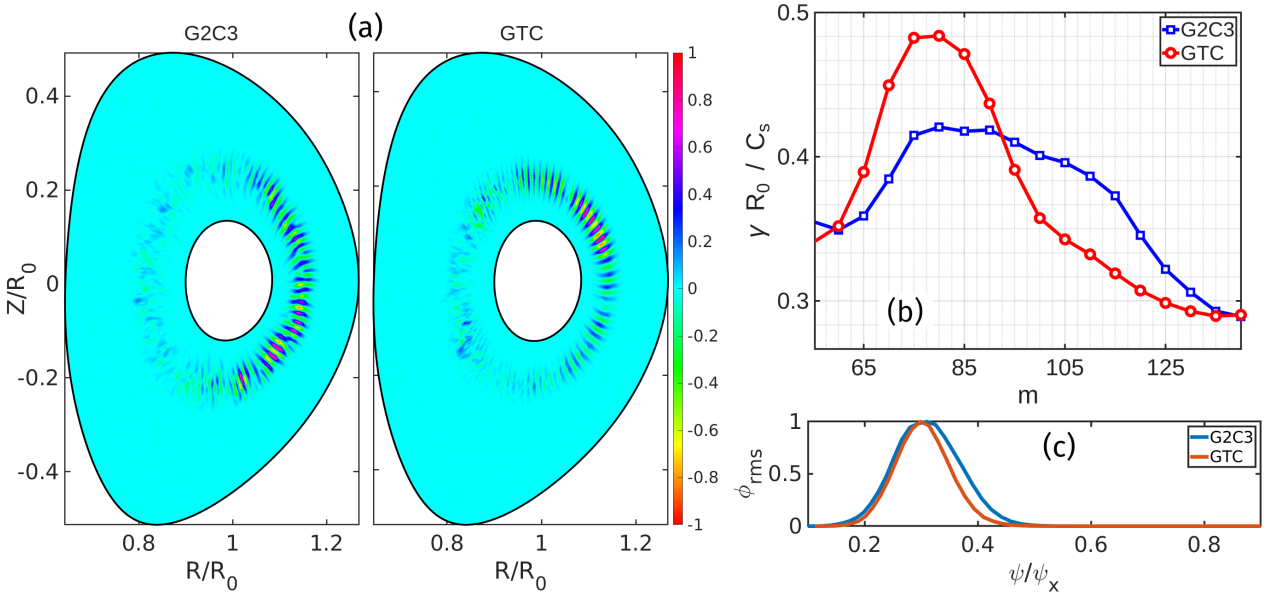


Figure 15. In (a) we show the 2D plot of the electric potential $\phi(R,\zeta,Z)$ for the ITG mode obtained using G2C3 and GTC. We utilize an annular domain to minimize computations, given that turbulence activity near the magnetic axis is minimal; The dispersion relation are plotted in (b). We calculate the rms value of the ϕ-field over each flux-surface (outer mid-plane, $Z=0$), to obtain the profile shown in (c), which shows the localization of the ITG mode, normalized to the maximum amplitude.

6.1. Mode structure analysis

To analyze the mode structure harmonics in the core we transform to the Boozer coordinates [54], such that each point on the poloidal plane can be represented in terms of the flux function and an angle variable as,

$$\mathscr{T}_{Boozer} : (R,\zeta,Z) \rightarrow (\psi,\theta,\zeta) \tag{57}$$

where the field lines on each flux surface are represented by $\bar{\theta} = q(\psi)\ \zeta + \theta$, with $q(\psi)$ as the safety factor. To find θ, we move along the field line in the $\hat{b}$ direction until we cross the outer mid-plane at $(R^+,\zeta^+,0)$, and move in the $-\hat{b}$ direction to find $(R^-,\zeta^-,0)$. Then

$$q = \frac{2\pi}{(\zeta^+ - \zeta^-)} \quad \text{and} \quad \theta = \frac{2\pi\ |\zeta^-|}{(\zeta^+ - \zeta^-)}. \tag{58}$$

An (m,n)-mode is of the form $\sim A_{m,n}\ sin(m\ \theta - n\ \zeta)$. We use 2D Fast-Fourier-Transform (FFT) [55] to perform the $(\theta,\zeta) \mapsto (k_\theta, k_\zeta)$ transformation on each flux surface. To calculate the FFT we construct a square grid in the (θ,ζ) coordinate for each flux surface, with $N_{max} \times N_{max}$ grids, where N_{max} is the number of grid points in a poloidal plane on the outermost flux surface. The $\mathscr{N}_{\|}$-map is used to perform field-line-aligned interpolation to transfer the field from the computational grid to the Boozer grid.

Figure 16(a),(b) shows the ϕ-field and the corresponding Fourier representation for the obtained ITG mode on the flux surface where the amplitude is maximum. Now, to extract only the (m,n) mode we build the filter response function in the Fourier space as

$$\mathscr{F}(k_\theta, k_\zeta) := \delta_{k_\theta,m}\ \delta_{k_\zeta,n} \tag{59}$$

where the δ's are the Kroneker delta functions. Then the filtered output is given by

$$\bar{\Phi}(k_\theta, k_\zeta)|_\psi = \Phi(k_\theta, k_\zeta)|_\psi\ \mathscr{F}(k_\theta, k_\zeta) \tag{60}$$

which is in the Fourier space, and we perform an inverse FFT to finally obtain the filtered potential in (θ,ζ) space. In linear simulations, we need to ensure that the noise does not excite other modes in the system which is achieved using the filter operation. Using the mode structure analysis described above we extract profiles of the dominant (m,n) mode as shown in Fig. 16(a),(b) and the corresponding exponential amplitude growth in Fig. 17(a). Furthermore, we extract the growth rates of the sub-dominant n-modes to generate the dispersion plot in Fig. 15(b), which shows a peaks at $m = 79$ and gradually falls off away from it.

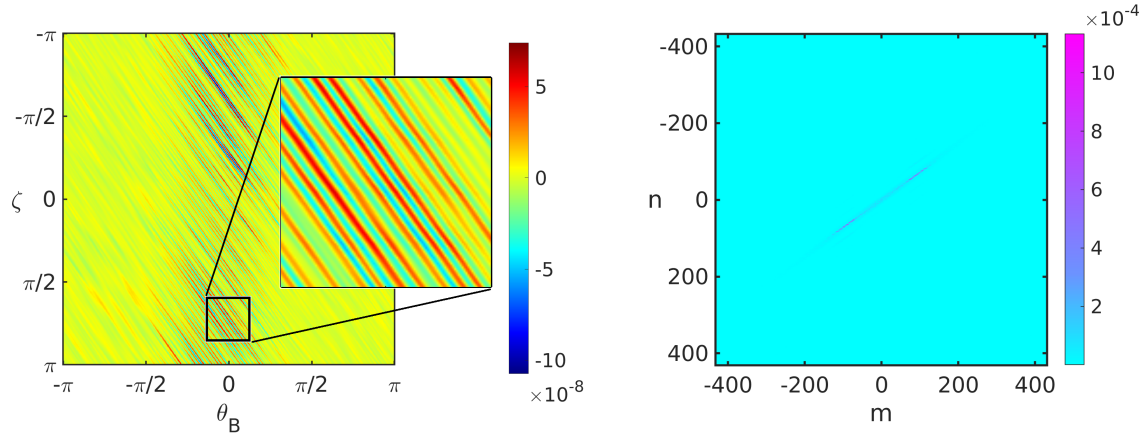


Figure 16. (a) The ϕ-field as a function of (θ,ζ) on the flux-surface with the highest amplitude (ψ_{max}). The inset shows the linear structure of the ITG mode; (b) FFT of the ϕ-field in (a). The peaks in amplitude correspond to the (m,n) (and $(-m,-n)$) value of the mode, and we note that $m/n(1.4043) \approx q(\psi_{max})$ as indicated by the black line.

6.2. Convergence analysis:

The convergence of the simulation results in terms of growth rate concerning the total number of marker particles, and the time step sizes are given in Fig. 17(b). We find that the growth

rate saturates with the ITG amplitude maximum localizing at the radial location $r \sim 1.15\, R_0$ and a radial full-width half maximum of $r \sim 0.01\, R_0$ (see Fig. 15(c)). Convergence w.r.t. the marker particle numbers show that the growth rate saturates for total particles $\gtrsim 25.6 \times 10^6$, which is approximately 6 particles per triangle. Also, the growth rate stabilizes for the time step $\lesssim 0.04(R_0/C_s)$.

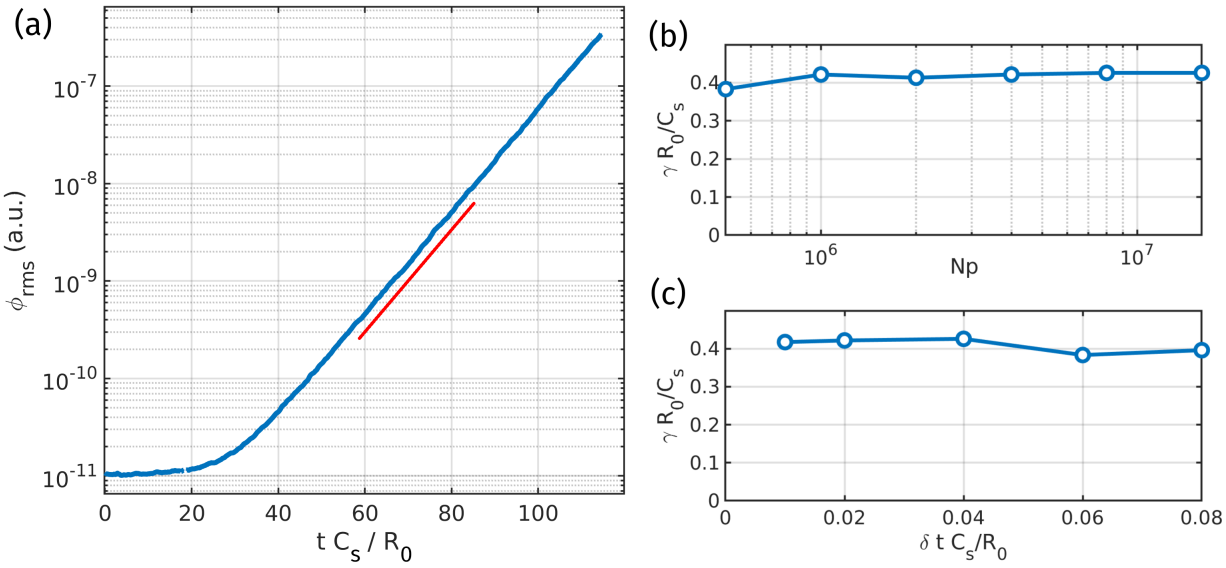


Figure 17. (a) Growth of ϕ_{rms} for the ITG mode; (b) Convergence of growth rates w.r.t. number of particles per MPI and w.r.t. time step in (c).

7. Conclusion

We describe the detailed implementation of various modules of G2C3, a gyrokinetic PIC code to study microinstabilities of fusion plasma confinement in a tokamak system. The code employs a neural network to perform gather-scatter operations in cylindrical coordinates and avoids the problems one encounters by the use of field line coordinates at the last closed flux surface. We also use a neural network to locate the triangle from the mesh which encompasses the particle. In G2C3, we implement a FEM-based gyrokinetic Poisson solver. We benchmark G2C3 by reproducing the linear ITG mode in the core region, with adiabatic electrons, for a Cyclone test case in the realistic DIII-D geometry. This result is the first step for G2C3 towards achieving a whole plasma volume simulation in the future.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility, under Award DE-FC02-04ER54698. This work is supported by National Supercomputing Mission (NSM) (Ref No:

DST/NSM/R&D_HPC_Applications/2021/4), Board of Research in Nuclear Sciences (BRNS Sanctioned nos. 39/14/05/2018-BRNS and 57/14/04/2022-BRNS), Science and Engineering Research Board (SERB sanctioned nos. EEQ/2017/000164 and EEQ/2022/000144), Anusandhan National Research Foundation (Sanctioned nos. ANRF/ARG/2025/002861/PS) and Infosys Young Investigator award. The results presented in this work have been simulated on ANTYA cluster at Institute of Plasma Research, Gujarat, SahasraT, and Param Pravega supercomputer at Indian Institute of Science, Bangalore, India.

Disclaimer

[1] A. Kuley and V. K. Tripathi, “Stabilization of ion temperature gradient driven modes by lower hybrid wave in a tokamak", Phys. Plasmas, **16**, 032504 (2009).

[2] A. Kuley and V. K. Tripathi, “Parametric upconversion of lower hybrid wave by runaway electrons in tokamak", Phys. Plasmas **17**, 062507 (2010).

[3] A. Kuley, C. S. Liu and V. K. Tripathi, “Lower hybrid destabilization of trapped electron modes in tokamak and its consequences for anomalous diffusion", Phys. Plasmas **17**, 072506 (2010).

[4] A. Kuley, Z. Lin, J. Bao, X. S. Wei, Y. Xiao, W. Zhang, G. Y. Sun, and N. J. Fisch, “Verification of nonlinear particle simulation of radio frequency waves in tokamak", Phys. Plasmas **22**, 102515 (2015).

[5] W. Horton, “Drift waves and transport", Reviews of Modern Physics **71**, no. 3: 735 (1999).

[6] V. Rozhansky, E. Kaveeva, P. Molchanov, I. Veselova, S. Voskoboynikov, D. Coster, G. Counsell, A. Kirk, S. Lisgo, the ASDEX-Upgrade Team and the MAST Team, “New B2SOLPS5.2 transport code for H-mode regimes in tokamaks", Nucl. Fusion **49**, 025007 (2009).

[7] B.D. Dudson, M. V. Umansky, X. Q. Xu, P. B. Snyder, and H. R. Wilson. “BOUT++: A framework for parallel plasma fluid simulations." Computer Physics Communications **180**, 1467 (2009).

[8] A.V. Chankin, D.P. Coster, G. Corrigan, S.K. Erents, W. Fundamenski, A. Kallen- bach, K. Lackner, J. Neuhauser, and R. Pitts, “Fluid code simulations of radial electric field in the scrape-off layer of JET", Plasma Phys. Control. Fusion **51** 065022 (2009).

[9] J.M. Canik, R. Briesemeister, C.J. Lasnier, W. Leonard, J.D. Lore, G. McLean, and J.G. Watkins, “Modeling of detachment experiments at DIII-D", J. Nucl. Mater. **463**, 569 (2015).

[10] Andreas Stegmeir, Omar Maj, David Coster, Karl Lackner,Markus Held, Matthias Wiesenberger, “Advances in the flux-coordinate independent approach", Computer Physics Communications **213**, 111 (2017).

[11] E. L. Shi, G. W. Hammett, T. Stoltzfus-Dueck, and A. Hakim, “Full-f gyrokinetic simulation of turbulence in a helical open-field-line plasma", Physics of Plasmas **26**, 012307 (2019).

[12] N. R. Mandell, A. Hakim, G. W. Hammett, and M. Francisquez, "Electromagnetic full-f gyrokinetics in the tokamak edge with discontinuous Galerkin methods", Journal of Plasma Physics **86**, 905860109 (2020).
[13] T. Singh, J.H. Nicolau, F. Nespoli, G. Motojima, Z. Lin, A. Sen, S. Sharma, and A. Kuley, "Global gyrokinetic simulations of electrostatic microturbulent transport in LHD stellarator with boron impurity", Nuclear Fusion **64**, 016007 (2024).
[14] T. Singh, K. Shah, D. Sharma, J. Ghosh, K.A. Jadeja, R.L. Tanna, M. B. Chowdhuri, Z. Lin, A. Sen, S. Sharma, and A. Kuley, "Gyrokinetic simulations of electrostatic microturbulence in ADITYA-U tokamak with argon impurity", Nuclear Fusion **64**, 086038 (2024).
[15] Tajinder Singh,Tariq Rafiq, Eugenio Schuster, Zhihong Lin, Animesh Kuley, "Global gyrokinetic simulations of kinetic ballooning mode in NSTX-U plasmas", Nuclear Fusion **65**, 106039 (2025).
[16] Abhishek Tiwari, Kshitish Barada, Jaya Kumar Alageshan, Santanu Banerjee, Tanmay Macwan, Terry L. Rhodes, Sarveshwar Sharma, Zhihong Lin, Animesh Kuley, "Ti/Te Dependence of Core Turbulence and Transport in DIII-D QH-Mode Plasmas", arXiv:2512.11328.
[17] J. Candy, and R. E. Waltz. "An Eulerian gyrokinetic-Maxwell solver" Journal of Computational Physics **186**, 545 (2003).
[18] J. Candy, A.B. Emily, and R. V. Bravenec. "A high-accuracy Eulerian gyrokinetic solver for collisional plasmas", Journal of Computational Physics **324**, 73 (2016).
[19] S. Jolliet, A. Bottino, P. Angelino, R.Hatzky, T. M. Tran, B. F. Mcmillan, O. Sauter, K. Appert, Y. Idomura, and L. Villard, "A global collisionless PIC code in magnetic coordinates", Comput. Phys. Commun. **177**, 409 (2007).
[20] Q. Pan, D. Told, E.L. Shi, G.W. Hammett, and F. Jenko, "Full-f version of GENE for turbulence in open-field-line systems", Phys. Plasmas **25** (2018) 062303.
[21] V. Grandgirard, J. Abiteboul, J. Bigot, T. Cartier-Michaud, N. Crouseilles, G. Dif-Pradalier, C. Ehrlacher, D. Esteve, X. Garbet, P. Ghendrih, G. Latu, M. Mehrenberger, C. Norscini, C. Passeron, F. Rozar, Y. Sarazin, E. Sonnendrucker, A. Strugarek, and D. Zarzoso, "A 5D gyrokinetic full- global semi-Lagrangian code for flux-driven ion turbulence simulations", Comput. Phys. Commun. **207**, 35 (2017).
[22] C. S. Chang, S. Ku, P. H. Diamond, Z. Lin, S. Parker, T. S. Hahm, and N. Samatova, "Compressed ion temperature gradient turbulence in diverted tokamak edge", Phys. Plasmas **16**, 056108 (2009).
[23] S. De, T. Singh, A. Kuley, J. Bao, Z. Lin, G. Y. Sun, S. Sharma, and A. Sen, "Kinetic particle simulations in a global toroidal geometry", Phys. Plasmas **26**, 082507 (2019).
[24] Z. X. Lu, Ph. Lauber, T. Hayward-Schneider, A. Bottino, and M. Hoelzl, "Development and testing of an unstructured mesh method for whole plasma gyrokinetic simulations in realistic tokamak geometry", Phys. Plasmas **26**, 122503 (2019).
[25] D. Michels, A. Stegmeir, P. Ulbl, D. Jarema, and F. Jenko, "Development and testing of an unstructured mesh method for whole plasma gyrokinetic simulations in realistic tokamak geometry", Comp. Phys. Comm. **264**, 107986 (2021).
[26] Kates-Harbeck, Julian, A. Svyatkovskiy, and W. Tang. "Predicting disruptive instabilities in controlled fusion plasmas through deep learning", Nature **568**, 526 (2019).
[27] Degrave, Jonas, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds et al. "Magnetic control of tokamak plasmas through deep reinforcement learning", Nature **602**, 414 (2022).
[28] Wei, Xishuo, S. Sun, W. Tang, Z. Lin, H. Du, and Ge Dong. "Reconstruction of tokamak plasma safety factor profile using deep learning" Nuclear Fusion **63**, 086020 (2023).
[29] Van de Plassche, K. Lucas, J. Citrin, C. Bourdelle, Y. Camenen, F.J. Casson, V.I. Dagnelie, F. Felici, A. Ho, S. Van Mulders, and J. E. T. Contributors, "Fast modeling of turbulent transport in fusion plasmas using neural networks" Phys. Plasmas **27**, 022310 (2020).
[30] Badiali, Chiara, P.J. Bilbao, F. Cruz, and L.O. Silva, "Machine-learning-based models in particle-in-cell codes for advanced physics extensions", Journal of Plasma Physics **88**, 895880602 (2022).
[31] Miller, M. Andres, R.M. Churchill, A. Dener, C.S. Chang, T. Munson, and R. Hager, "Encoder–decoder neural network for solving the nonlinear Fokker–Planck–Landau collision operator in XGC", Journal of Plasma Physics **87**, 905870211 (2021).
[32] L. L. Lao, J. R. Ferron, R. J. Groebner, W. Howl, H. St John, E. J. Strait, and T. S. Taylor, "Equilibrium

analysis of current profiles in tokamaks", Nucl. Fusion **30**, 1035 (1990).
[33] D. Sharma, R. Srinivasan, J. Ghosh, P. Chattopadhyay, and Aditya Team, "Aditya Upgradation – Equilibrium study", Fusion Engineering and Design, **160**, 111933 (2020).
[34] S. Haykin, "Neural networks: a comprehensive foundation", Prentice Hall PTR, (1998).
[35] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks **2**, 359 (1989).
[36] A. Kolmogorov, "On the representation of continuous functions of several variables by superposition of continuous functions of a smaller number of variables", Proceedings of the USSR Academy of Sciences, 108, pp. 179–182, (1956); English translation: Amer. Math. Soc. Transl., 17, pp. 369–373, (1961).
[37] V.I. Arnold, "On functions of three variables", Proceedings of the USSR Academy of Sciences, 114, pp. 679–681, (1957); English translation: Amer. Math. Soc. Transl., 28, pp. 51–54, (1963).
[38] A. Allievi and R. Bermejo, "A Generalized Particle Search–Locate Algorithm for Arbitrary Grids", J. Comput. Phys., **132**, 157, (1997).
[39] P. J. Catto, "Linearized gyro-kinetics", Plasma Physics **20**, 719 (1978).
[40] R. G. Littlejohn, "Variational principles of guiding centre motion", Journal Plasma Phys. **29**, 111 (1983).
[41] T. Singh, J.H. Nicolau, Z. Lin, S. Sharma, A. Sen, and A. Kuley, Global gyrokinetic simulations of electrostatic microturbulent transport using kinetic electrons in LHD stellarator, Nuclear Fusion **62**, 126006 (2022).
[42] Y. Xiao, I. Holod, Z. X. Wang, Z. Lin, and T. G. Zhang, "Gyrokinetic particle simulation of microturbulence for general magnetic geometry and experimental profiles", Phys. Plasmas **22**, 022516 (2015).
[43] S. Banerjee, S. Mordijck, K. Barada, L. Zeng, R. Groebner, T. Osborne, T.L. Rhodes, P.B. Snyder, B. Grierson and A. Diallo, "Evolution of ELMs, pedestal profiles and fluctuations in the inter-ELM period in NBI and ECH dominated discharges in DIII-D", Nuclear Fusion **61**, 056008 (2021).
[44] Santanu Banerjee, K. Barada, C. Chrystal, R. Groebner, S. Mordijck, T. Odstrcil, T. Osborne, T. Rhodes, F. Scotti, Z. Yan, L. Zeng, J. Damba, F. Laggner, S. Haskey, B. Grierson, J. Chen, S. Saarelma, and A. Pankin, "Decoupling of peeling and ballooning thresholds of pedestal stability and reduction in ELM frequency via enhanced turbulence with edge electron cyclotron heating in DIII-D", Nuclear Fusion **64**, 086010 (2024)
[45] C.M. Greenfield, J.C. DeBoo, T.H. Osborne, F.W. Perkins, M.N. Rosenbluth and D. Boucher, "Enhanced fusion performance due to plasma shape modification of simulated ITER discharges in DIII-D" Nuclear Fusion **37**, 1215 (1997).
[46] "PETSc: Portable, Extensible Toolkit for Scientific computation", Mathematics and Computer Science Division, Argonne National Laboratory, http://www-unix.mcs.anl.gov/petsc/petsc-2.
[47] T. Singh, D. Sharma, T. Macwan, S. Sharma, J. Ghosh, A. Sen, Z. Lin, and A. Kuley, "Gyrokinetic simulation of electrostatic microturbulence in ADITYA-U tokamak", Nuclear Fusion, **63**, 056008 (2023).
[48] Abhishek Tiwari, Joydeep Das, Jaya Kumar Alageshan, Gareth Roberg-Clark, Gabriel Plunk, Pavlos Xanthopoulos, Sarveshwar Sharma, Zhihong Lin, Animesh Kuley, "Zonal flow suppression of turbulent transport in the optimized stellarators W7-X and QSTK", Plasma Physics and Controlled Fusion **67**, 085025 (2025).
[49] W.W. Lee, "Gyrokinetic Particle simulation model", J. Comput. Phys., **72**, 243 (1987).
[50] Y. Nishimura, and Z. Lin. "A finite element mesh in a tokamak edge geometry." Contributions to Plasma Physics 46.7-9 551-556 (2006).
[51] W. X. Wang, Z. Lin, W. M. Tang, W. W. Lee, S. Ethier, J. L. V. Lewandowski, G. Rewoldt, T. S. Hahm, and J. Manickam, "Gyro-kinetic simulation of global turbulent transport properties in tokamak Experiments", Phys. Plasmas **13**, 092505 (2006).
[52] A. J. Brizard, and T. S. Hahm, "Foundations of nonlinear gyrokinetic theory" Rev. Mod. Phys. **79**, 421 (2007).
[53] J. Nocedal, and S.J. Wright, "Numerical optimization", Springer series in operation research and financial engineering (2nd ed.). New York, NY: Springer. p 108 (2006).
[54] A.H. Boozer, "Plasma equilibrium with rational magnetic surfaces", Phys. Fluids **24**, 1999 (1981).

[55] Ferguson Jr, and E. Warren “A simple derivation of Glassman’s general N fast Fourier transform" Computers & Mathematics with Applications 8, no. 6 401-411, (1982).