Optimal mesh for global tokamak PIC simulation

A Thesis submitted for the completion of
requirements for the degree of

Bachelor of Science
(Research)

by

Amal R Biju
Undergraduate Programme
Indian Institute of Science

Under the supervision of

Prof. Animesh Kuley
Department of Physics, Indian Institute of Science

To my God and my beloved family

Declaration

I declare that this work is my own and was carried out in the Plasma Theory group of Prof. Animesh Kuley in the Physical Sciences Department, Indian Institute of Science, Bengaluru. It was carried out during my undergraduate for a Bachelor of Science (Research) Degree at the Indian Institute of Science from 2018 to 2022. Wherever I have referred to another author's work, I have rightly attributed it in the references. I have acknowledged the people who have helped me in the lab carry out my work.

Amal R Biju
Undergraduate Program
Indian Institute of Science
Bengaluru, PIN: 560012
email:amalbiju@iisc.ac.in

Acknowledgements

I want to express my gratitude to Prof. Animesh Kuley, my supervisor, for making this thesis possible and advising me throughout the time. I want to acknowledge the GTC-X team, including Jaya Kumar A, Joydeep Das, Sourav Bag, Raghav T S, and Giridharan R P. Animesh, the leading developer of GTC-X, was always there when I had any questions about equilibrium quantities or the physics of fusion devices in general. Jay Kumar was my first contact when I had any questions about the approaches we take in GTC-X. He also helped me understand the GTC-X through lots of discussions and also helped me by evaluating the methods and algorithms that I introduced. I would also like to thank my friend Nivedith K A, who helped me analyze my code in the initial stages of my thesis.

Finally, I am deeply grateful to my family for supporting me throughout my life and during the work on this thesis.

Abstract

Performing gyrokinetic simulations of core and SOL plasmas combined is particularly difficult. First, the plasma density and temperature are low in the SOL, and the turbulent fluctuations are the same size as the background plasma. This is entirely different from what happens in the core region. Second, the magnetic geometry of SOL is complex. It contains regions of open magnetic flux surfaces and X-points. This renders flux coordinates, commonly used in gyrokinetic simulations of the plasma core, ineffective.

The gyrokinetic toroidal code (GTC) has been upgraded for global simulations by coupling the core and scrape-off layer (SOL) regions across the magnetic separatrix. In my thesis, the improved code's ability to construct field-aligned grids, regardless of the geometry of the tokamak, is showcased through the creation of such grids using the realistic equilibrium data from DIII-D. This involved loading experimental equilibrium data from g-file and generating field lines and field-aligned grids. I have confirmed the obtained flux surfaces with the experimental one.

The meshing or triangulating of the grids is the next step as it is essential in particle tracking and solving equations of physical quantities on them. Meshing is done by TRIANGLE software as it creates extra grid points and enables us to customize the meshing which suits our needs. I have also discussed how we can customize our mesh. Finally, we propose some methods to optimize the mesh, which ultimately helps to solve equations with more minor errors, especially field estimations using finite element methods.

Contents

List of Figures

Part I

Introduction and Basic developments

Chapter 1

Motivation and Introduction to Tokamak

The climate of our planet is changing dramatically. The effects of these are the melting of glaciers and rising sea levels. As a consequence of this natural habitat of many species is destroyed [1]. This trend cannot be explained by natural climate variability; instead, it is caused by the emission of greenhouse gases [2]. To limit the effects of global warming, it is necessary to limit the emission of greenhouse gases and develop new environmentally friendly energy sources.

From inspiration from stars' energy production, fusion energy offers a unique pathway to achieve that goal. Energy production in fusion is carbon-free and sustainable. Consequently, developing net energy-producing fusion experiments as the first step toward a fusion power plant has been a dream since the 1950s.

1.1 Nuclear fusion

In our universe, many stars like our sun produce energy from the fusion of hydrogen by the proton-proton chain [3]. At temperatures and densities producible in the laboratory, these reactions have a low cross-section. Therefore, they are not suited for fusion reactors on earth. The most promising fusion experiments are done with two isotopes of hydrogen, i.e., deuterium and tritium (D-T). This is because the D-T fusion process has the highest known cross-section at experimentally relevant parameters and is, therefore, the easiest to access [4]. Since the products of this fusion reaction have a slightly lower mass than the inputs, this difference in mass is converted to kinetic energy by the phenomenon known as mass defect. During the D-T fusion process, visualized in Fig.1.1, a neutron with 14.1 MeV energy and a helium atom with 3.5 MeV energy is released. Due to the Coulomb force between nuclei, they repel each other, and become difficult to fuse. To make fusion

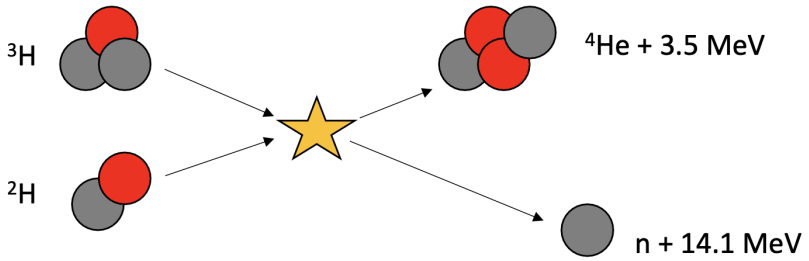


Figure 1.1: Figure displaying the fusion reaction of deuterium and tritium to helium and a neutron. The reaction releases a total of 17.6 MeV energy [6]

reactions possible, the two nuclei need to get close enough such that quantum tunneling through the Coulomb barrier is likely to happen. This can either be achieved by high temperatures or high pressures. In these conditions, the atoms are ionized and form the state of a plasma. Part of the energy produced in the fusion process stays within the plasma and makes, in theory, a self-sustained fusion reaction possible. To obtain such an energy-producing, also called burning, plasma, the product of its temperature T, density n, and energy confinement time τ_E must exceed the threshold [5].

$$nT\tau_E > 3.10^{21} KeV s m^{-3}$$

1.2 Confinement and plasma stability

The plasma is confined by gravity in the stars, where the temperature and densities are large and have a large energy confinement time. So the reaction in stars is self-sustained and produces energy itself. There are different ways to confine the plasma, and the most promising one is the confinement using a magnetic field. Due to the Lorentz force, charged particles are trapped in a spiral-like motion around magnetic field lines. They bend into a toroidal shape to avoid particle losses at the end of field lines. The two most promising concepts for fusion power plants are the tokamak, and the stellarator [7]. In tokamaks densities around $10^{19}/m^3$ $10^{20}/m^3$, Temperatures of 1KeV 40KeV and energy confinement times of τ_E = 0.1s - 1s can be achieved [5, 8].

In first approximation, the plasma and its interaction with the magnetic field $\vec{B}$ in these fusion devices can be described as a fluid within magnetohydrodynamics (MHD). The equilibrium state of the plasma is classified by the force balance [9] between the Lorentz force $\vec{j} \times \vec{B}$, caused by the plasma current $\vec{j}$, and plasma pressure 'p'. the

variable 'c' denotes the speed of light.

$$\vec{j} \times \vec{B} = c\nabla p$$

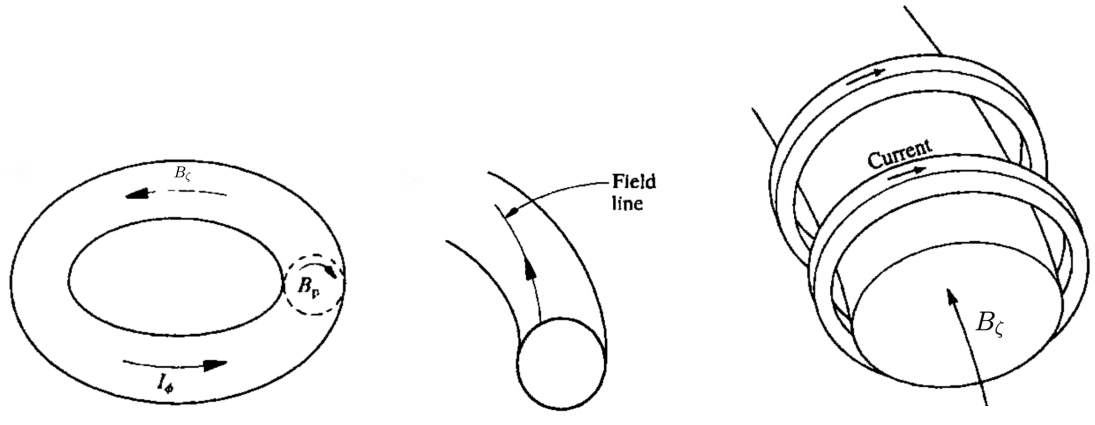
Like the tokamak, every magnetic field line generates a magnetic flux surface in axisymmetric devices. The pressure is constant on a magnetic flux surface, and particles stream freely along magnetic field lines.

1.3 Tokamak

The tokamak is a toroidal plasma confinement system, the plasma being confined by a magnetic field. The principal magnetic field is the toroidal field. However, this field alone does not allow confinement of the plasma. In order to have an equilibrium in which the magnetic forces balance the plasma pressure, it is also necessary to have a poloidal magnetic field. In a tokamak, this field is produced mainly by the current in the plasma itself, and this current flow in the toroidal direction. These currents and fields are illustrated in Fig.1.2. The combination of the toroidal field B_ζ and poloidal field B_θ gives rise to magnetic field lines, which have a helical trajectory around the torus, as shown in Fig.1.2. The toroidal magnetic field is produced by currents in coils linking the plasma, as shown in Fig.1.2.

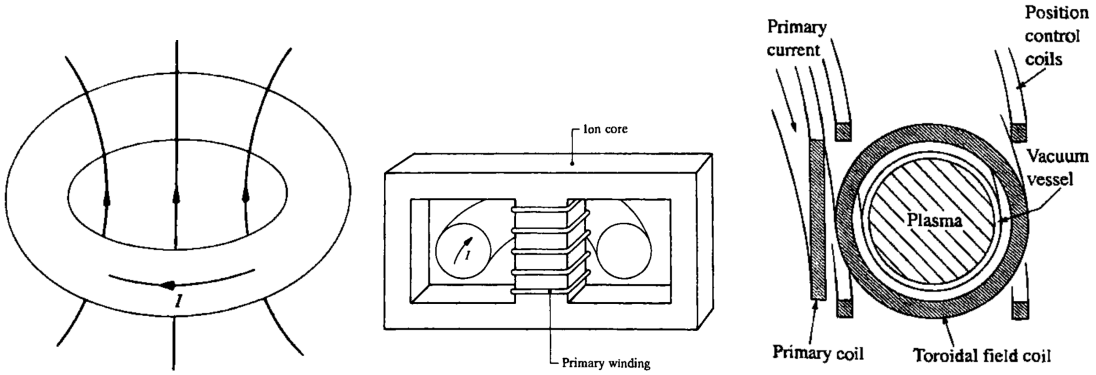
The plasma pressure is the product of the particle density and temperature. The fact that the reactivity of the plasma increases with both of these quantities implies that in a reactor, the pressure must be sufficiently high. The pressure which can be confined is determined by stability considerations and increases with the strength of the magnetic field. However, the magnitude of the toroidal field is limited by technological factors. For a given magnetic field, the plasma pressure which can be stably confined in the tokamak increases with toroidal plasma current, but only up to a limiting value. The resulting poloidal magnetic fields are typically an order of magnitude smaller than the toroidal field. The flux change through the torus is responsible for the toroidal electric field, which in turn drives the plasma current, as illustrated in Fig1.3(a). A current makes the flux change passed through a primary coil around the torus, as shown in Fig1.3(b). The complete system of toroidal and poloidal coils is illustrated in Fig1.3(c). The confinement in the tokamaks is done mainly by crossed toroidal(predominantly) and poloidal magnetic fields. This confinement constrains the plasma particles to gyrate and translating along the field lines. This prevents the particles from colliding with the walls. Since the magnetic field used mainly is toroidal, the gradient of this field points toward the torus. Hence, the tokamak's inner walls experience a large magnetic force due to this gradient, known as the "grad-B force". So early tokamaks, which had circular cross-

sections, experienced tremendous force in their inner wall. Modern tokamaks have a D-shaped cross-section, where the flat side of the D lies on the inside of the torus. This flat surface evens out the force acting on the different portions of the inner wall, thereby reducing its pressure. Due to this D shape, plasma particles may drift away from the field lines(they may also drift due to curvature of the magnetic field and also due to the $\vec{E} \times \vec{B}$ force). Due to this drift, plasma particles collide with tokamak walls and cause sputtering. This would not only damage the walls, but the sputtered material from walls also enters plasma, heat up, and cool down, which hinders the fusion. These impurities in the plasma give rise to radiation losses and also dilute the fuel. Therefore, the restriction of their entry into the plasma plays a fundamental role in the successful operation of tokamaks. Two techniques currently used are, the first is to define an outer boundary of the plasma with a material limiter as shown in Fig1.4(a), the second is to keep the particles away from the vacuum vessel through a modification of the magnetic field to produce a magnetic divertor as shown in Fig1.4(b). From the first method, the limiter is a covering made of light materials such that the plasma particle collides with them instead of the inner walls of the torus. This prevents the wall from being damaged and reduces the impurities. However, since particles collide with limiter, the problem of sputtering remains; further, there is also the production of fusion ash (heavier ions produced as a result of fusion which remained within the fuel in the tokamak) persisting. To overcome this, we use the second method, i.e., the introduction of divertor plates. The divertor plates changed the geometry of the plasma within the torus. It divided the cross-section of the torus into two regions - the core, having closed field lines, and the scrape-off layer (SOL), consisting of open field lines, separated by a magnetic separatrix which is the last closed flux surface of the core region.



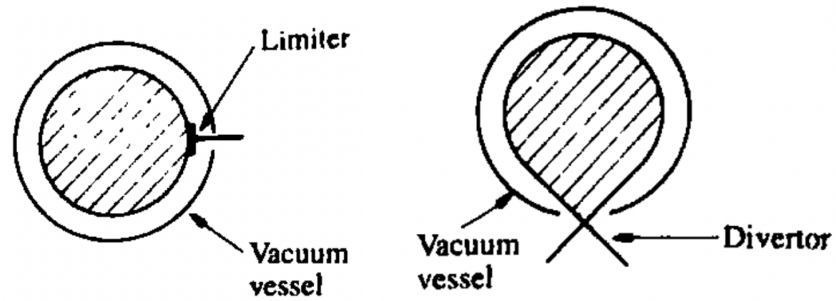
(a) (b) (c)

Figure 1.2: (a)Toroidal magnetic field B_ζ, and poloidal magnetic field B_P due to toroidal current I_ζ. (b)Combination of B_ζ and B_P causes the field lines to twist around plasma. (c)Toroidal magnetic field is produced by current in external coils. [5]



(a) (b) (c)

Figure 1.3: (a)Change of flux through torus induces a toroidal electric field which drives the toroidal electric current. (b)The flux change is produced by primary winding. (c)Arrangement of coils in a tokamak.[5]



(a) (b)

Figure 1.4: Seperation of plasma from vacuum vessel by (a)limiter and (b)divertor.[5]

There is also another region, known as the private region, lying below the magnetic separatrix in Fig1.5. The separatrix between the private and SOL regions and that between the core and the SOL touch at a given point is the X-point. At the X-point, the magnetic flux goes to zero. The presence of the divertor elongates the plasma, bringing it away from the walls. More importantly, it shapes the field lines such that the heavier ablated impurities and the fusion ash descend downwards due to their weight, follow the field lines along the separatrix, and escape out of the plasma through the divertor plates. This significantly improves the sustainability of the plasma at high temperatures. Modern-day tokamaks have either one (e.g., ITER, France; DIII-D, U.S.A.) or two (e.g., EAST, China) divertors.

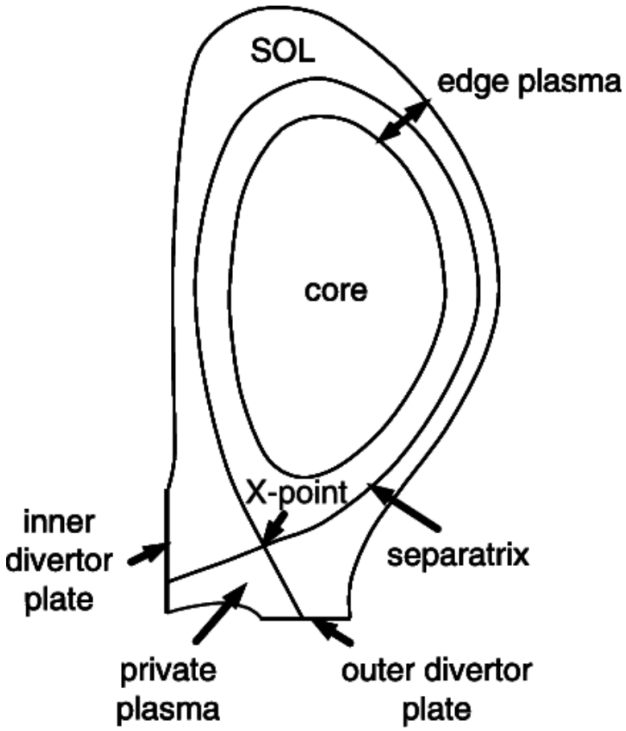


Figure 1.5: A typical poloidal cross section of a tokamak having a single divertor.

1.4 Simulation approaches

One of the important challenges in achieving a proper operating condition in ITER and future fusion reactors is associated with the nonlinear turbulant dynnamics of the plasma in SOL region [10]. The plasma comes in contact with main wall in boundary region. So to avoid damage, the plasma should be cooled in orders of magnitude before it collides with the wall. Therefore, the boundary region (SOL region) of magnetic confinement fusion devices is characterised by steep density and temperature gradients, which drives microinstabilities and turbulance. The characteristics of plasma in SOL region have many effects such as, it affects the confinement properties of tokamak, regulate the heat transport to the walls, influence the level of fusion ash, the impurity dynamics and plasma shaping effects. Since the radio waves send for heating first interacts with plasma in SOL region, it plays a significant role in radio-frequency heating. But expanding the simulations from the core to the edge and SOL is difficult. In the boundary region the gradients are steep and the temperature and density is low. As a result, the am- plitude of the turbulent fluctuations is comparable to the plasma background. So these spatial and temporal scales associated with different instabilities in that region is a major challenge. Furthermore, the geometry of the magnetic equilibrium is complex in that region.

Simulations of fusion plasmas are based on the gyrokinetic equations, which is a set of

nonlinear, integro-partial differential equations that describe the evolution of the distribution of particles in a five-dimensional phase space [11]. Due to the high computational demand of gyrokinetic simulations and the increased collsionality in the edge and SOL, fluid and gyrofluid models are often used [12, 13, 14, 15, 16, 17]. They assume that collisions move the velocity distribution of particles close to a Maxwellian that can be described with a few moments like density and temperature. They are either based on the drift reduced Braginskii equations [18, 19] or on gyrofluid models [20]. However, kinetic effects are known to be important in the edge and SOL regions [21, 22, 23] such as ion orbit losses, X-point losses and plasma sheath dynamics to name a few. So we need a kinetic approach that can couple the different geometries of the core and SOL regions across the magnetic separatrix. Such a study will provide helpful insights into the dynamics of the plasma in the SOL, which will be of utmost importance in the design of present and future fusion reactors.

The simulation codes are based on different numerical approaches with their respective advantages and disadvantages. The different approaches are split into three categories.

The particle-in-cell approach

The first gyrokinetic plasma simulations were based on the particle-in-cell (PIC) approach [24, 25, 26]. Here a sample of so-called markers are integrated along with the characteristics of the gyrokinetic Vlasov equation, and the electromagnetic fields are solved on a three-dimensional grid in position space. Particle-in-cell codes are straightforward to implement and have shown good scalability on modern supercomputers [27]. Due to the probabilistic nature of these algorithms, the results contain statistical noise, which has shown to be problematic for specific applications [28, 29, 30]. Many markers are needed to resolve the full distribution function in SOL plasmas and keep the numerical noise at a manageable level, which may require more immense computational resources than comparable continuum codes.

The Eulerian approach

The gyrokinetic Vlasov-Maxwell system is discretized on a five-dimensional position-velocity space grid in the continuum or Eulerian schemes. They use a variety of numerical methods available for partial differential equations. This includes finite difference, finite volume, finite element, and discontinuous Galerkin methods with explicit and implicit time-stepping schemes. Compared to PIC codes, continuum codes are free of numerical noise at the expense of introducing a velocity space grid.

The semi-Lagrangian approach

In the semi-Lagrangian approach, the gyrokinetic Vlasov-Maxwell system is, similar to Eulerian schemes, discretized on a five-dimensional position-velocity space grid. Therefore, they are also free of numerical noise. Semi-Lagrangian schemes are commonly divided into three categories: (i) The (standard) backward, (ii) the forward, and (iii) the conservative semi-Lagrangian method. Semi-Lagrangian schemes can allow for larger timesteps than comparable Eulerian schemes, which makes them, next to fusion applications, also attractive for numerical weather predictions and atmospheric models [31].

1.5 Handling X-point geometries

The main challenge in extending simulations from core to SOL is the complex magnetic geometry in these regions. There are regions of open and closed magnetic field lines. Many gyrokinetic codes present in the community implement so-called flux-coordinate systems [**t73**]. While these allow for efficient simulations in regions with regular magnetic flux surfaces, they become ill-defined at the separatrix [32]. To allow for predictive SOL simulations, coordinate systems free of singularities that can treat realistic magnetic geometries must be used. PIC codes can perform simulations in X-point geometries by using a cylindrical coordinate system. PIC codes follow the trajectories of marker particles in continuous real space, and therefore, parallel structures in the plasma are well represented.

1.6 GTC to GTC-X

We have seen above that studying the dynamics of the SOL region and coupling it with core across the separatrix is essential for present and future tokamak devices. A significant advancement in this regard was the development of a kinetic code, the gyrokinetic toroidal code (GTC) [33], which has been successful in studying turbulence and transport properties in the core. The GTC is a well-benchmarked code based on first principles. However, it uses conventional magnetic coordinates where the equations of motion encounter a singularity of the metric at the magnetic separatrix. This is mainly because poloidal magnetic fields vanish at X-point(s). As a result, it cannot couple the core and SOL regions. Moreover, the poloidal angle suffers a discontinuity across the separatrix.

A significant enhancement that we have made over the GTC is using a cylindrical coordinate system (R, Z, ζ). R is the radial coordinate in the toroidal plane, Z represents the direction perpendicular to the toroidal plane, and ζ is the toroidal angle. The new

coordinate system overcomes all the difficulties mentioned above faced by GTC, leading us to name our code GTC-X (X standing for the X-point, since the new code does not suffer from the singularity at the X-point unlike GTC). Using a cylindrical coordinate system to advance particle dynamics allows particle motion in arbitrarily shaped flux surfaces, including the magnetic separatrix and the X-point(s), which GTC could not have done.

In the new code, we have developed a method of field-aligned particle-grid interpolations using an axisymmetric mesh in the cylindrical coordinates. The grids can be made as per the resolution requirements of the simulation. These field-aligned particle-grid interpolations can achieve the same order of numerical efficiency as the field-aligned mesh in magnetic flux coordinates used earlier in GTC [34]. Then these grids are used to construct triangulation, where the PIC model approach uses these triangles, and these grids are also used to solve for various physical quantities like the electric field, density, and temperature. We also discuss how we can modify the triangulation to meet our necessities and optimize it.

Chapter 2

Equilibrium

2.1 Tokamak equilibrium

The tokamak equilibrium has two fundamental aspects. Firstly is the internal balance between the pressure of the plasma and the forces due to the magnetic field. Secondly, the shape and position of the plasma are determined and controlled by currents in external coils.

We know that the main component of the magnetic field is the toroidal field produced by poloidal currents in external coils. A smaller, poloidal magnetic field is produced mainly by the toroidal current in the plasma. The entire poloidal field comprises this internally generated field due to toroidal currents in the primary winding and other coils used for plasma shaping and control. Finally, the toroidal magnetic field is modified by poloidal currents in the plasma to a small extent.

The general form of the spatial variation of the magnetic field is illustrated in Fig2.1. Ampere's law obtains the basic shape of the toroidal field B_ζ. Thus, taking a line integral around a circular toroidal circuit inside the toroidal field coils, neglecting the small poloidal plasma current.

$$2\pi R B_\zeta = \mu_0 I_T$$

Where R is the major radius coordinate and I_T is the total current in the coils. From above equation.

$$B_\zeta \propto \frac{1}{R}$$

In present tokamaks this radial fall is significant. Taking a minor radius a, the change in B_ζ across the plasma is

$$\Delta B_\zeta = B_{\zeta 0}\left(\frac{R_0}{R_0 - a} - \frac{R_0}{R_0 + a}\right)$$

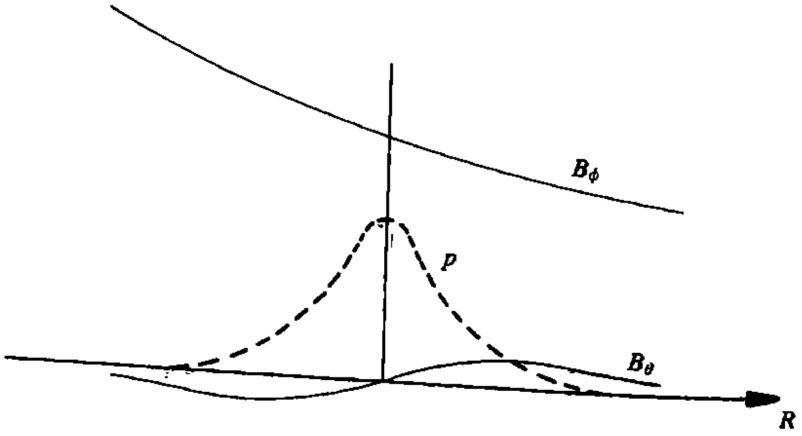


Figure 2.1: Basic radial variation of the toroidal magnetic field B_ζ (with an approximate 1/R dependence), the poloidal magnetic field B_θ, due mainly to the plasma current, and the plasma pressure, p. R is the distance from the major axis of the torus and the graphs are for the median plane. [5]

$$\approx B_{\zeta 0}\frac{2a}{R_0}$$

where $B_{\zeta 0}$ is the field at the mid-plane R=R_0. For an aspect-ratio R/a=3, $\Delta B_\zeta/B_{\zeta 0} = \frac{3}{4}$. This variation of B_ζ across the plasma has an important effect on the trajectories of the plasma particles.

The distribution of the poloidal magnetic field depends on the toroidal current profile. The electrical conductivity determines the steady-state current profile if a toroidal electric field drives this current. Since this increases with electron temperature as $T_e^{3/2}$, the current is peaked in the central region where the temperature is highest. It is possible to drive the current by non-ohmic means by radiofrequency waves. Another component of the plasma current is the so-called bootstrap current. This current arises from the radial gradients of density and temperature and the toroidal magnetic geometry.

Because the mass density of the plasma is very low and the component internal forces on the plasma are pretty large, the internal forces must balance. The plasma pressure exerts an outward force across the minor radius, and the poloidal magnetic field exerts an inward force. The imbalance between these two forces is taken up by the magnetic pressure of the toroidal magnetic field. This force may be inward or outward.

The entire magnetic field produces an infinite set of nested toroidal magnetic surfaces. The field lines follow a helical path on their magnetic surface as they wind around the torus. The sound speed in plasma is typical $10^5 ms^{-1}$ - $10^6 ms^{-1}$ and consequently, the pressure is constant along the magnetic field, any imbalance being rapidly removed.

The direction of the magnetic field changes from surface to surface. This shear of the magnetic field has important implications for the stability of the plasma. The average twist of the magnetic field on each surface can be characterized by the so-called safety

factor, q, which measures the pitch of the helical field lines. The shear is then determined by the radial rate of change of q.

The toroidal geometry of the plasma leads to a force that expands the plasma ring. This force is balanced by applying a vertical magnetic field. This force interacts with the toroidal current to give an inward force.

While the equilibrium is partly determined by external conditions such as total current, applied toroidal magnetic field, and the applied heat, many characteristics result from the plasma behavior. Thus the density and temperature profiles are governed by the transport properties, which are affected by plasma instabilities.

2.2 Flux functions

For axisymmetric equilibria, that is, equilibria independent of the toroidal angle ζ, the magnetic field lines lie in nested toroidal magnetic surfaces as illustrated in Fig 2.2. The essential condition for equilibrium is that the force on the plasma is zero at all points. This requires that the magnetic force balances the force due to the plasma pressure, that is

$$\vec{j} \times \vec{B} = \nabla p$$

It is clear from this equation that $\vec{B} \cdot \nabla p = 0$. Thus, there is no pressure gradient along the magnetic field lines, and the magnetic surfaces are surfaces of constant pressure. Furthermore above equation gives $\vec{j} \cdot \nabla p = 0$, and consequently, the current lines also lie in the magnetic surfaces as shown in Fig 2.3. In studying tokamak equilibria, it is convenient to introduce the poloidal magnetic flux function ψ. This function is determined by the poloidal flux lying within each magnetic surface and is, therefore, a constant on that surface. Thus ψ satisfies

$$\vec{B} \cdot \nabla \psi = 0$$

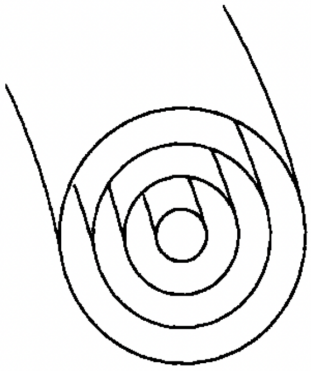


Figure 2.2: Magnetic flux surfaces forming a set of nested toroids. [5]

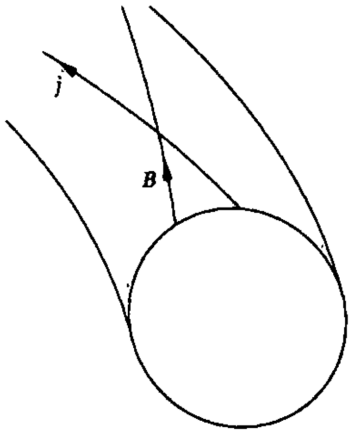


Figure 2.3: Magnetic field lines and current lines lie in magnetic surfaces. [5]

Introducing a cylindrical coordinate system based on the major axis of the torus, as shown in Fig 2.4. and defining ψ to be the poloidal flux per radian in ζ, the poloidal magnetic field is related to ψ by

$$B_R = -\frac{1}{R}\frac{\partial \psi}{\partial Z}, \quad B_Z = \frac{1}{R}\frac{\partial \psi}{\partial R}$$

conforming to the requirement $\nabla \cdot \vec{B} = 0$, that is

$$\frac{1}{R}\frac{\partial (RB_R)}{\partial R} + \frac{\partial B_Z}{\partial Z} = 0$$

The flux function ψ is arbitrary to an additive constant which may be chosen for convenience.

From the symmetry of $\vec{j}$ and $\vec{B}$ it is clear that a current flux function also exists. This function, f, is related to the poloidal current density by

$$j_R = -\frac{1}{R}\frac{\partial f}{\partial Z}, \quad j_Z = \frac{1}{R}\frac{\partial f}{\partial R} \quad \texttt{(poloidal current density equations)}$$

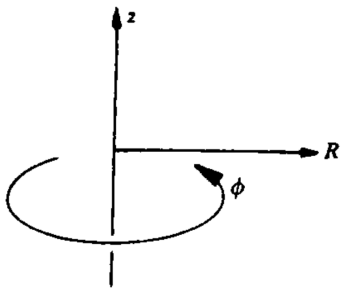


Figure 2.4: Cylindrical coordinate system. R=0 is the major axis of torus. [5]

Comparing the above equations with Ampere's equation,

$$j_R = -\frac{1}{\mu_0}\frac{\partial B_\zeta}{\partial Z}, \quad j_Z = \frac{1}{\mu_0}\frac{1}{R}\frac{\partial (RB_\zeta)}{\partial R}$$

gives the relation between f and the toroidal magnetic field as

$$f = \frac{R\vec{B_\zeta}}{\mu_0}$$

It can be shown that f is a function of ψ by taking the scalar product of $\vec{j} \times \vec{B} = \nabla p$ with $\vec{j}$, to obtain $\vec{j} \cdot \nabla p = 0$, and then subtituting in poloidal current density equations for $\vec{j}$. Thus

$$\frac{\partial f}{\partial R}\frac{\partial p}{\partial Z} - \frac{\partial f}{\partial Z}\frac{\partial p}{\partial R} = 0$$

and hence

$$\nabla f \times \nabla p = 0$$

proving that f is a function of p. Then, since $p = p(\psi)$, it follows that $f = f(\psi)$.

The flux functions as defined here give the poloidal flux associated with a unit angle in ζ.

2.3 Experimental equilibrium data

We used experimental equilibrium data generated by the discharges of the DIII-D tokamak of the United States for our work here. All the result obtained here is for the case of DIII-D. Results such as the reading of the equilibrium parameters, field calculation, grids construction, and triangulation of poloidal 2D cross-section are done for DIII-D, but the algorithms used here can be used for any tokamaks with single divertors. They can also be used for tokamaks with multiple divertors (with slight modification). Fig 2.5 shows some of the equilibrium data which we used in our work.

2.4 Normalisation

For the numerical implementation, it is helpful to renormalize the equations. With the normalization, different physical quantities can be scaled to numerical values of order O(1). This is beneficial because floating-point numbers are most dense around 1, and therefore smaller differences between numbers can be resolved. Furthermore, it is possible to choose different normalizations for different species. All the quantities we use here have a dimension which is a combination of magnetic field and length. So for normalization we

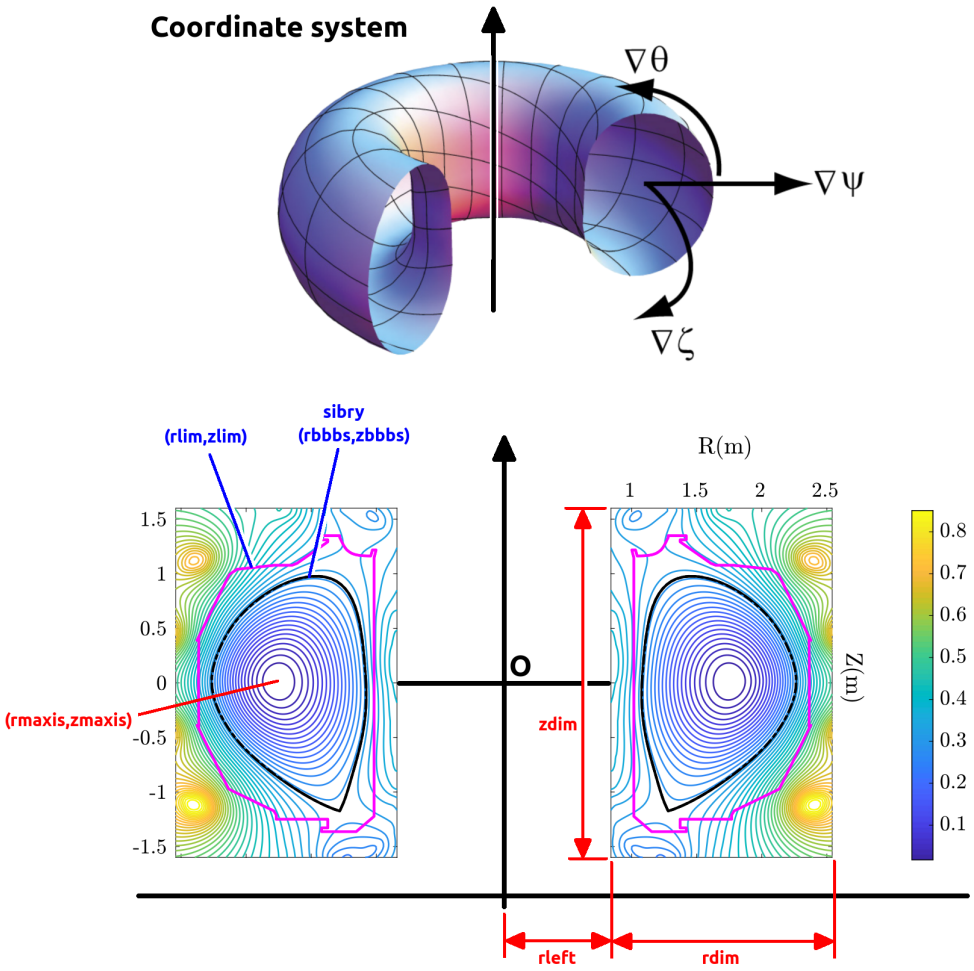


Figure 2.5: Geometric view of few experimental equilibrium data generated by the discharges of the DIII-D along with coordinate system used.

divide the quantity with its dimention, were the dimentional quantities (magnetic field and length) are taken as its value at magnetic axis

By normalization we need the ψ value at magnetic axis to be 0. So we add the negative of ψ value at magnetic axis to all the ψ values, before normalization. Since the dimention of ψ is $[BL^2]$, the normalization of ψ is done as,

$$\psi_{new} = \psi + (-\psi_{mag})$$

$$\psi_{normal} = \frac{\psi_{new}}{B_{mag}.L_{mag}^2}$$

Similarly for poloidal flux function,

$$F_{normal} = \frac{F(\psi)}{B_{mag}.L_{mag}}$$

For all other magnetic field and length quantities, we divide it by B_{mag} and L_{mag} respectivily.

In our case,

$$B_{mag} = 1.8964T$$

$$L_{mag} = 1.7591m$$

$$\psi_{mag} = -0.2499Tm^2$$

Chapter 3

Geometric analysis and introduction to TRIANGLE

An adequate representation of the analysis domain is a non-manifold boundary topology with associated shape information in a geometry-based analysis environment [35]. A geometric model boundary representation is a hierarchy of regions, shells, faces, loops, edges, and vertices,(see Fig 3.1) and uses entities for vertices, edges, loops, and faces [36]that can effectively define the adjacencies seen in analysis model idealizations of physical domains. The geometric model is an important input for reliable automatic meshing [37].

The mesh is a discretized representation of the geometric model. It consists of the four types of topological entities, which are regions (3D), faces (2D), edges (1D), and vertices (0D), with controlled size, shape, and distribution [39, 40, 41]. Concerning the simulation, a mesh entity that does not bound any higher dimensional entities has termed an element.

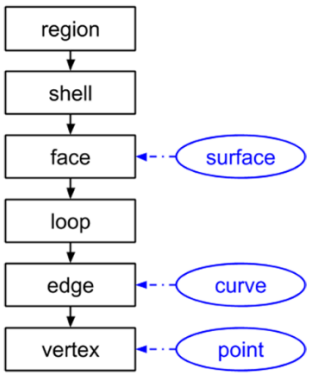


Figure 3.1: Topological entities (rectangles) and associated shape infor- mation (ellipses) in the geometric model. [38]

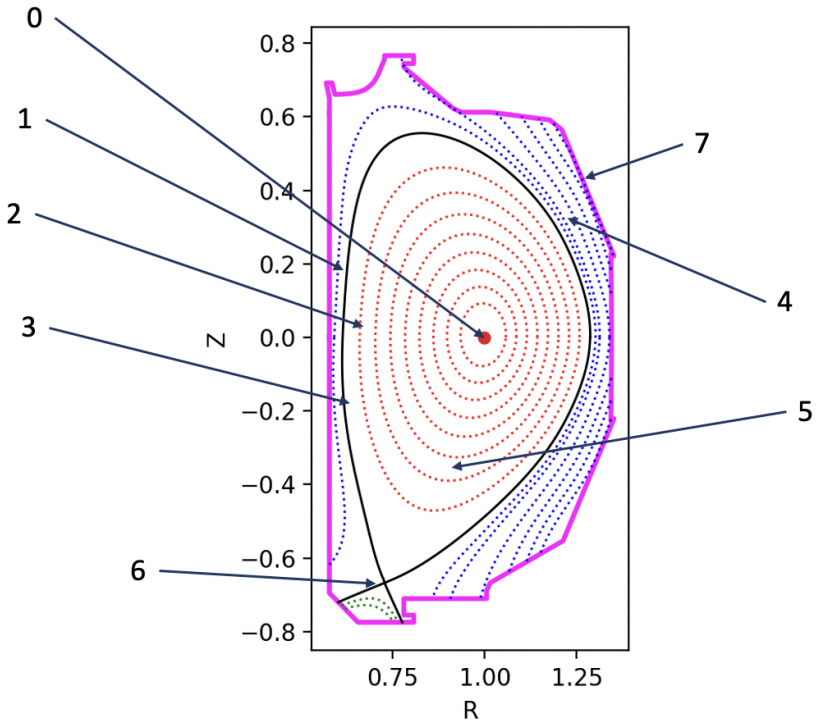


Figure 3.2: The model components include the (0) magnetic axis, (1) open magnetic flux surfaces, (2) closed magnetic flux surfaces, (3) separatrices, (4) scrape-off layer, (5) plasma core, (6) X-points, (7) limiter points.

3.1 Geometric description

We know tokamak devices use a magnetic field to confine the plasmas for sustainable fusion reactions. The basic tokamak geometry is symmetric torus along the toroidal direction. Magnetic field lines wind around the torus and form magnetic flux surfaces [5]. The meshing procedure first generates the 2D geometric model and generates a 2D mesh based on the geometry.

Fig 3.2 illustrates the 2D geometric model of the toroidal cross section of DIII-D. In order to comply with different meshing requirements of the plasma simulation codes, the geometry of the tokamak contains the combination of the physics and physical components. The physics components are labeled with the numbers from 0 to 6 and physical components are labeled with the number 7.

A separatrix (3 in Fig. 2) adjacent to the magnetic axis (0 in Fig. 2) divides the 2D poloidal cross-section into two distinct areas, the scrape-off layer (4 in Fig. 2) and the plasma core (5 in Fig. 2). The scrape-off layer is the region between the separatrix and limiter points(Wall). In the plasma core, axisymmetric equilibria of the plasmas, the magnetic flux surfaces (2 and 3 in Fig. 2) are nested toroids, forming non-intersecting loops on the 2-dimensional toroidal cross-section between the magnetic axis and the

separatrix. The magnetic field lines diverge at the X-point (6 in Fig. 2), which is a minimax point of the magnetic flux field on the separatrix. The flux surfaces intersect with the wall in the area of the scrape-off layer. The plasma area (9 in Fig. 2) is the inner area bounded by the limiter points (boundary) [5].

The geometry of DIII-D contains the plasma area with the scrape-off layer (4 in Fig. 2) separated by the separatrix (3 in Fig. 2) and the plasma core (5 in Fig. 2). It also includes magnetic flux surfaces (1 and 2 in Fig. 2) in the plasma area and the inner wall boundary (limiter points) to meet the requirement of placing mesh vertices.

3.2 Topological representation

Fig 3.3 illustrates the topological structure of model faces in DIII-D geometry. Faces interior to the separatrix are bounded by two loops corresponding to a pair of adjacent closed magnetic flux surfaces (faces B and E) or a pair defined by the magnetic axis and the adjacent closed magnetic flux surface (face A). Faces exterior to the separatrix are bounded by one loop corresponding to a collection of open magnetic flux surfaces and a portion of the wall boundary (faces D, C and F).

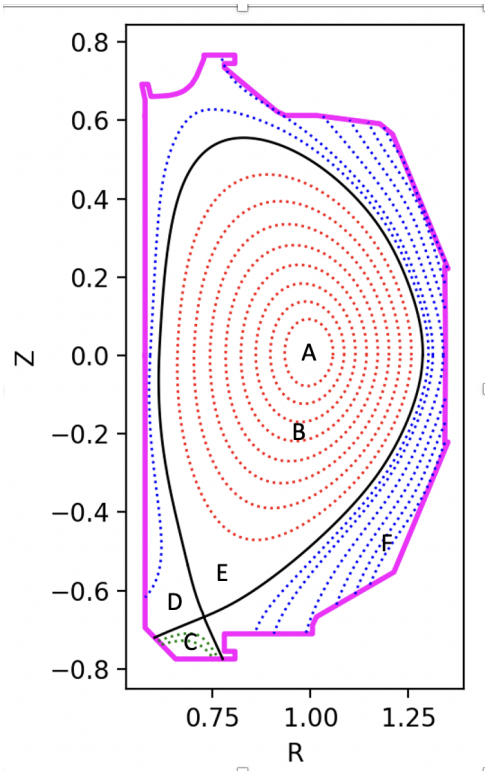


Figure 3.3: Geometric faces on the toroidal cross section in DIII-D

3.3 Shape definition

The critical shape information on the 2D toroidal cross-section is the geometry of curves. There are two kinds of curves in the geometric model. The first is physical domain curves that define the inner/outer walls of the reactor. The second is physics curves that define the features interior of the reactor. A physics curve corresponds to a specific magnetic flux surface.

Physical curves

The physical curves in our physical domain is the geometry of reactor wall which is defined by limiter points.

Physics curves

The physics curves in our domain formed by the intersection of the magnetic flux surfaces and a plane with a constant ψ value. These are flux surfaces made up of common magnetic field lines. Magnetic fields in axisymmetric equilibria can be described by the poloidal magnetic flux field, $\psi = \psi(R, Z)$, and the field related to the poloidal current density, $F(\psi)$ [5]. Given $\psi = \psi(R, Z)$ and $I(\psi)$, the magnetic field $\vec{B}$ is defined as

$$\vec{B} = \nabla\psi(R, \zeta, Z) \times \nabla\zeta + \frac{F(\psi)}{R}\hat{\zeta}$$

$$\vec{B} = B_R\hat{R} + B_Z\hat{Z} + B_\zeta\hat{\zeta}$$

$$B_R = -\frac{1}{R}\frac{\partial\psi}{\partial Z}, \quad B_Z = \frac{1}{R}\frac{\partial\psi}{\partial R}, \quad B_\zeta = -\frac{F(\psi)}{R}$$

The value of ψ does not change along the direction of B from the definition. Therefore, each magnetic flux surface associates with a constant ψ.

3.4 TRIANGLE

TRIANGLE is a two-dimensional mesh generator and triangulator. TRIANGLE can generate exact Delaunay triangulations, conforming Delaunay triangulations, constrained Delaunay triangulations, Voronoi diagrams, and excellent triangular meshes. The triangular meshes can be generated with no small or large angles and are thus suitable for finite element analysis.

3.4.1 Demonstration

TRIANGLE is a software specialized for creating two-dimensional finite element meshes but can also perform more straightforward related tasks such as forming Delaunay triangulations. In the Fig 3.4, we show few things TRIANGLE can do. For finite element analysis TRIANGLE is able to put constrains on the angles while producing triangulations. So number of triangles formed depends greatly on these angle constrains. An example is shown in Fig 3.5.

In the examples in Fig 3.4 and Fig 3.5, the number of triangles formed is governed by the geometry and the angle constraint imposed; Triangle attempts to keep the number of triangles small while creating triangulations so that simulations on Triangle's meshes will not take longer than necessary. But in many cases small triangles are desired to simulate the physics to a sufficient degree of accuracy in many cases. In Fig 3.7 we show different angle constrains in different parts.

Finally, a sequence of successively refined meshes is suitable for multilevel methods. Each of these was generated with a single global area constraint specified on the command line, as opposed to the method of element-by-element imposed constraints. It is illustrated in Fig.

3.4.2 Definitions of few geometric terms

Delaunay triangulation

A triangulation of a particular vertex set with the property that no vertex in the vertex set falls inside of the circumcircle (circle that passes through all three vertices) of any triangle in the triangulation is called Delaunay triangulation of a vertex set. See Fig 3.8(a)

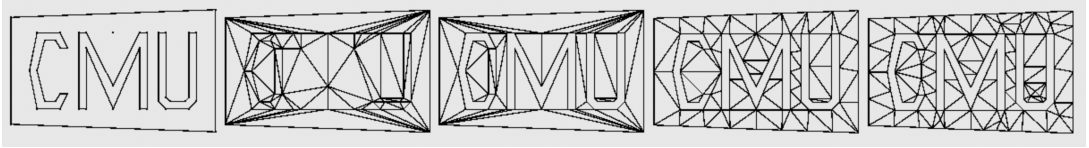


Figure 3.4: This is a sample of Planar Straight Line Graph (PSLG), then a Delaunay triangulation of its vertices by TRIANGLE, then a constrained Delaunay triangulation of the PSLG by TRIANGLE, a conforming Delaunay triangulation of the PSLG by TRIANGLE, and a quality conforming Delaunay triangulation of the PSLG by TRIANGLE with no angle smaller than 25^o. The last triangulation is suitable for finite element analysis. [42]

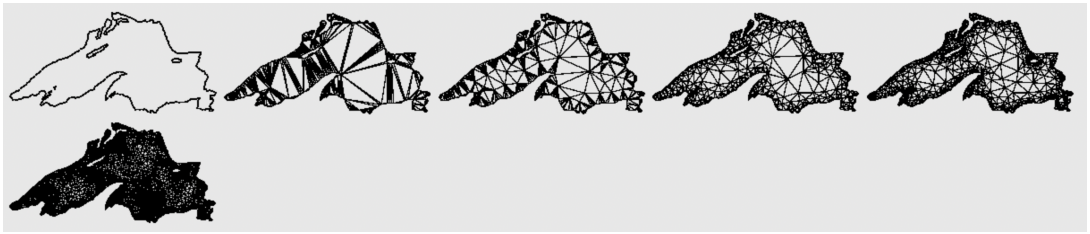


Figure 3.5: Here we have a PSLG of Lake Superior, which is then triangulated with constrains of minimum angles of zero, five, fifteen, twenty-five, and 33.8 degrees. (For angles larger than 33.8 degrees, the Algorithm of TRIANGLE usually does not converge.) [42]

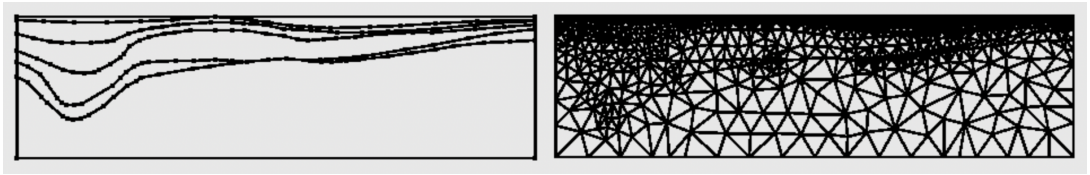


Figure 3.6: Here different maximum angle constrain is imposed in different layers. [42]

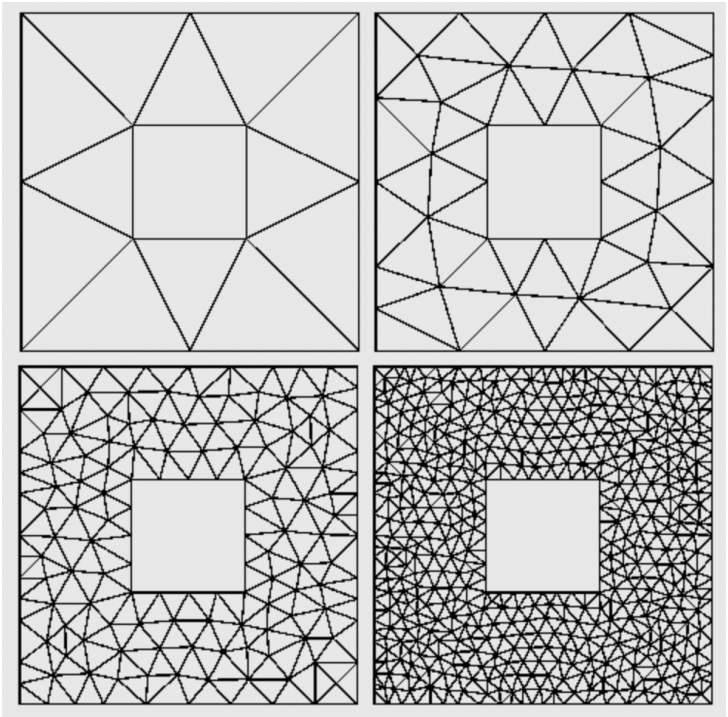


Figure 3.7: Here successive meshes are refined from previous ones. [42]

Voronoi diagram

Dividing the plane into polygonal regions (some of which may be infinite), where each region belongs to the set of points in the plane that are closer to some input vertex than to any other input vertex, is called the Voronoi diagram of a vertex set. (The Voronoi diagram is like a geometric dual of the Delaunay triangulation.). See Fig 3.8(b)

Planar Straight Line Graph (PSLG)

A collection of vertices and segments are known as Planar Straight Line Graph (PSLG). Segments are edges whose endpoints are vertices in the PSLG and whose presence in any mesh generated from the PSLG is enforced. See Fig 3.8(c)

Constrained Delaunay triangulation

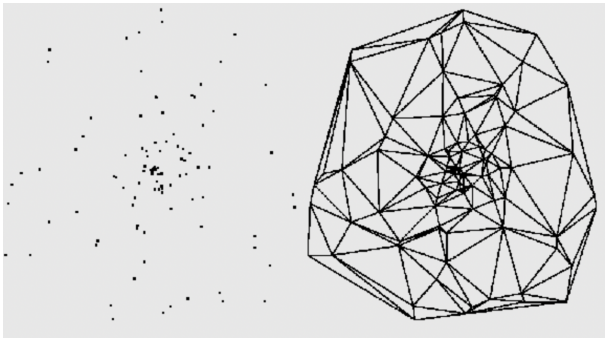
Constrained Delaunay triangulation of a PSLG is a Delaunay triangulation, but each segment does not divide, or the PSLG segment is present as a single edge in the triangulation. A constrained Delaunay triangulation is not a Delaunay triangulation. Its triangles need not be Delaunay, but they are all constrained by Delaunay. See Fig 3.8(d)

Conforming Delaunay triangulation (CDT)

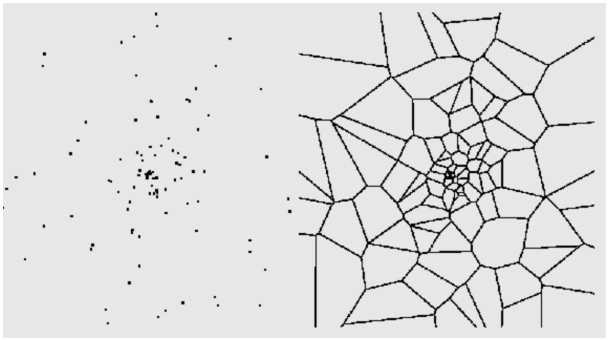
Conforming Delaunay triangulation or CDT of a PSLG is a proper Delaunay triangulation in which using steiner points each PSLG segment have been subdivided into several parts. These points (steiner) are necessary to allow the segments to existing in the mesh while maintaining the Delaunay property. Steiner points are also inserted to meet the minimum angle and maximum triangle area constraints. See Fig 3.8(e)

Constrained Conforming Delaunay triangulation (CCDT)

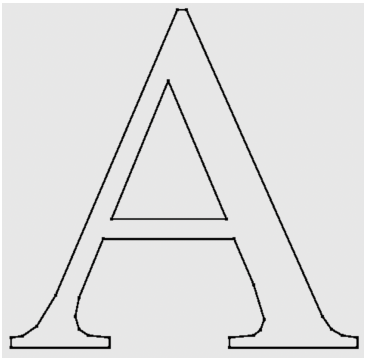
Constrained conforming Delaunay triangulation or CCDT of a PSLG is done as constrained Delaunay triangulation which includes Steiner points.Since the triangles do not need to be Delaunay (although they still must be constrained Delaunay). It usually takes lesser number of vertices to make a good-quality CCDT than a good-quality CDT. See Fig 3.8(f)



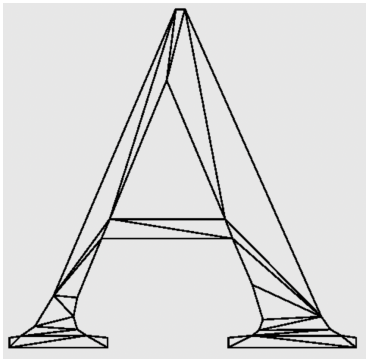
(a) Delaunay triangulation of a vertex set



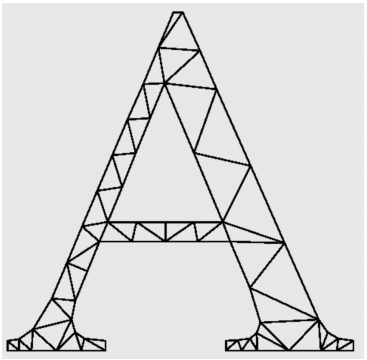
(b) Voronoi diagram of a vertex set



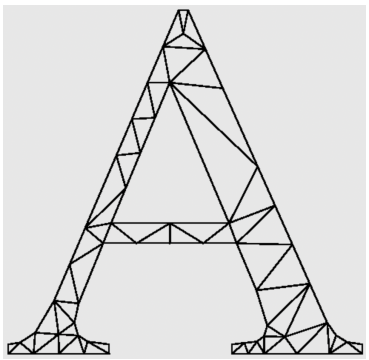
(c) Planar Straight Line Graph (PSLG)



(d) Constrained Delaunay triangulation of a PSLG



(e) Conforming Delaunay triangulation (CDT) of a PSLG



(f) Constrained Conforming Delaunay triangulation (CCDT) of a PSLG

Figure 3.8: Examples of above definitions. [42]

Part II

Construction of Field and Field-aligned grids

Chapter 4

Experimental Data

4.1 Poloidal Current function

Function which represents the current generated by the motion of plasma particles in poloidal direction is called Poloidal current Function. Here the data points for DIII-D shot No:158103 at 3050 ms is provided. We used the MATLAB code, "readEFITgfile.m" to read "g-file" (format "A = readEFITgfile(FILE_NAME)"), which provides us the data for poloidal current function. This data is in the form 1x129 array. we then plot it against Nψ(which ranges from 0 to 129). Plot in Fig 4.1 represents the poloidal current function in equilibrium.

4.2 Poloidal Flux Function

Poloidal flux function which is Poloidal flux ψ at a point P in the (R,Z) cross-section of the plasma is defined to be the total flux through the surface S bounded by the toroidal

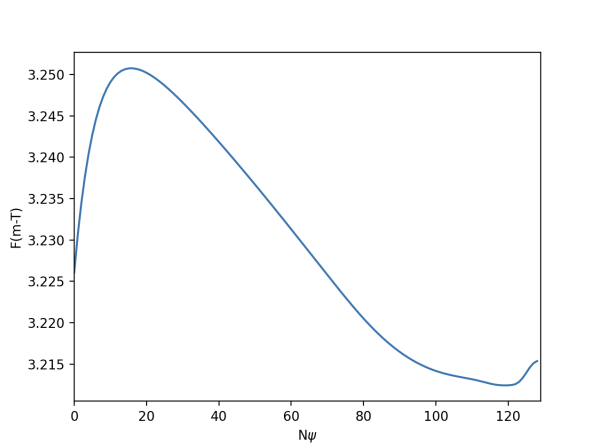


Figure 4.1: Poloidal current function F(ψ) in m-T on a uniform flux grid

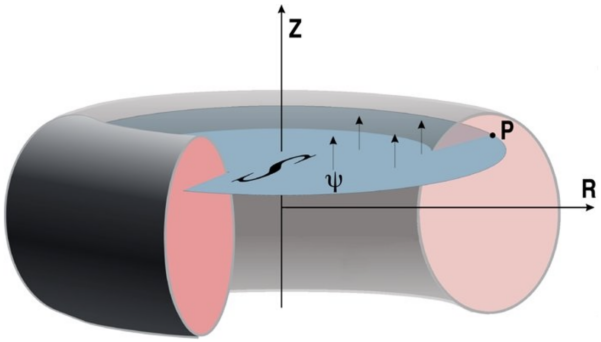


Figure 4.2: Poloidal Flux Function

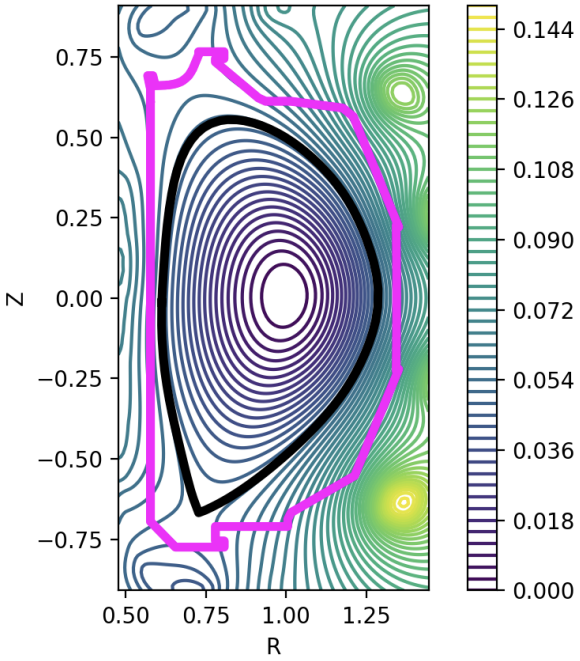


Figure 4.3: Poloidal flux function in web/rad on rectangular (R, Z) grid points for DIII-D shot No. 158103 at 3050 ms. The magnitude of the flux function is indicated by color. The last closed flux surface and limiter points are represented by black and magenta lines, respectively.

ring passing through P(See Fig 4.2). We also get the data for poloidal flux function, which is a 129x129 matrix. After appropriate normalization we plot this matrix as a contour plot, along with last closed flux surface(Separatrix) and limiter points(which represents the wall of the fusion reactor), see Fig 4.3.

4.3 Magnetic Field Components

4.3.1 B_Z and B_R

We know that magnetic field for an axisymmetric system is given by [43].

$$\vec{B} = \nabla\psi(R,\zeta,Z) \times \nabla\zeta + \frac{F(\psi)}{R}\hat{\zeta}$$

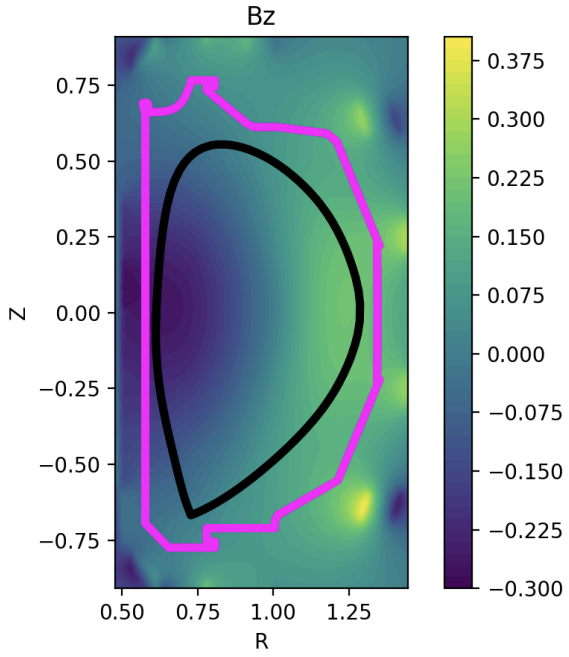
where ψ(R, Z) is the poloidal flux function that labels the magnetic surfaces for both closed and open field lines (See Fig 4.3) and F(ψ) is the poloidal current function [See Fig 4.1], which provides the components of magnetic field in cylindrical coordinates as [43],

$$B_R = -\frac{1}{R}\frac{\partial\psi}{\partial Z}, \quad B_Z = \frac{1}{R}\frac{\partial\psi}{\partial R}, \quad B_\zeta = -\frac{F(\psi)}{R}$$

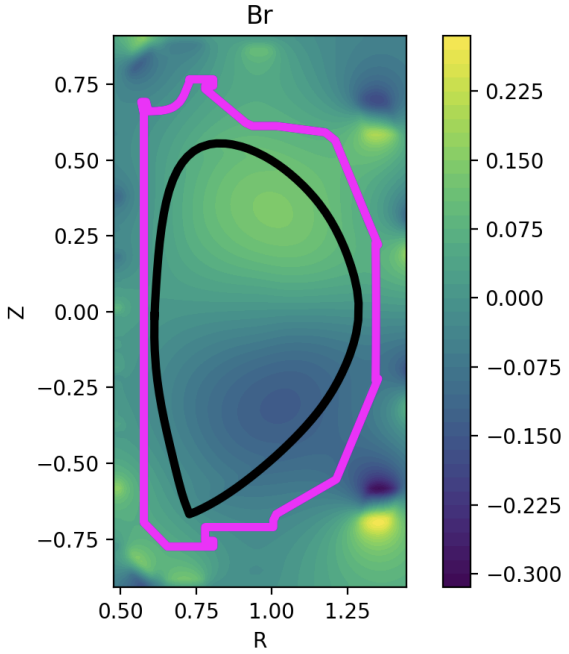
We now plot the B_Z and B_R with the help of above equations. In the partial differential part of above equations we use the gradiant function where it returns the gradiant of ψ in R and Z separately(ie, $\frac{\partial\psi}{\partial R}$ in one array and $\frac{\partial\psi}{\partial Z}$ in other). The plot of B_R and B_Z in RxZ grid plain in coloured contour form is shown in Fig 4.4.

4.3.2 B_ζ

From the equation we can plot the contour graph of B_ζ, but since we are concerned only up to the limiter points we need to scale down the F(ψ). First we know the that the ψ value is available only in the matrix form(129x129 grids), so we need some kind of interpolation on ψ such that the ψ value for any arbitrary (R,Z) is available. Here we use Rectangular Bivariant spline method, which interpolates the ψ in the whole RxZ domain. Now the ψ in F(ψ) should range from smallest value of ψ in whole domain of RxZ to largest value of ψ in limiter points. After finding the largest value of ψ in limiter points, we divide the range of ψ into 129 parts(new N_ψ) where 129 is the number of values in N_ψ. From the analysis of ψ we know that its value increases as we go from magnetic axis to outwards, the largest value of ψ inside the limiter points will be on the limiter points. We also need to interpolate the F(ψ) with new N_ψ, So we use Interpolate Univariant function(1D spline function) which gives us the value of F(ψ) for any value within the range of new N_ψ. For every ψ value inside limiter points we find the corresponding value of N_ψ(new) then find the F(ψ), which is used to find B_ζ, see Fig 4.5.



(a) B_Z



(b) B_R

Figure 4.4: Components of the magnetic field (a)B_Z and (b)B_R for DIII-D shot No. 158103 at 3050 ms. The magnitude of the magnetic field components is indicated by colours. The last closed flux surface and limiter points are represented by black and magneta lines, respectively.

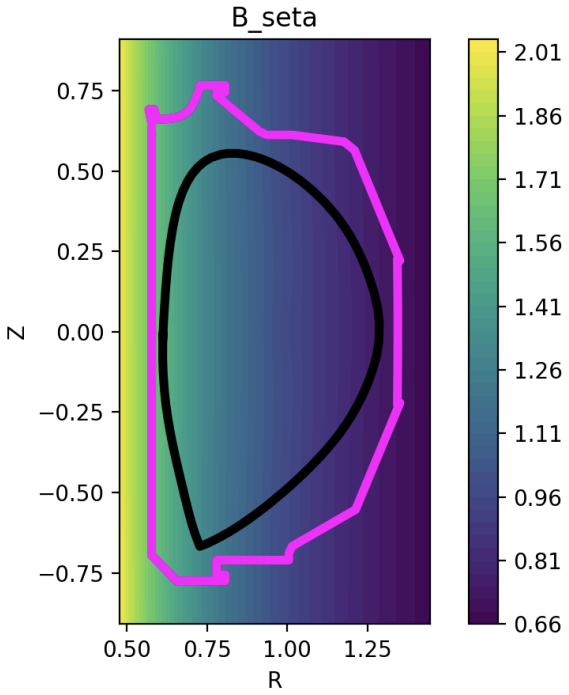


Figure 4.5: B_ζ for DIII-D shot No. 158103 at 3050 ms. The magnitude of the magnetic field components is indicated by colours. The last closed flux surface and limiter points are represented by black and magneta lines, respectively.

Chapter 5

Construction of Magnetic Flux Surfaces

5.1 Initial construction and need for error reduction

We know that particles in a fusion reactor travel along the magnetic field lines, so computational construction of field lines which passes through any desired point is crucial to simulations in fusion reactor. After the construction of field lines, we will be able to construct equi-spaced grids along the field lines. These equi-spaced grids later acts as probes which define the value of a particular physical quantity at that point and also used in solving the partial differential equation.

We trace out the magnetic field lines in poloidal direction with the help of following equations [43]:

$$\frac{dR}{dS} = b_R = \frac{B_R}{B_p}, \quad \frac{dZ}{dS} = b_Z = \frac{B_Z}{B_p}$$

where $B_p = \sqrt{B_R^2 + B_Z^2}$ is the magnitude of the poloidal magnetic field. 'S' is the increment along the field line, so by choosing S to be small(S tends to dS), we can calculate the increments in R and Z directions and ultimately field lines can be traced out. Since ψ is given as data points and not as a continous function, the components of magnetic field will also be data points. So we implement an interpolation scheme called "RecBivarianteSpline" class belonging to the "sci.interpolate" package (in python), which gives us the interpolated value of a particular magnetic field for a given point in 2D poloidal cross-section. This newly interpolated values of B_R and B_Z is used in tracing field lines. Field lines are traced by substituting above equations in Euler update equation which gives:

$$R_{i+1} = R_i \pm (dS_R F_R) \quad Z_{i+1} = Z_i \pm (dS_Z F_Z)$$

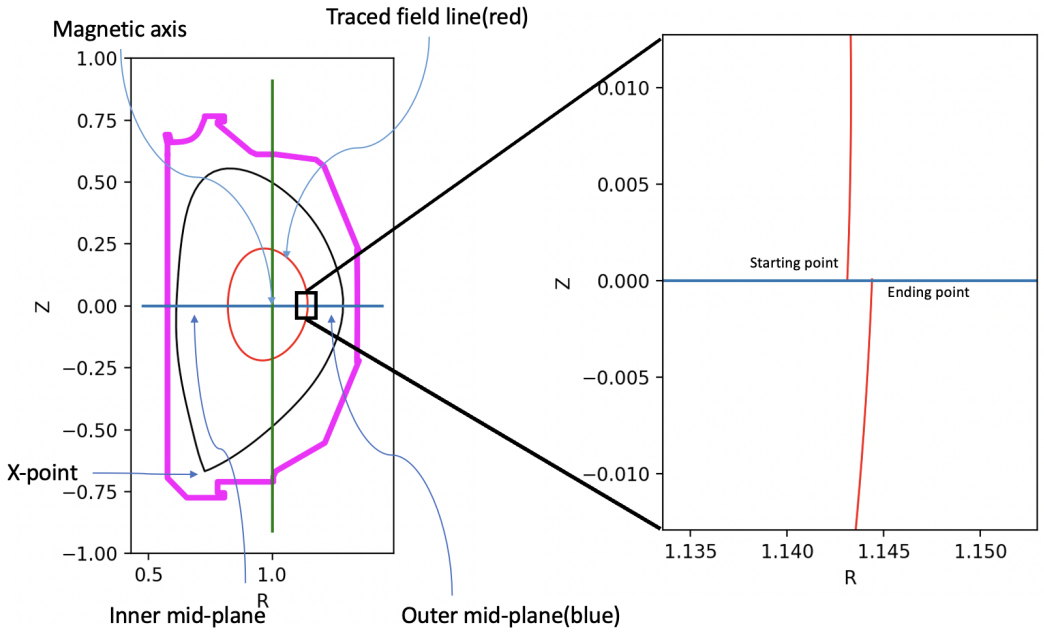


Figure 5.1: First figure shows traced field line indicated as red, along with outer mid-plane indicated as blue. The intersecting point of blue and green line is the magnetic axis. The adjacent figure is the zoomed part in the box, where we can clearly see the starting and ending points do not meet

$$F_R = b_R, \quad F_Z = b_Z$$

Where dS_R and dS_Z are small increments in R and Z directions of field lines respectively. "i" is the i^{th} iteration/increment. $\pm$ sign is for direction of motion. eg: "+" for anti-clockwise tracing of field lines and "-" for clockwise tracing of field lines. We start the tracing from outer mid-plane(Z=0 line of poloidal cross-section) and at a particular distance from magnetic axis. the Result obtained is shown in Fig 5.1.

From Fig 5.1, we can see that the starting point and ending point do not meet, which indicates there is a discontinuity for this closed flux surface(or field line) at that region and this is not correct. The reason for this is error carried by $(dS_R F_R)$ and $(dS_Z F_Z)$ terms in the increments (Euler update) in R and Z directions respectively. So longer we trace the flux surface, larger will be the error carried. This creates big problems as we move the starting point closer to separatrix(black line in Fig 5.1), where it takes longer interations to reach outer mid-plane and creates huge deviations in starting and ending points(or huge discontinuity in outer mid-plane). So we need to find a way to reduce the error in each iterations.

5.2 Error reduction

5.2.1 Traditional error reduction and constructing flux surfaces inside core region

Since we saw that error carried by Euler update can cause huge problem, we seek to tackle it first by traditional method. The main method by which error can be reduced is by reducing the error carrying term. Here the error carrying terms are $(dS_R F_R)$ and $(dS_Z F_Z)$. Since F_R and F_Z are constant for a particular point, the only thing that can be reduced is dS_R and dS_Z,ie reduce the step-size. To construct equi-spaced flux surfaces(field lines), we need to place the starting points equi-spaced in the outer mid-plane. we build equispaced radial grids (spacing of size Δr) on the outer midplane for each field line, which can be calculated as [43]:

$$\Delta r = \frac{1}{N_{core}}\{R_{LF}(\psi = \psi_{sep}, Z = 0) - R_0\}$$

Where R_{LF} is the radial position as a function of poloidal flux and Z on the inner mid-plane side and N_{core} is the number of field lines we need in the core region. ψ_{sep} is the value of the poloidal flux on the separatrix (last closed flux surface). R_0 is the radial position of magnetic axis. The starting points are determined by:

$$R = R_0 + i\Delta r \quad (where \quad i = 1, 2, 3, ..., N_{core} - 1), \quad Z = 0$$

These starting points are situated in the outer mid-plane. The field lines traced exactly as above with step-size reduced by few orders. We use the below condition to stop the tracing after one full rotation and we get the field lines as given in Fig 5.2.

$$Z_i < 0 \quad and \quad Z_{i+1} \geq 0$$

5.2.2 Problems in X-point

The discontinuity of flux surfaces in outer mid-plane in inner mid-plane side is not the only problems caused by error accumulation. The scattering of traced flux surfaces near X-point is also a result of error accumulation. This is because the change in ψ near X-point is so small that a small error in traced flux surface will scatter the next iterations in any of the three different directions depending on the direction of the tracing(if you start from X-point, you can go four different directions). In the Fig 5.3 we can see the scattering

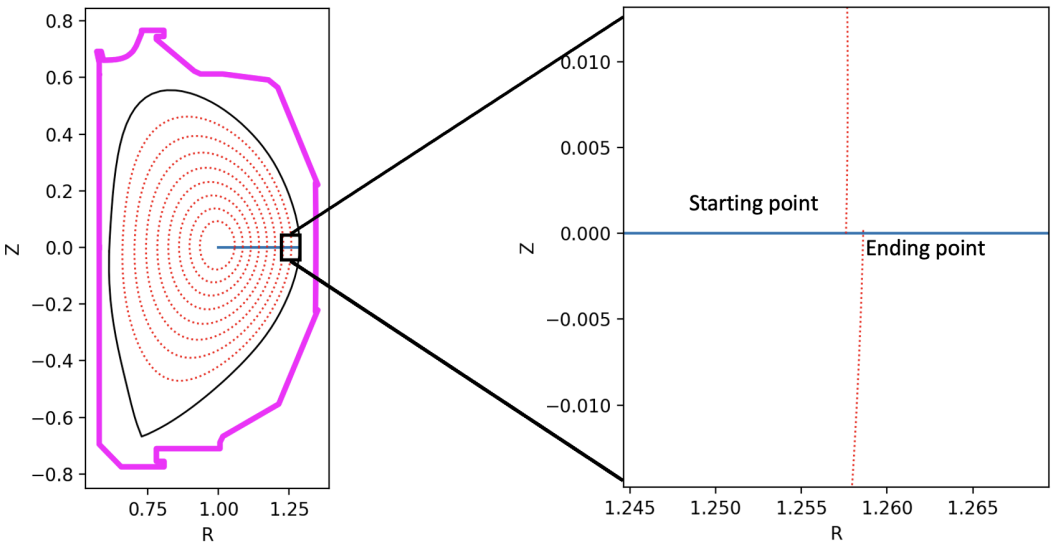
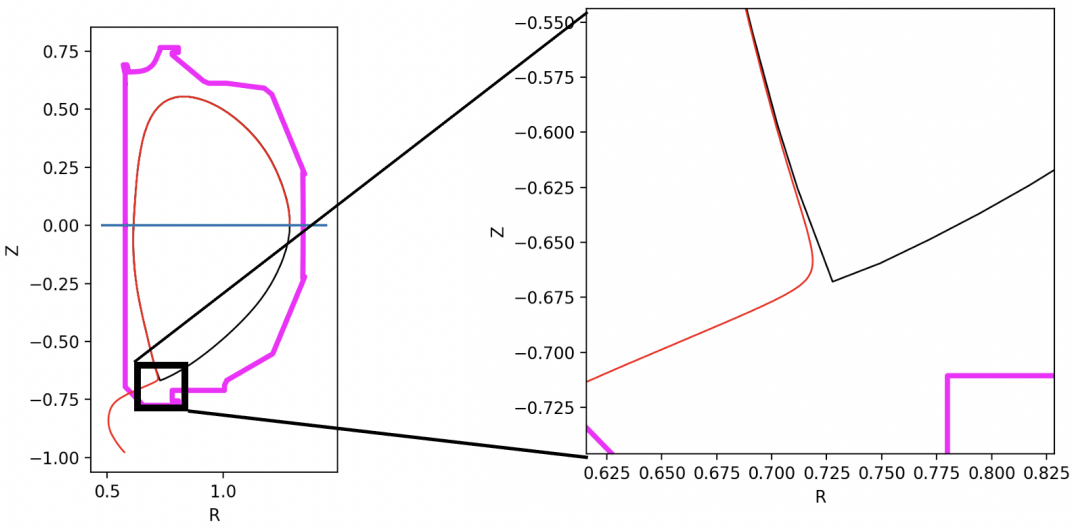


Figure 5.2: Left one is the traced flux surfaces, where starting points are on outer mid-plane which is shown in blue colour(it extends from$(R_0, 0)$ to $(R_{LF}(\psi = \psi_{sep}, Z = 0), 0))$ and flux surfaces are traced anti-clockwise. The right one is the zoomed part of last flux surface inside separatrix, where we can see the deviation in starting and ending points is still there, but less than previous

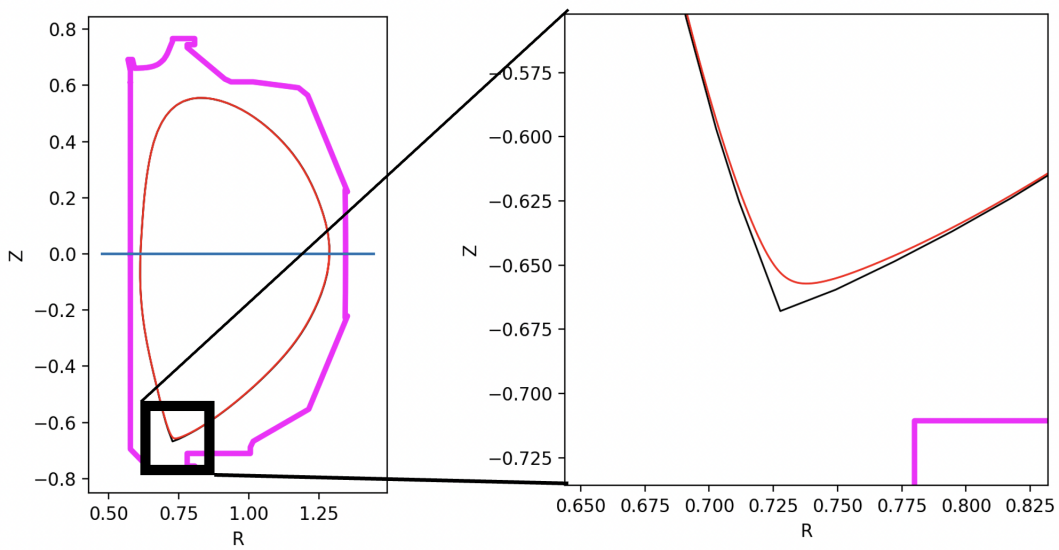
of flux surfaces near X-point. From Fig 5.3, we can see that in anti-clockwise tracing the error pushes the flux surface outwards, so it picks up that direction upon reaching X-point. While in clockwise tracing the error pushes the flux surface inwards and it takes that direction as shown. We concludes that with our traditional error reduction none of the clockwise or anti-clockwise tracing can trace the X-point correctly. So we need additional error reduction methods.

5.2.3 Additional error reduction method

To overcome the problems in tracing X-point, we use an additional method to decrease the error in each step. Since the flux surfaces are characterized by having a constant value of ψ, we can say that at each step to construct a flux surface the initial and final points should have its corresponding ψ values to be equal. We can try to reduce the difference in ψ values(which in turn reduces the error in each step), as we construct the flux surfaces. From the ψ graph we can see that the ψ values inside a closed flux surface is less than that in the outside, ie, ψ values generally increase as we go from center to outwards. Suppose we are at a point in flux surface and want to find the next point(to create a flux surface). we will get a point from Euler update but the psi value of that point is not same as the previous point. So we construct a radial vector of length same as the step size and we rotate anti-clockwise until we cross the original flux surface(crossing original flux surface means when ψ value of previous point gets equal to the ψ value of



(a) Anti-Clockwise tracing



(b) Clockwise tracing

Figure 5.3: This is the result of tracing last closed flux surface in both directions. Here the black and red lines are the experimental and theoretically traced last closed flux surfaces.

next point). The angle subtended by a rotation is used in the next rotation where its half value is used and so on. This process is continued until the difference in ψ values comes under a prefered tolerance level.

Here in the error reducing function, we need to ensure that at each rotation of vector it should cross the original flux surface, so if we take an arbitrary angle initially and use that for rotation, there might be some instance in which it cannot cross original flux surface, (since we are also decreasing the angles by half for next rotation). So here we take a small angle (which is much lesser than the initial angle between vector and original flux surface) and keep on rotating using small angle until it crosses original flux surface. For next rotation we take half of that small angle and keep on rotating in opposite direction until it crosses original flux surface again and keep on doing that until the error is less than tolerance.

Here is the mathematical reason to choose small angle and repeated rotation in same direction method rather than big angle and single rotation.

If final angular position after 1 rotation= θ

Then final angular position after 2 rotation=$\theta - \frac{\theta}{2}$

Final angular position after 3 rotation= $\theta - \frac{\theta}{2} + \frac{\theta}{4}$

Final angular position after n rotation= $\theta[1 - 1/2 + 1/4 - 1/8.....ntimes]$

So if n tends to ∞, sum of [1 - 1/2 + 1/4 - 1/8.....n times] becomes 2/3 where $\frac{2\theta}{3}$ will be the approximate angular position of original flux surface, but this is highly unlikely to happen because, for this to work we need particular θ for every points so that at each rotation of vector it should cross the original flux surface(even when we are reducing θ of previous rotation by half for using it to next rotation), also that θ can be any value(for a point) and different values for different points(to meet the above condition) and we don't have any clue on that.

If we use this method and after some rotation it failed to cross the original flux surface, then the subsequent rotations will not cross the original flux surface and the oscillation of vector will converge to $\frac{2\theta}{3}$ which is not the original flux surface.

So crossing of original flux surface at each rotation is essential for the vector to converge to original flux surface.

The small θ we used for our case is $\frac{\pi}{620}$. The result of using this method is shown in Fig [**trace5**]. From the figure 4.4, we can tell that our problem of discontinuity in outer mid-plane is almost solved within the tolerance limit. We can see that the difference in starting and ending point is reduced greatly, So we are able to solve for flux surfaces in core region except last closed flux surface. The separatrix(last closed flux surface) is still unresolved(see zoomed part of X-point region in Fig 5.4).

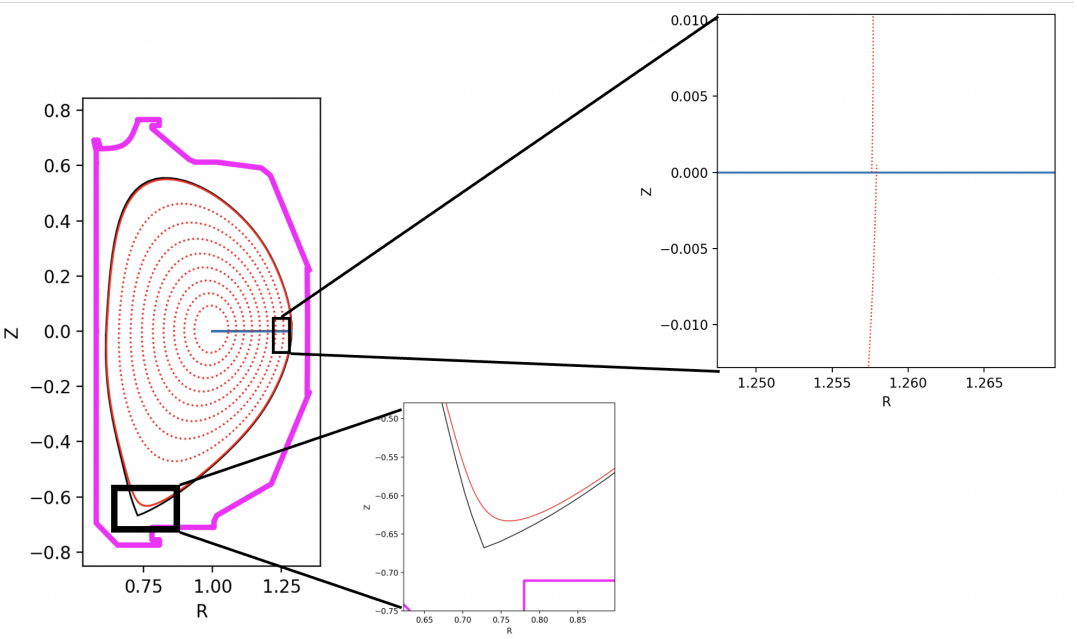


Figure 5.4: This is the result of construction of flux surfaces with both traditional and additional error reduction methods. It also shows the zoomed part of last flux surface within separatrix in outer mid-plane region and zoomed part of X-point region.

5.3 Handling X-point

We conclude that no matter how many error reduction methods we use(there is also a risk involved in using lot of error reduction methods in one step, which increases the computational complexity) there will be a small error while we reach the X-point. So in this section we will solve the problems associated with X-point within the tolerance limit.

5.3.1 Tracing by parts

We know that in the Euler update the error made in one step is always carried to the next step, so the error accumilated is proportional to the distance traced along flux surface(or the number of incrementes covered) in one run. As we go towards the separatrix the length of each flux surface increases. So if we trace the last closed flux surface by parts, that means divide the whole flux surface into different segments and trace each of them independently, instead of tracing the whole in one run. This reduces the error in tracing the whole surface as we have separate starting points for each segments and this reduces the average distance traced per run.

5.3.2 Starting from X-point

We know that in order to trace the exact sharp tip of X-point, there must be almost zero error accumulated while reaching the X-point. But even after tracing the flux surface by parts as mentioned above, it is still almost impossible to get the zero error. So the way we choose to get the sharp is by starting from X-point, where there will be no error in starting point.

But when we start from the X-point, there are four possible directions in which you can trace(see Fig 5.5 (a)). Since all these four directions belong to same flux surface, we cannot distinguish between them other than their direction.So we need a method to pick up direction as we wish. Actually this whole last closed flux surface is a single flux surface where it has an intersection with itself (at the X-point, see Fig 5.5 (a)). So two of the four directions(two opposite directions) can be picked from the clockwise and anti-clockwise rotation equations respectively. This is done by the interchange of $\pm$ in Euler equations. But we cannot pick up the other two directions by Euler equation, So here we do a small trick with in our tolerance range.

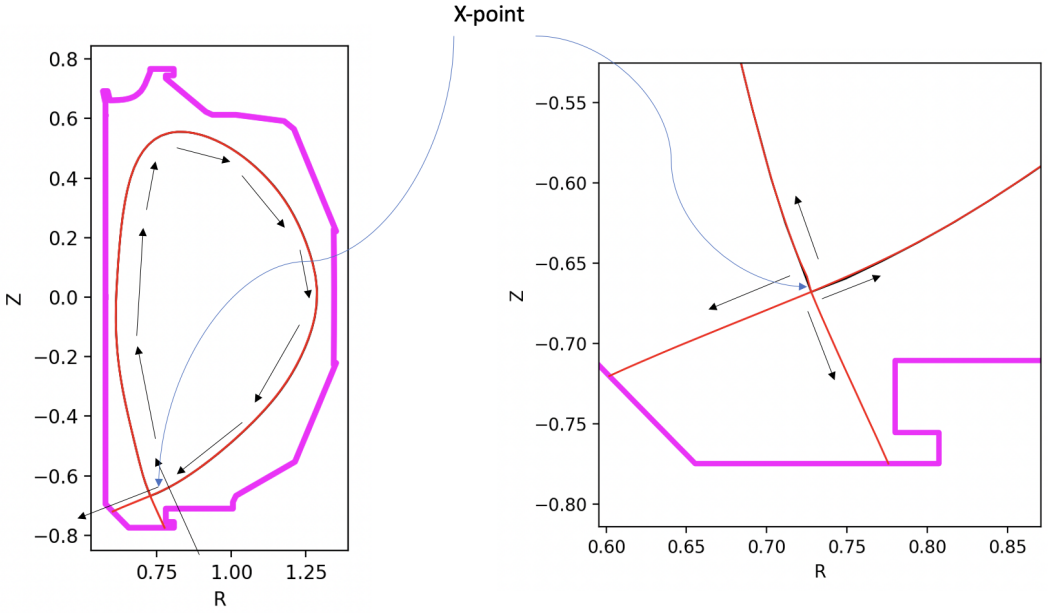
We know the remaining directions are almost perpendicular to other two. We give a small vector in these directions(in the remaining opposite directions), which acts like a small kind of push. Then we implement the Euler equations with all the error reduction schemes that we discussed before to the resultant after the vector push. While implementing the Euler equations we should use both clockwise and anti-clockwise rotation equations to get the flux surfaces in both direction. Here what happens is that after our small push, the Euler equations and our all error reduction method will make the traced line drift towards the original flux surface. But this push and the correction by drifting should be small such that it falls under our tolerance limit.

In Fig 5.5 (b) we can see the resultant separatrix traced by all the above mentioned methods. We can also see the different segments used to trace in different colours. In the zoomed part of X-point, we can see that the red and yellow flux surfaces are traced by clockwise and anti-clockwise tracing equations. The other two opposite directions(indicated by brown and blue colours) are picked up an initial push and its drift towards original by Euler equations and all the error reduction methods we discussed.

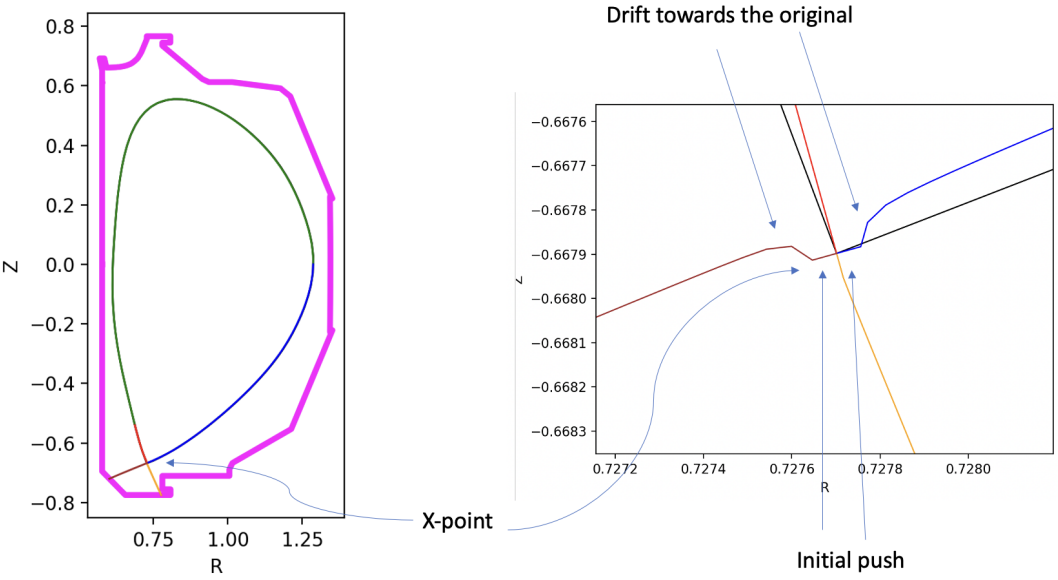
5.4 Flux surfaces beyond X-point

5.4.1 Initial construction

Till this point we were only concerned with the flux surfaces in the core region. Now we extend the ideas developed for core region to other regions. In core region all the flux



(a) Directions in X-point



(b) Final solution for our X-point

Figure 5.5: First figure in figure (a) shows the direction of field lines in last closed flux surface and the second one(zoomed part of X-point) shows the possible directions a tracing can go if you start from X-point. First figure in figure (b) is the final result for X-point, where the different colours are the different segments we used in tracing and second figure shows how we traced all four directions.

surfaces were closed, so if you have a starting point(anywhere in the core region) you could trace the whole flux surface associated with that point. But in other regions inside limiter points(SOL and private region) flux surfaces are not closed anymore. So if we want to construct the whole flux surface inside the limiter points (points which define the wall or boundary in the cross-section) which is associated with a particular point, we need to use anti-clockwise equation and clockwise equation simultaneously such that tracing is done in both directions from that point. This tracing is done till the flux surface constructed hits the limiter surface.

Similarly as we did in the core regions, we need starting points. So we need to place the starting points equi-spaced in the outer mid-plane. we build equispaced radial grids (spacing of size Δr) on the outer midplane for each field line, which can be calculated as:

$$\Delta r = \frac{1}{N_{SOL}}\{R_{LF}(\psi = \psi_{lim}, Z = 0) - R_{LF}(\psi = \psi_{sep}, Z = 0)\}$$

Where R_{LF} is the radial position as a function of poloidal flux and Z on the inner mid-plane side and N_{SOL} is the number of field lines we need in the SOL region. ψ_{sep} is the value of the poloidal flux on the separatrix (last closed flux surface) and ψ_{lim} is the value of the poloidal flux on the limiter points (Plasma facing wall) The starting points are determined by:

$$R = R_{LF}(\psi = \psi_{sep}, Z = 0) + i\Delta r \quad (where \quad i = 1, 2, 3, ..., N_{SOL}), \quad Z = 0$$

These starting points are situated in the outer mid-plane. The field lines are traced exactly as we did before but use both clockwise and anti-clockwise equations simultaneously as discussed above. A particular tracing of the flux surface is stopped upon crossing the limiter surface. To stop the tracing of flux surface upon hitting limiter points, we check that weather the line segment created by a particular tracing iteration intersects with any of the line segment in set of all line segments which make the limiter points, and this is done for every tracing iterations in both clockwise and anti-clockwise tracing. If we find any intersection, tracing is stopped in that direction for the flux surface. The above mentioned intersection in present if $0.0 \leq$ t ≤ 1.0 and $0.0 \leq$ u ≤ 1.0, where t and u are given by,

$$t = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_1 - y_3 \end{vmatrix}} = \frac{(x_1 - x_3)(y_3 - y_4) - (x_3 - x_4)(y_1 - y_3)}{(x_1 - x_2)(y_3 - y_4) - (x_3 - x_4)(y_1 - y_2)}$$

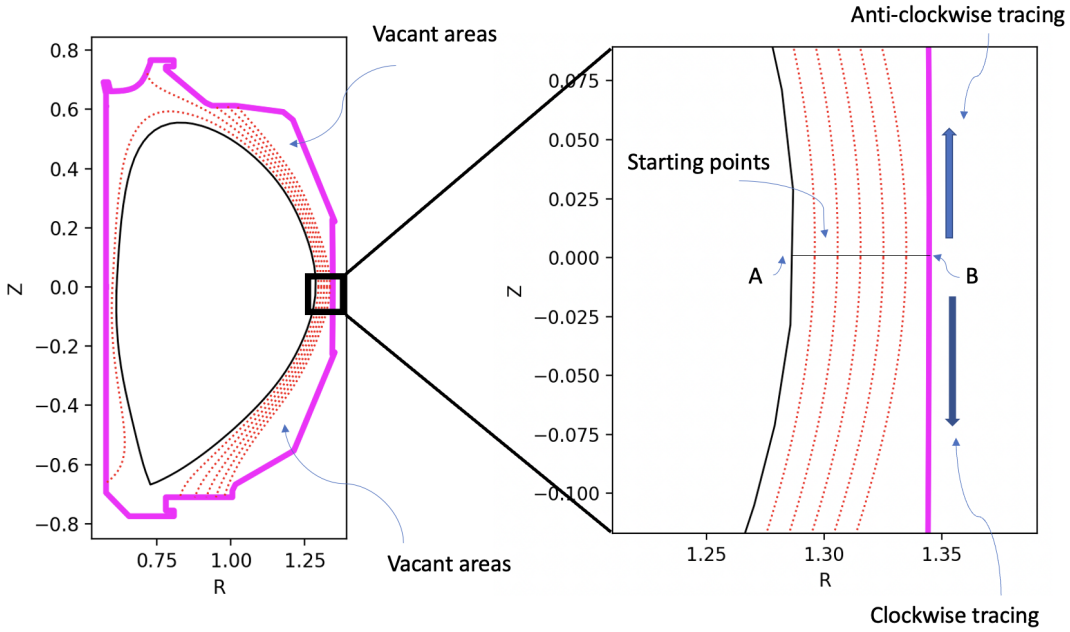


Figure 5.6: The left figure shows the resultant of flux surfaces made in section 5.4.1. In the right figure we show the zoomed part which contains the starting points and directions of tracing. A and B indicate $R_{LF}(\psi = \psi_{sep}, Z = 0)$ and $R_{LF}(\psi = \psi_{lim}, Z = 0)$

$$u = \frac{\begin{vmatrix} x_1 - x_3 & x_1 - x_2 \\ y_1 - y_3 & y_1 - y_2 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}} = \frac{(x_1 - x_3)(y_1 - y_2) - (x_1 - x_2)(y_1 - y_3)}{(x_1 - x_2)(y_3 - y_4) - (x_3 - x_4)(y_1 - y_2)}$$

Where (x_1, y_1) and (x_2, y_2) are the end points of line segment created by the increment in tracing. (x_1, y_1) and (x_2, y_2) represents the end points of a line segment in set of all line segments which make the limiter points (set of end points of all line segments in limiter surface).

The resultant figure is given by Fig 5.6. From the Fig 5.6, we can see that there are some areas which remain vacant(without the flux surfaces). This results in lack of grid points in these regions(as grid points are made on flux surfaces), which leads to bigger triangles in these regions(as triangles are constructed by joining grid points), which causes huge errors in our field estimations using finite element methods. So we need to figure out a way to deal with vacant areas.

5.4.2 Dealing with vacant areas

We saw in the previous section that, if you choose the starting line within the limiter point, it doesn't matter how many flux surfaces we choose to make, there will always be some vacant areas. So one way to reduce the vacant areas is by taking the starting line

beyond limiter points.

SOL region

To reduce the vacant areas in SOL region, we put the starting line in both inner and outer mid-plane regions. The Δr and starting point R equation now changes to:

$$\Delta r = \frac{1}{N_{SOL}}\{R_{LF}(\psi = \psi_{out}, Z = 0) - R_{LF}(\psi = \psi_{sep}, Z = 0)\}$$

$$R = R_{LF}(\psi = \psi_{sep}, Z = 0) + i\Delta r \quad (where \quad i = 1, 2, 3, ..., N_{SOL}), \quad Z = 0$$

Here ψ_{out} is some value of ψ outside limiter points. The resultant flux surfaces is shown in Fig 5.7.

Private region

We know that none of the flux surfaces in SOL region passes through private region, So we won't get flux surfaces in private regions by the two starting lines mentioned above

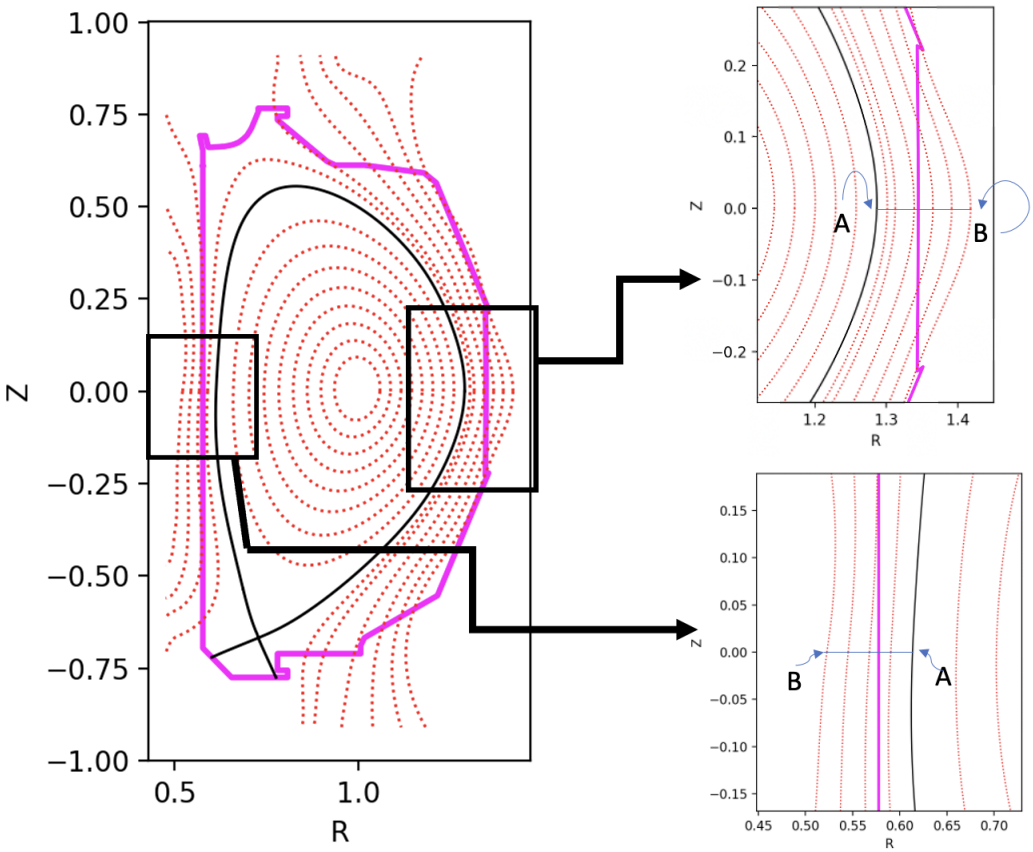


Figure 5.7: This figure shows the flux surfaces constructed by extending the starting line beyond limiter points. The two zoomed parts consist of two starting lines we used for SOL region. The A and B represent $R_{LF}(\psi = \psi_{sep}, Z = 0)$ and $R_{LF}(\psi = \psi_{out}, Z = 0)$

section. So we need additional starting line for private region. We take the starting line for private region to be fully out side the limiter points. The Δr and starting point R equation now changes to:

$$\Delta r = \frac{1}{N_{private}} |\vec{R}_{private}(\psi = \psi^2_{out}, Z = z_2) - \vec{R}_{private}(\psi = \psi^1_{out}, Z = z_1)|$$

$$\vec{R} = \vec{R}_{private}(\psi = \psi^1_{out}, Z = z_1) + i\Delta r \cdot \hat{i} \quad (where \quad i = 1, 2, 3, ..., N_{private})$$

Here $\vec{R}_{private}(\psi = \psi^2_{out}, Z = z_2)$, $\vec{R}_{private}(\psi = \psi^1_{out}, Z = z_1)$ and $\vec{R}$ are the radial vector in poloidal 2D plane which represents a point outside limiter points and near private regions. $\hat{i}$ is in the direction of $(\vec{R}_{private}(\psi = \psi^2_{out}, Z = z_2) - \vec{R}_{private}(\psi = \psi^1_{out}, Z = z_1))$. $N_{private}$ is the number of flux surfaces you need in limiter points. But since the private region lies near the X-point, the ψ values are almost same, so it is difficult to get more number of flux surfaces in private regions. So to get best results keep the $N_{private}$ low and starting line close to limiter points all possible. The starting and ending points of the starting line for private region is given by $\vec{R}_{private}(\psi = \psi^1_{out}, Z = z_1)$ and $\vec{R}_{private}(\psi = \psi^2_{out}, Z = z_2)$ respectively. The resultant flux surfaces is shown in Fig 5.8.

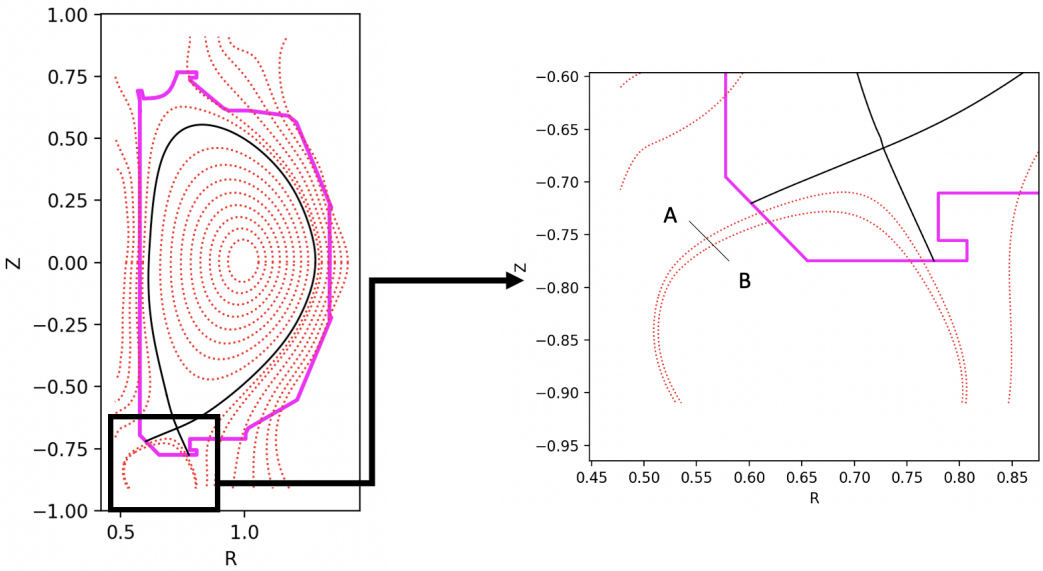


Figure 5.8: This figure represents the zoomed part of private regions where the starting line is shown. A and B represents the end points, which is $\vec{R}_{private}(\psi = \psi^1_{out}, Z = z_1)$ and $\vec{R}_{private}(\psi = \psi^2_{out}, Z = z_2)$ respectively.

5.4.3 Selecting field lines inside the limiter points and final output

Now we are able to generate flux surfaces in all the regions. But since we are interested in what happens inside the limiter points, we need to select the flux inside the limiter points. For this to happen we check weather each increment is inside or outside the limiter points. The decision of inside and outside is determined by counting the number of times the traced flux surface intersects with the limiter points. So while tracing the flux surface the number of times it crossed limiter point is also remembered and the part of flux surface which have this number to be an odd number is indicated as it is inside limiter points and all the part of flux surface while the number is even is indicated as outside. Here while implementing this idea we keep the starting number to be one for all flux surfaces started tracing from inside the limiter points and the number to be zero for all flux surfaces started tracing from outside the limiter points.

The final output of the tracing is shown in Fig 5.9.

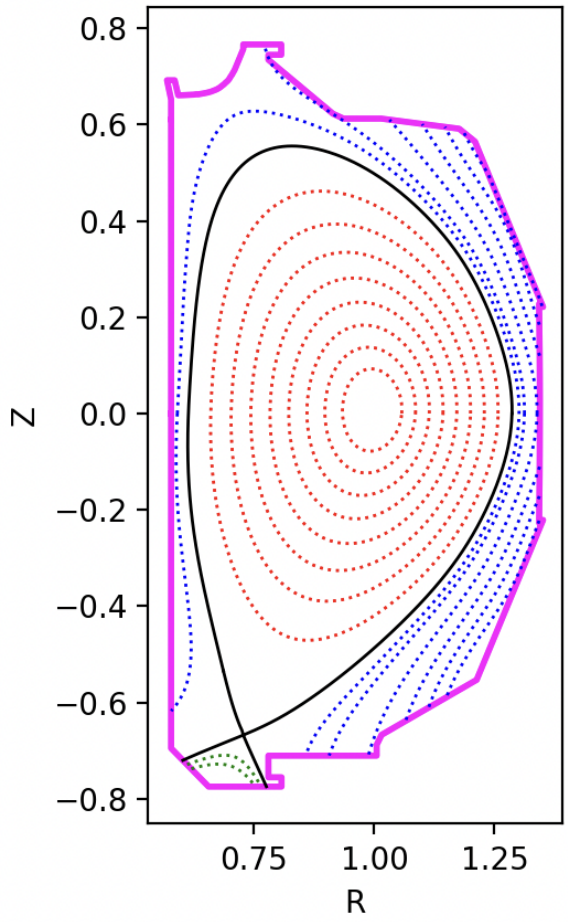


Figure 5.9: This is the full flux surfaces(inside limiter points) in 2D poloidal cross section. Flux surfaces in core, SOL and private region are given as red, blue and green respectively.

Chapter 6

Construction of field aligned grid points

The grid points are constructed along the flux surfaces. These grid points act as probes which represent any physical quantity averaged over the neighbouring area. These also help in discretization of 2D poloidal cross section, where we will be able to the solve equations concerning the physical quantities.

6.1 Core region

Grid points are constructed on the flux surfaces, where each grid point is placed after a particular length(s, which is the distance between two grids) is traced. It is exactly like tracing a flux surface where we put a grid point after every 's' distance. The starting points in tracing the flux surfaces will get the first grid point. But since the flux surfaces in the core region are closed, we cannot use any value as 's'. While constructing grids, we should keep in mind that the distance between grids or 's' should be almost same everywhere and no two adjacent grids should be very close to each other or very far to each other in compared to other grid points. 's' is given by:

$$s = \texttt{integer}\left[\frac{\text{length of flux surfaces}}{\text{No: of grid points}}\right]$$

There might be a chance that for a particular 's', the grid points near the outer mid-plane(The first and last grid point in a flux surface) will be very much closer or very much farther. So in that case we should eliminate the last grid point(for closer case) or add one more grid point(for farther case). So to implement this we define to quantities

$S_{original}$ and $S_{estimate}$ which are given by:

$$S_{original} = \texttt{integer}\left[\frac{\text{length of flux surfaces}}{\text{No: of grid points}}\right]$$

$$S_{estimate} = \frac{\text{length of flux surfaces}}{\text{No: of grid points}}$$

$$error = (S_{estimate} - S_{original}) \times 100$$

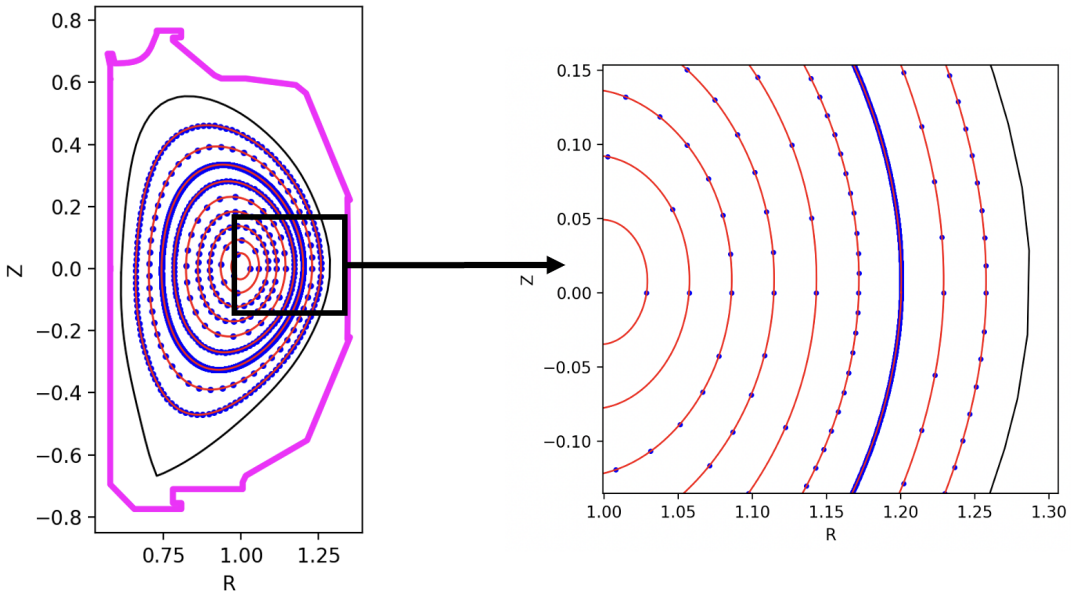
First we find the error, then if it is less than 50% then we increase $S_{estimate}$(by decreasing No: of grid points) until error becomes less than tolerance level. If the error is more than 50% then we reduce $S_{original}$(by increasing No: of grid points) until error becomes less than tolerance level. When the error is less than tolerance, we take $S_{estimate}$ as 's'. This way we can avoid the problem of grids aggregating or dispersing. But the main issue comes in deciding the tolerance level. If the tolerance level is too high, we won't get any results from this technique. If the tolerance level is too small we get vastly different 's' for each flux surfaces(see Fig 6.1 (a)). So by trial and error methods we find that tolerance of 15% gives good result(Almost same 's'). This is confirmed by calculating the variance of grid points, were tolerance level of 1% had a variance of 0.00057164388075594 and tolerance level of 15% had a variance of $1.0157628455804806 \times 10^{-5}$, almost reduced 50 times. The Fig 6.1 (b) is the final result in constructing grid points in core region.

6.2 Last closed flux surface

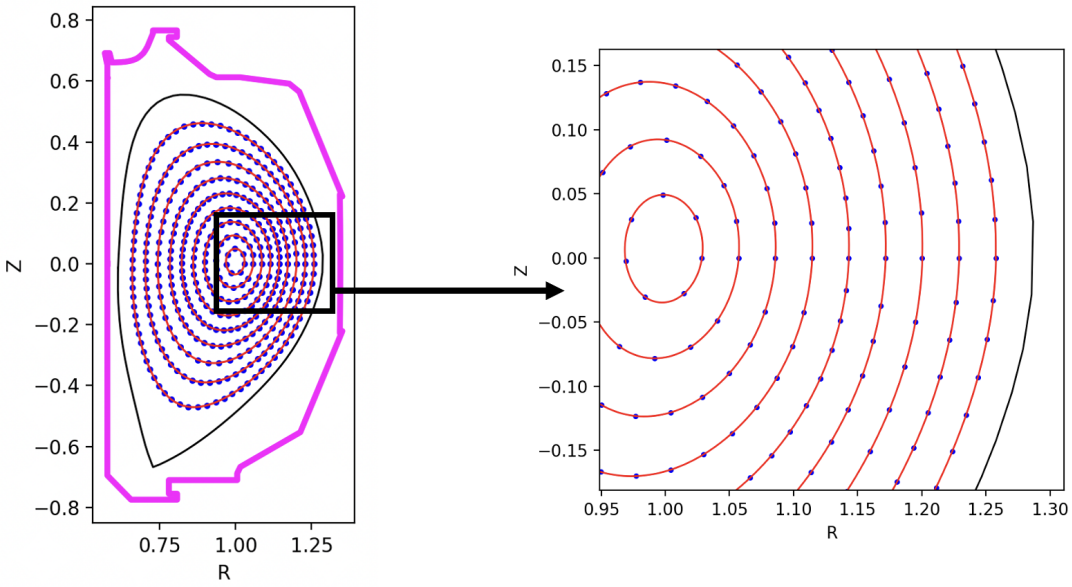
The last closed flux surface consist of a closed loop around core region and two arms which make the private region. In the closed loop part we construct the grids same as we did in the core region. For the two arms part we don't need to find an optimum 's' as it is not closed. So it is sufficient to take some 's' comparable to the 's' we used in core region and use it to construct grids in the two arms. But here one should also put a grid point when the flux surface intersects limiter points. Here as same as flux surface is created, grid construction is also started from X-point.

6.3 SOL and Private region

We know that grid construction is done in the same manner as the flux surface is traced, where each grid point is placed after 's' distance is travelled. So for SOL and private region, we go in the same manner as we did in flux surface tracing (starting the grid construction from respective staring points). Similar to the selecting of flux surfaces, we only construct grids in flux surface when it is inside limiter points. We also put grids



(a) When tolerance level is 1%



(b) When tolerance level is 15%

Figure 6.1: Above is result of grid construction in core region with two different tolerances. the red lines are flux surfaces and blue dots are field aligned grids

when flux surfaces intersect limiter points, this is because limiter points are unevenly distributed ie, closely spaced in curves and no points in straight lines (only at end points), so we get grids in parts where there is no limiter points. This is extremely useful in the construction of mesh. The final result by combining all mesh is shown Fig 6.2

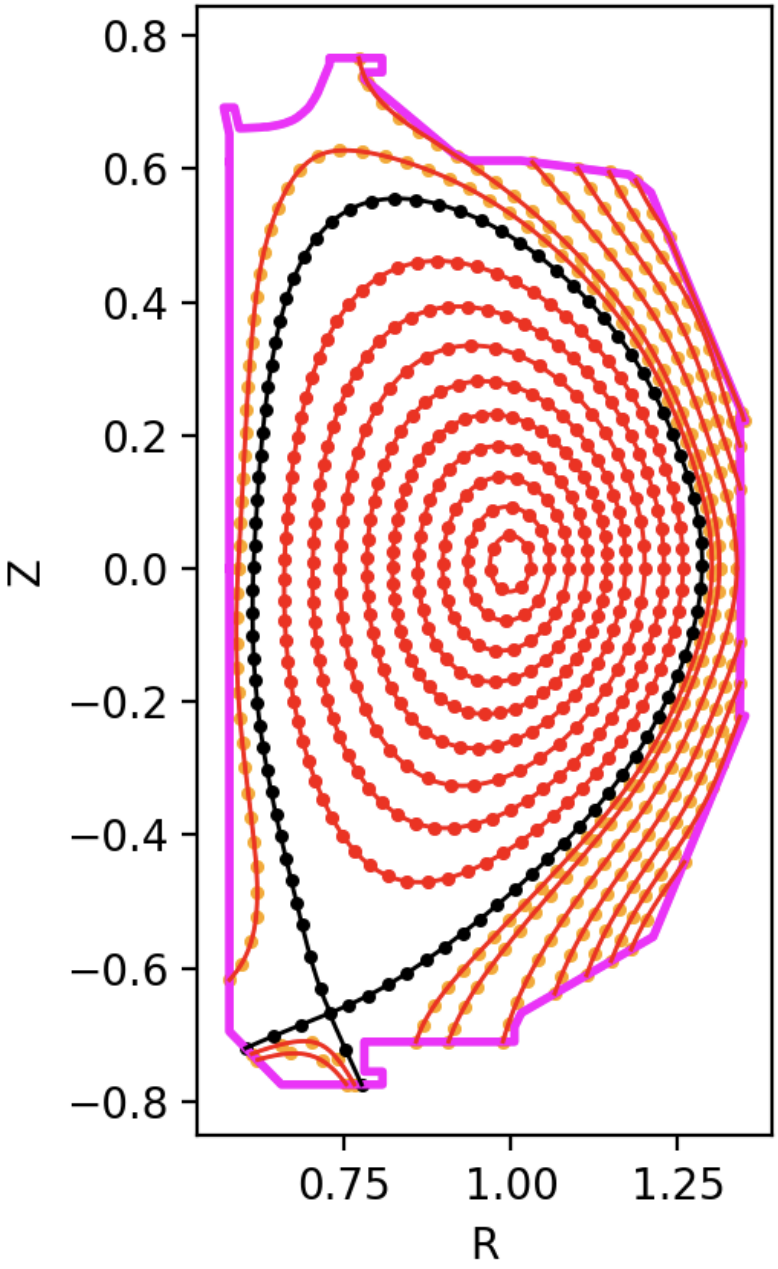


Figure 6.2: Final grids on a poloidal plane coupling core and SOL. The field aligned mesh at the core, separatrix, SOL, and private regions are represented in red, black, yellow (SOL, and private) respectively

Part III

Mesh Generation and Optimization

Chapter 7

Partially-structured mesh generation

7.1 Exporting Data points

As described in the introduction, we use TRIANGLE software to construct triangles using the generated grid [42]. We need to export the data points to readable files for Triangle to use the software. These readable files are all text files with suitable extensions. The type of extension in which we export the data determines how Triangle should interpret the data. There are six file formats(six extensions) that act as input and output files for Triangle. They are:

1. .node files
2. .ele files
3. .poly files
4. .area files
5. .edge files
6. .neigh files

Here we will describe the first three file formats which we used.

7.1.1 .node files

The syntax for .node files is;

1. First line: < No: of vertices> <dimension (must be 2)> < No: of attributes> <No: of boundary markers (0 or 1)>

```
# A set of fifteen points in 2D, no attributes, no boundary markers.
15  2  0  0
# And here are the fifteen points.
 1      0       0
 2     -0.416   0.909
 3     -1.35    0.436
 4     -1.64   -0.549
 5     -1.31   -1.51
 6     -0.532  -2.17
 7      0.454  -2.41
 8      1.45   -2.21
 9      2.29   -1.66
10      2.88   -0.838
11      3.16    0.131
12      3.12    1.14
13      2.77    2.08
14      2.16    2.89
15      1.36    3.49
```

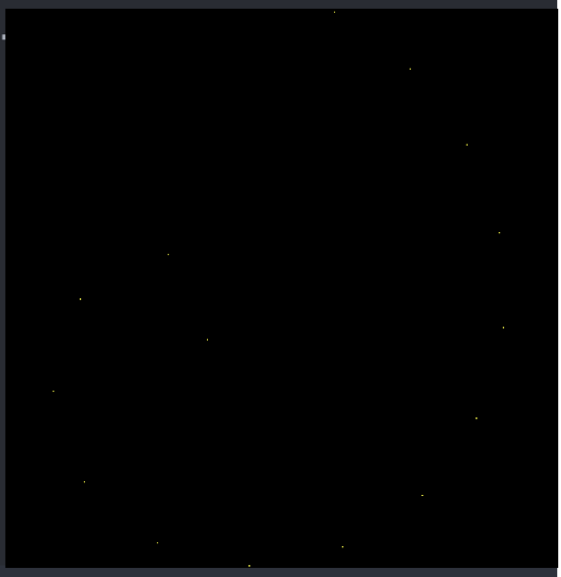


Figure 7.1: A sample .node file and its corresponding plot.

2. Remaining lines: <vertex No:> <x> <y> [attributes] [boundary marker]

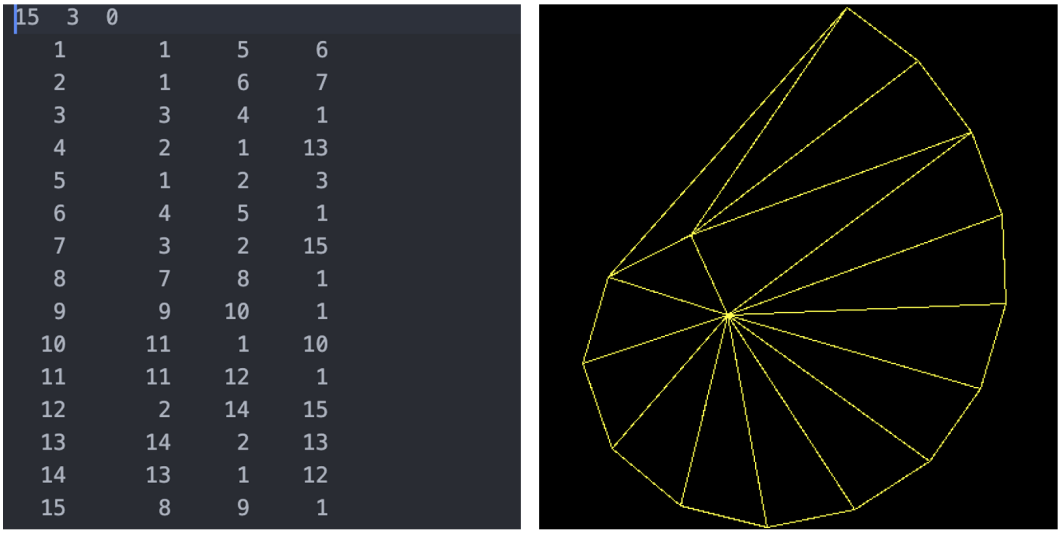
Vertices(data points) must be numbered consecutively, starting from one or zero while writing the file. The attributes, typically floating-point values of physical quantities (like mass or conductivity) associated with the nodes of a finite element mesh, are taken unchanged to the output mesh. If there is '1' in the fourth entry of the first line, the last number of the remainder of (each line) the file is assumed to contain boundary markers. Boundary markers identify boundary vertices and vertices resting on PSLG(Planar Straight Line Graph) segments. The .node files produced by Triangle contain boundary markers in the last column, (See Fig. 7.1 for an example).

7.1.2 .ele files

The syntax for .ele files is;

1. First line: <No: of triangles> <nodes per triangle> <No: of attributes>
2. Remaining lines: <triangle No:> <node> <node> <node> ... [attributes]

Numbering of the triangles must be done consecutively, starting from 1 or 0. The three nodes are listed in counterclockwise order around each triangle. (The remaining nodes, if any, depending on the type of finite element used). As in .node files, the attributes are typically floating-point values of physical quantities (such as mass or conductivity)



```
15  3  0
   1       1     5     6
   2       1     6     7
   3       3     4     1
   4       2     1    13
   5       1     2     3
   6       4     5     1
   7       3     2    15
   8       7     8     1
   9       9    10     1
  10      11     1    10
  11      11    12     1
  12       2    14    15
  13      14     2    13
  14      13     1    12
  15       8     9     1
```

Figure 7.2: A sample .ele file and its corresponding plot.

associated with a finite element mesh's elements (triangles), (See Fig. 7.2 for an example).

7.1.3 .poly files

The syntax for .poly files is;

1. First line: <No: of vertices> <dimension (must be 2)> <No: of attributes> <No: of boundary markers (0 or 1)>
2. Following lines: <vertex No:> <x> <y> [attributes] [boundary marker]
3. One line: <No: of segments> <No: of boundary markers (0 or 1)>
4. Following lines: <segment No:> <endpoint> <endpoint> [boundary marker]
5. One line: <No: of holes>
6. Following lines: <hole No:> <x> <y>
7. Optional line: <No: of regional attributes and/or area constraints>
8. Optional following lines: <region No:> <x> <y> <attribute> <maximum area>

A .poly file is a PSLG, plus some additional information. PSLG stands for Planar Straight Line Graph. By definition, a PSLG is a list of vertices and segments. Holes, concavities, regional attributes, and constraints on the areas of triangles may also include in a .poly file. The first section lists are identical to the format of .node files. Vertices

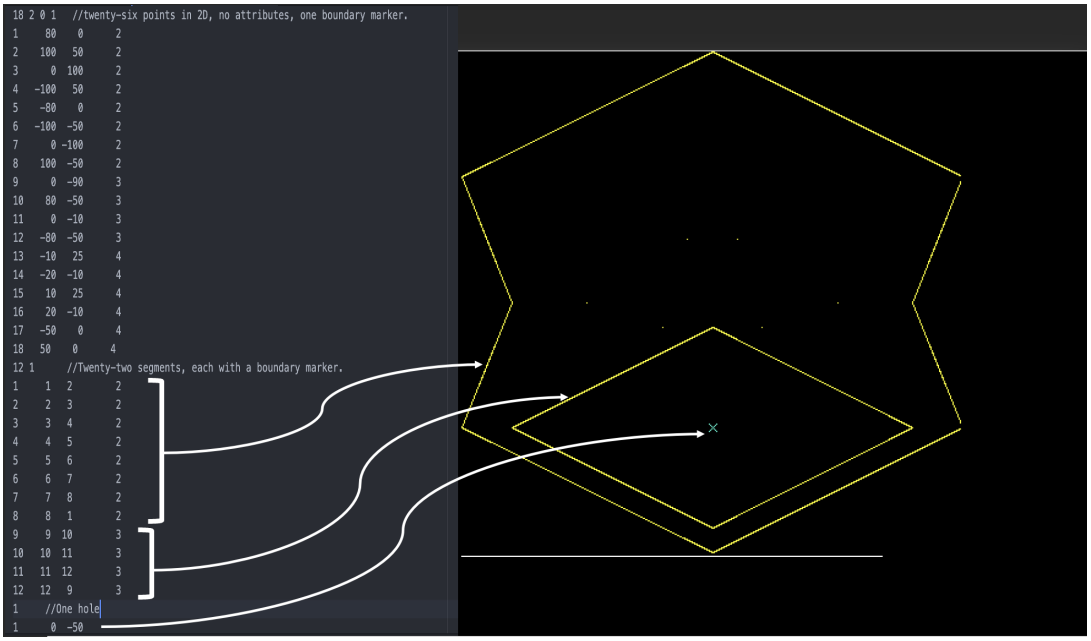


Figure 7.3: A sample .poly file and its corresponding plot.

can be listed in a separate .node file and <No: of vertices> may be set to zero to indicate that; .poly files produced by Triangle always have this type of format. A set of vertex represented in this manner has the upper hand that it can be easily triangulated with or without segments (depending on what is read, .poly or .node). The second section lists the segments in the mesh. Segments are edges in the triangulation that is enforced (although each segment may be subdivided into smaller edges). In a .poly file each segment is specified by giving the indices of its two endpoints; that means that you must include its endpoints in the vertex list. Each segment, like each vertex, may have a boundary marker. The third section lists holes in the triangulation. Holes are specified by identifying a point inside each hole. Triangle creates holes after the triangulation is formed by eating triangles, spreading out from each hole point until PSLG segments block its progress; we must be careful to enclose each hole in segments, or your whole triangulation might be eaten away. If the two triangles abutting a segment are eaten, the segment itself is also eaten. We should not place a hole directly on a segment; if we do, Triangle will arbitrarily choose one side of the segment, (See Fig. 7.3 for an example).

We use the above mentioned file formats to give and receive data between our code and Triangle.

7.2 Generating mesh using Triangle

7.2.1 Command line switches

The command line syntax for running the Triangle is;

```
triangle [-prq__a__uAcDjevngBPNEIOXzo_YS__iFlsCQVh] input_file
```

Here the underscore indicates that a float(number) may optionally follow certain switches. We should not put a space between a switch and its numeric parameter(float). `input_file` must be a file with extension .node, or extension .poly if the -p switch is used. If -r is used, you must supply .node and .ele files, and possibly a .poly file and a .area file. All the letters inside the square brackets of syntax are command-line switches. Here we will explain the use of some command-line switches we use;

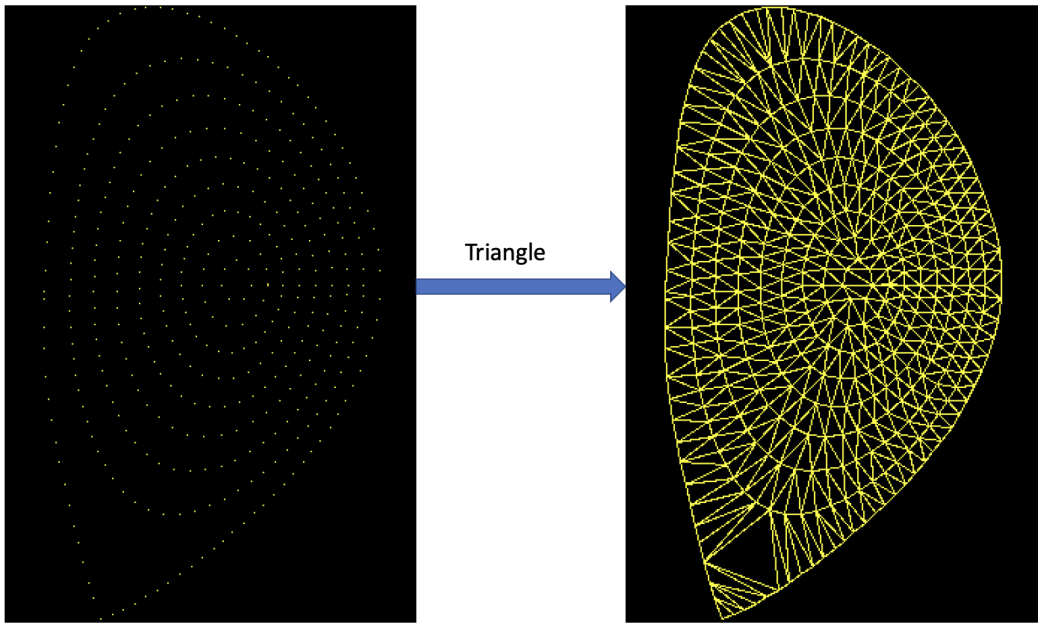
1. -p Triangulates a Planar Straight Line Graph (.poly file)
2. -r Refines a previously generated mesh.
3. -q Quality mesh generation with no angles smaller than 20 degrees. An alternate minimum angle may be specified after the 'q'
4. -a Imposes a maximum triangle area constraint. A fixed area constraint (that applies to every triangle) may be specified after the 'a', or varying area constraints may be read from a .poly file or .area file.

7.2.2 Generating Delaunay triangulations

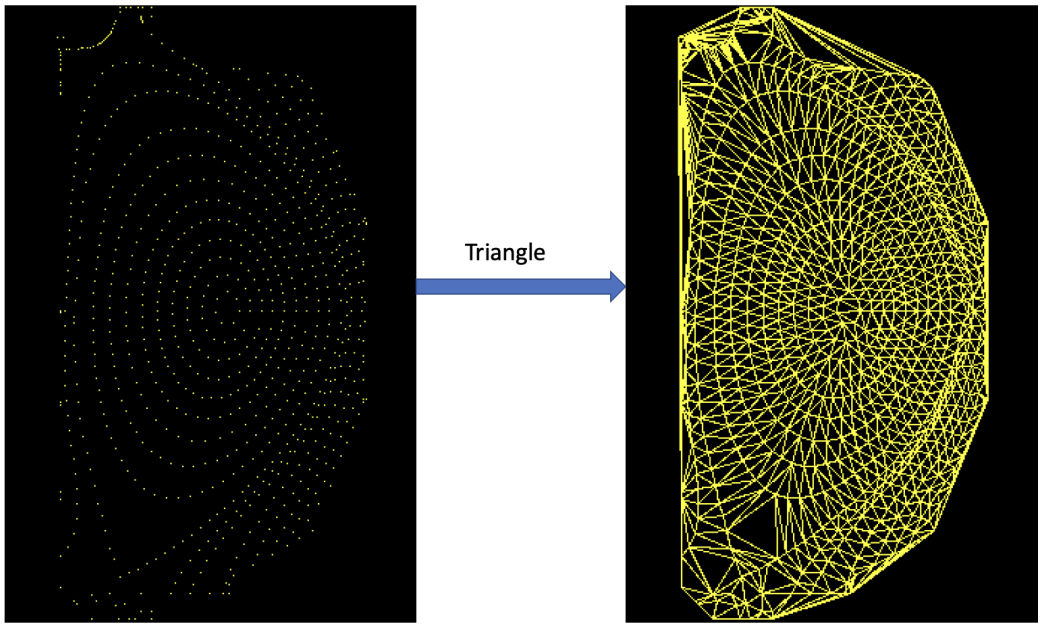
The default behaviour of Triangle is to generate Delaunay triangulations. If we make a .node file of all the vertices(Grids that we generate) and run Triangle on them with following command;

```
triangle input_file_name.node
```

We get the Delaunay triangulations. It is shown in the figure 7.4. From the figure we can see that the triangles constructed is of very huge in parts where there are less grid points. This makes us to make use of above mentioned command switches to make much more quality meshing.



(a) Triangulation of core region



(b) Triangulation of core+SOL+private region

Figure 7.4: This is the triangulation by Triangle by interconnecting the grids provided

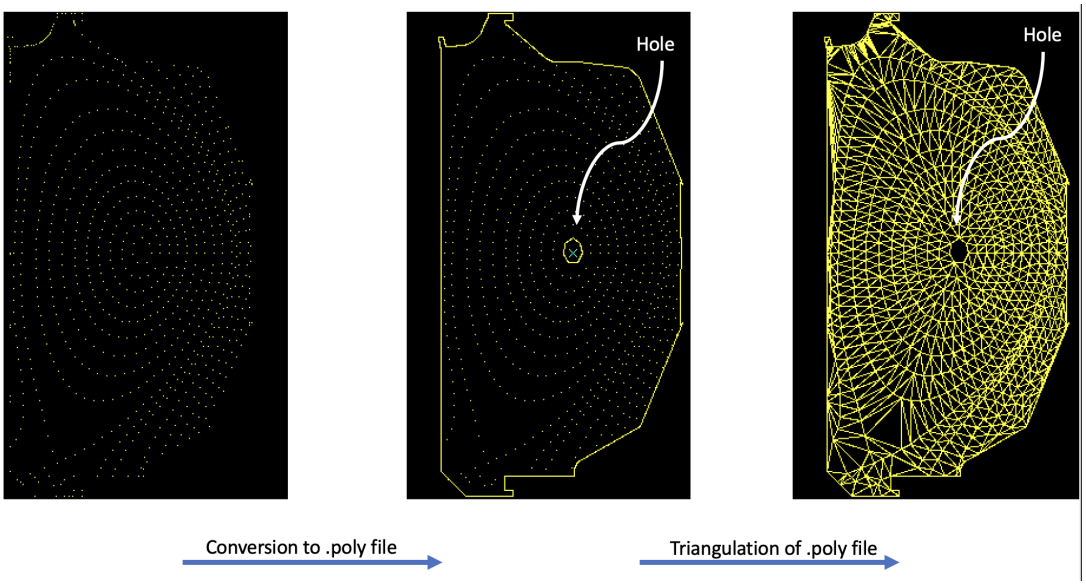


Figure 7.5: Conversion to .poly file input with the introduction of hole. Triangulation of this .poly file is in third figure

7.2.3 Mesh modifications

Converting the data points to .poly files

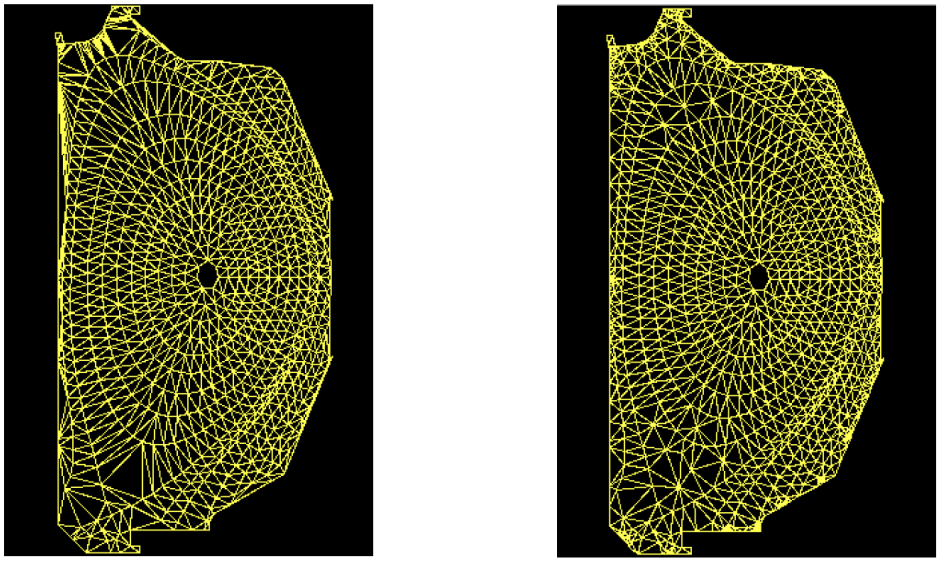
We know that just ordinary .node file can't incorporate holes in the surfaces. Since the area inside the smallest flux surface is negligible compared to others and triangulation inside that makes unnecessary small triangles in that region and cause disordered triangulation. So the best way to leave that place blank is by incorporating a hole in that region. To incorporate a hole we write the whole nodes in .poly file where we also give the data for hole. The .poly file is like a PSLG, so by the following command we can make a triangulation of this .poly file.

```
triangle -p input_file_name.poly
```

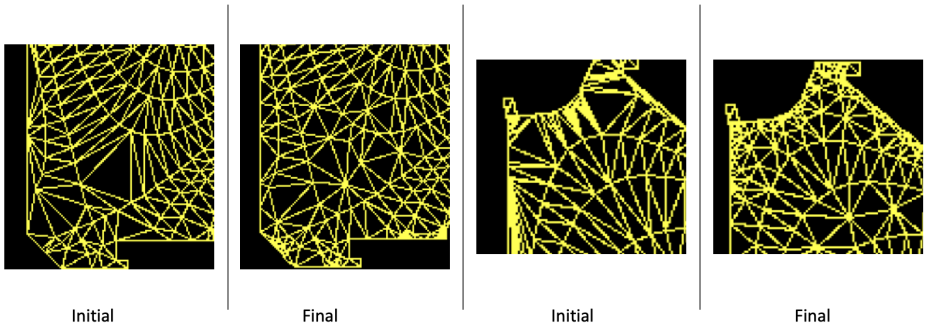
Now we get the Delaunay triangulations from a .poly file which is shown in figure 7.5

Quality meshing: Angle and Area constraints

When the -q or -a command switch is used, Triangle generates a conforming constrained Delaunay triangulation. These switches set constraints on angles and triangle sizes in the mesh. The -q switch sets a minimum angle constraint. The number that follow the '-q' command switch is taken as the minimum angle constrain; otherwise, if no number is specified the default minimum angle of twenty degrees is taken. (See figure 7.6). We can see that all the sharp triangles are removed. We should note here that the angle constraint does not apply to small angles between input segments; such angles cannot



(a) Refinement with angle constraint



(b) Improvements after refinement

Figure 7.6: This is the triangulation by Triangle by the command, this shows the minimum angle allowed is 20^o by default as there is no angle specified after -q command switch

be removed because that would mean changing the input data by Triangle. We use the below command to introduce angle constraint.

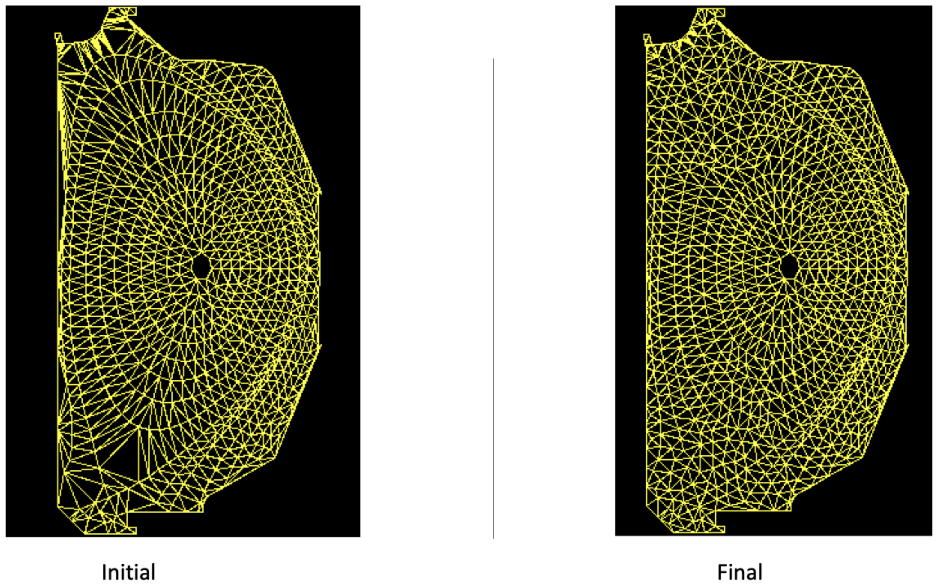
`triangle -rpq input_file_name.1` (result in figure 7.6)

The -r command switch represents the triangle is refined from previous triangulation, so the input to this refinement is given as the output of previous triangulation, this is represented by `input_file_name.1`

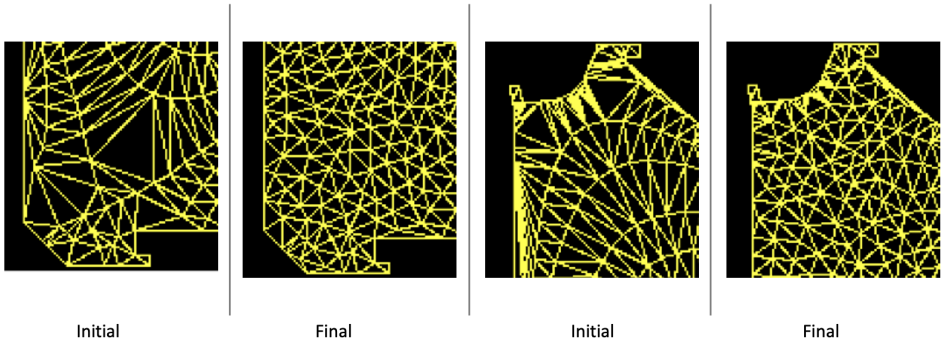
The -a switch sets a maximum area constraint. There are three ways to use this switch. The first way is to specify a maximum area on the command line. See figure 7.7. We use the below command to get area constraint of 0.001(in dimensionless units)

`triangle -rpa0.001 input_file_name.1`(result in figure 7.7)

It is possible to implement both angle and area constraint. This can be achieved by



(a) Refinement with area constraint



(b) Improvements after refinement

Figure 7.7: This is the refinement by Triangle from previous triangulation.

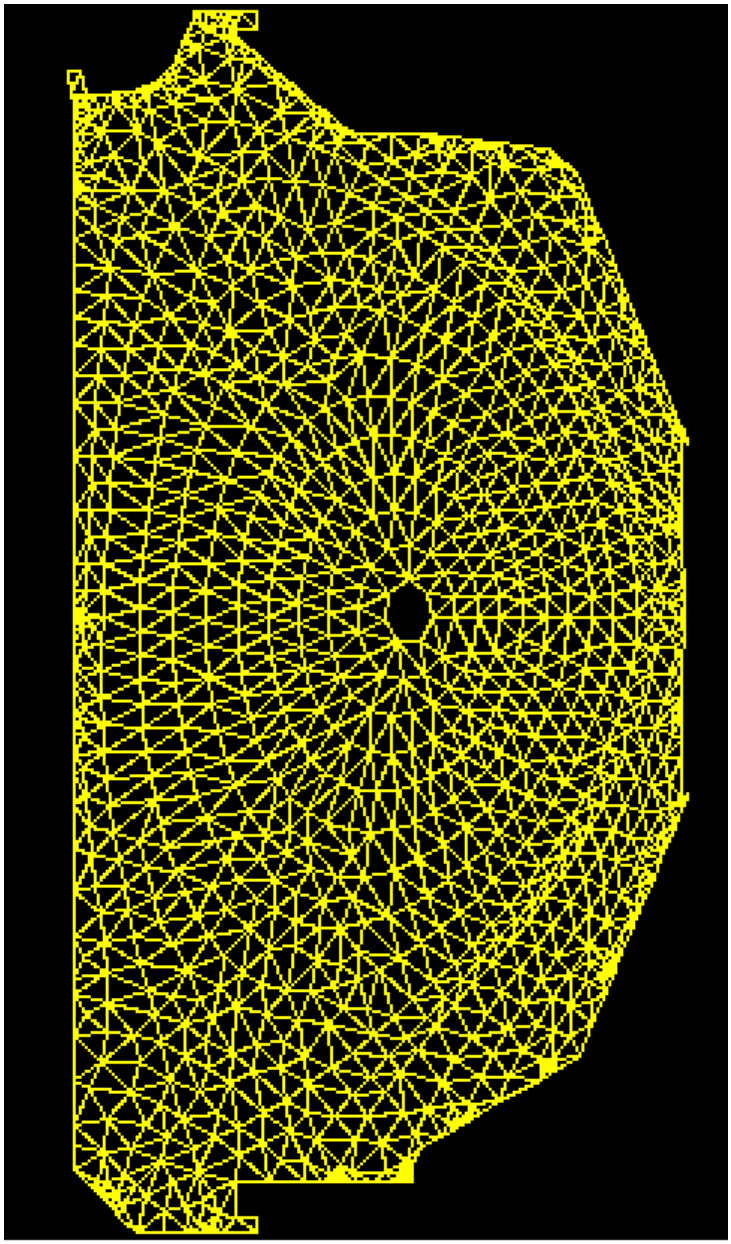


Figure 7.8: This is the partially structured mesh obtained by area and angle constraint.

following command.

`triangle -rpqa0.001 input_file_name.1`(result in figure 7.8)

Chapter 8

Optimization Processes

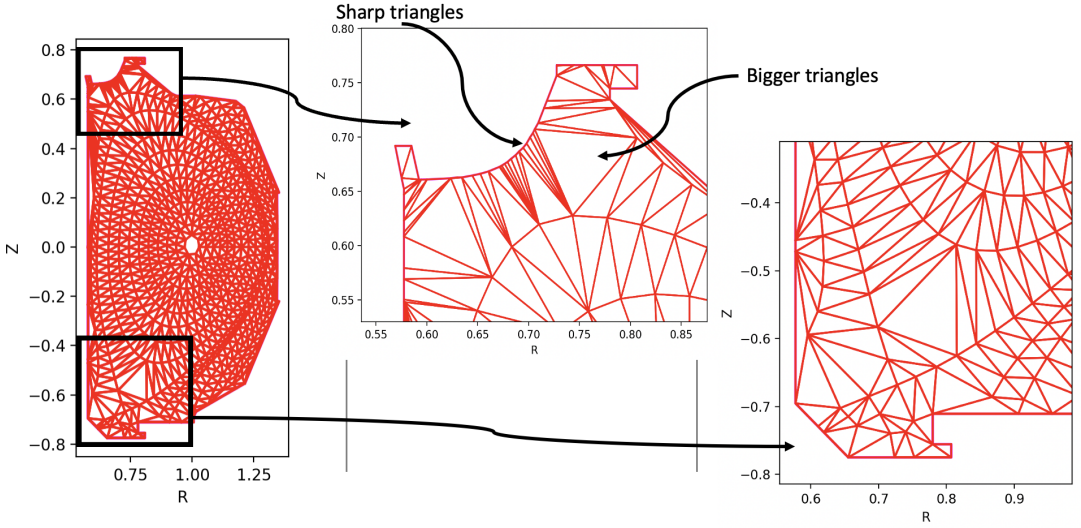
In this chapter, we describe various processes in which we can optimize the mesh. The main parameter that we use for the optimization is the variance of areas of triangles formed, and the other one is how close these triangles are to the equilateral triangles. However, the latter can be ensured by imposing an angle constrain of a particular value, So all our following discussion is based on how we can reduce the variance of areas of triangles formed. The lesser the variance, the more optimized the mesh is. The extent to which this optimization can go depends on the needs of simulation procedures, which the users determine.

8.1 Imposing area and angle constrains

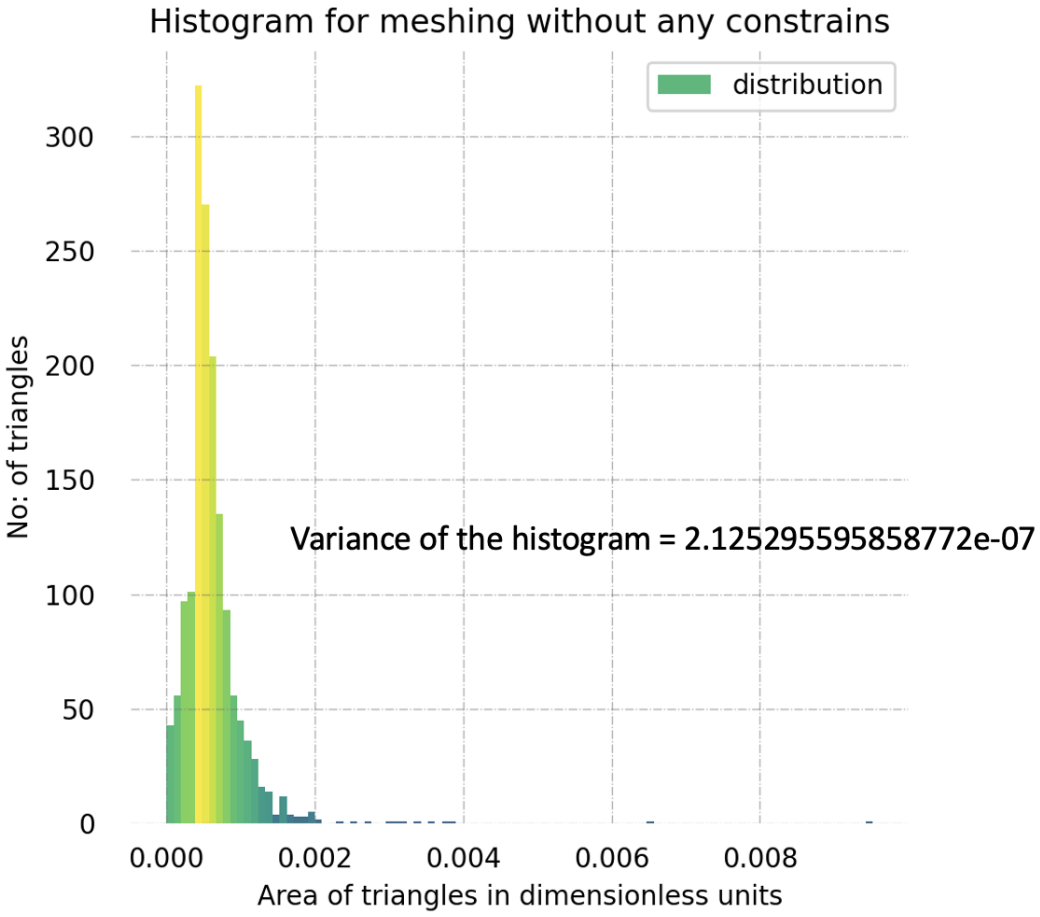
Imposing area constraints can significantly reduce the variance of areas of triangles formed. This is because whenever a giant triangle stands out from the rest, imposing an area constrain with an area less than this triangle will force the TRIANGLE to split it until all the areas of split triangles fall under the given area constrain.

In Fig 8.1(a), we have a triangulation with no constraints at all, and in Fig, we have a histogram showing the area of triangles with the number of triangles. We also show the variance of the distribution in Fig 8.1(b). In Fig 8.1(a), we have shown the zoomed part of regions that cause an increase in the histogram variance.

Now we introduce the area and angle constraints. Fig 8.2(a), shows the triangulation with the area and angle constraints. Here we have zoomed in on the parts which contribute more to variance. If we compare the zoomed part in Fig 8.1(a) and Fig 8.2(a), we can conclude that more enormous triangles are split into smaller triangles, and sharp triangles are replaced by triangles that are closer to equilateral triangles. All these will ultimately lead to optimization. The area constraint is 0.001(dimensionless units). So



(a) triangulation with no constraints at all and the zoomed part of regions that cause an increase in the histogram variance.



(b) The corresponding histogram with exact value of variance.

Figure 8.1: triangulation with no constraints at all and its histogram.

it can be verified from the histogram (Fig 8.2(b)) that all triangles are under this area because the range of the histogram is from 0 to 0.001.

From the histograms of both triangulations (Fig 8.1(b) and Fig 8.2(b)), we can see that we were able to reduce the variance of histograms by imposing area and angle constraints. The widths or range of the histogram is an indicator of variance. The higher the widths or range, the higher the variance. The exact value of the variance is also provided in respective histograms. By analyzing the exact values, we can see that we were able to reduce the variance by ten times by imposing area and angle constraints.

8.2 Approximating Limiter points

In Fig 8.2(a), we can see that there are small triangles formed near the limiter points. When limiter points are closer together, TRIANGLE tends to construct smaller triangles in that region. In the limiter points there are curved regions; such regions have limiter points placed close to each other (See Fig 8.3).

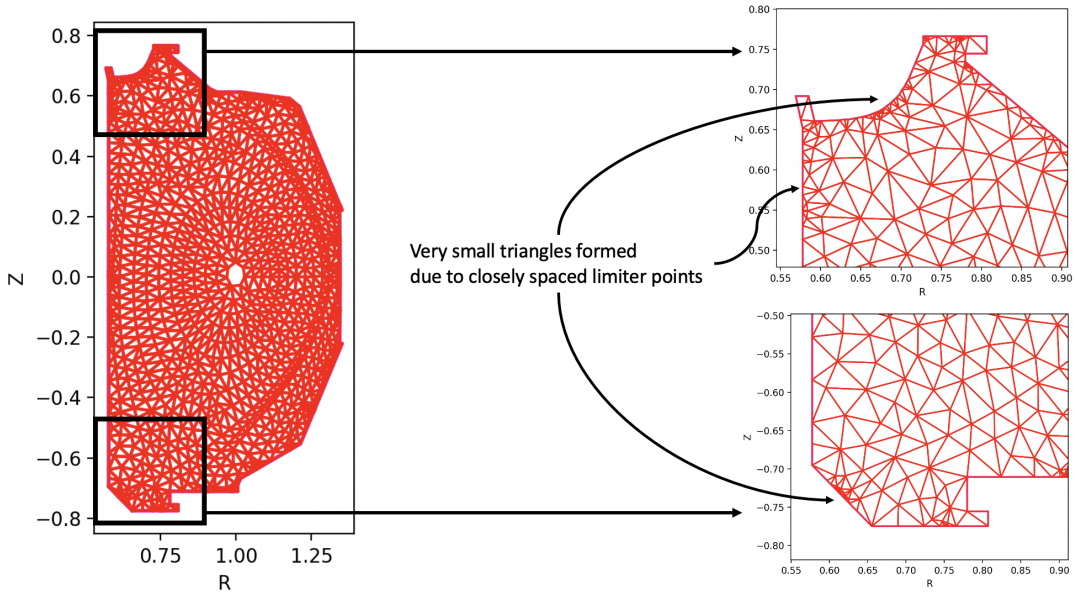
These smaller triangles also contribute to variance highly as their area is vastly different (smaller) than the area of triangles in the core and majority of the SOL region. So to avoid these smaller triangles, we need to remove closely spaced limiter points.

8.2.1 Selecting initial set of dominant points

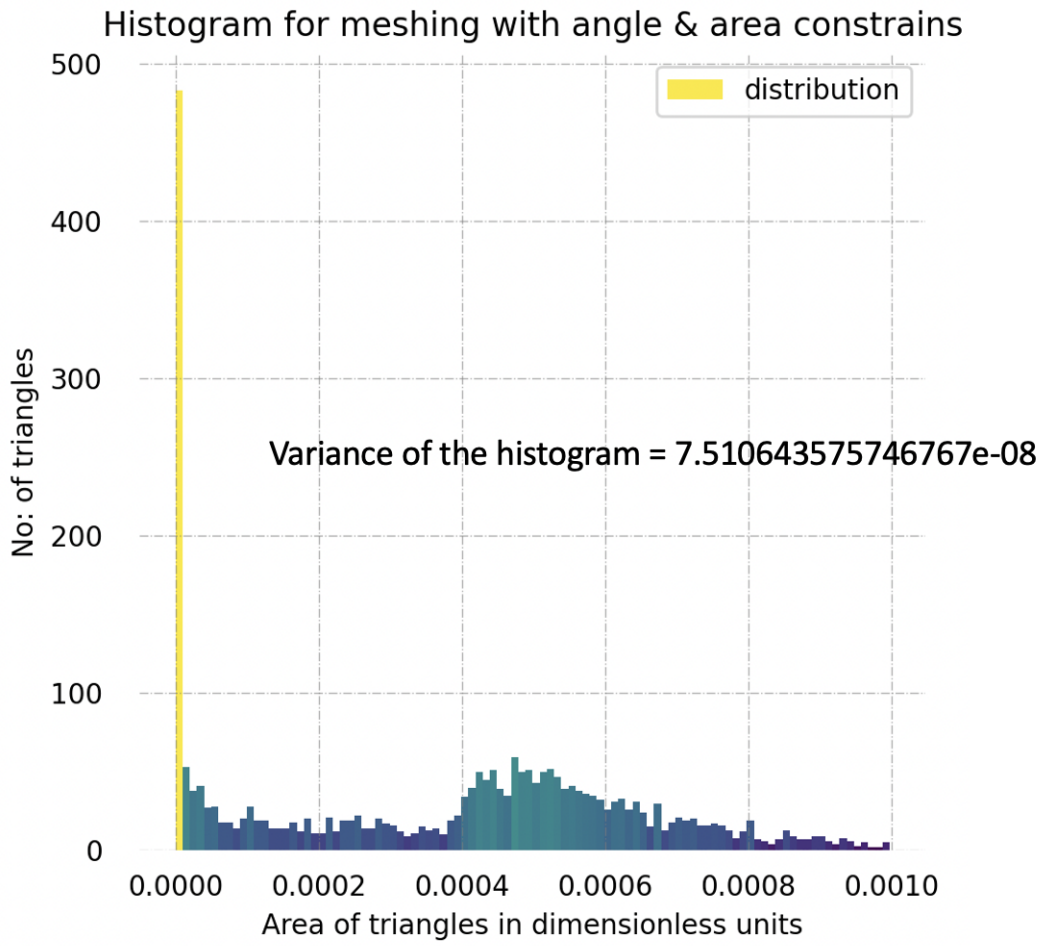
The way in which we remove closely spaced limiter points is by using a polygonal approximation method, see Fig 8.4. The first step in polygonal approximation is the selection of initial dominant points. The initial dominant points are co-planar points which are not in the end points of a line segment or we can say that initial dominant points are points in which there is no bending in lines that make up a polygon. These initial dominant points are identified by checking, it is co-planar to adjacent points or not [44]. The implementation of this algorithm is shown in Fig 8.5.

8.2.2 Polygonal approximation by removal of low significant points

The next step in the polygonal approximation is computing significant measures of every initial dominant point. To find the significant measure of a point S_k, we take it's preceding and exceeding points as S_{k-1} and S_{k+1} respectively. Now, these three points are subjected to coordinate transform as follows. We translate the line segment connecting S_{k-1} and S_{k+1} so that the point S_{k-1} coincides with the origin of the XY coordinate system and



(a) triangulation with area and angle constraints and the zoomed part of regions that caused an increase in the histogram variance in previous triangulation (Fig 8.1(a))



(b) The corresponding histogram with exact value of variance.

Figure 8.2: triangulation with area and angle constraints at all and its histogram.

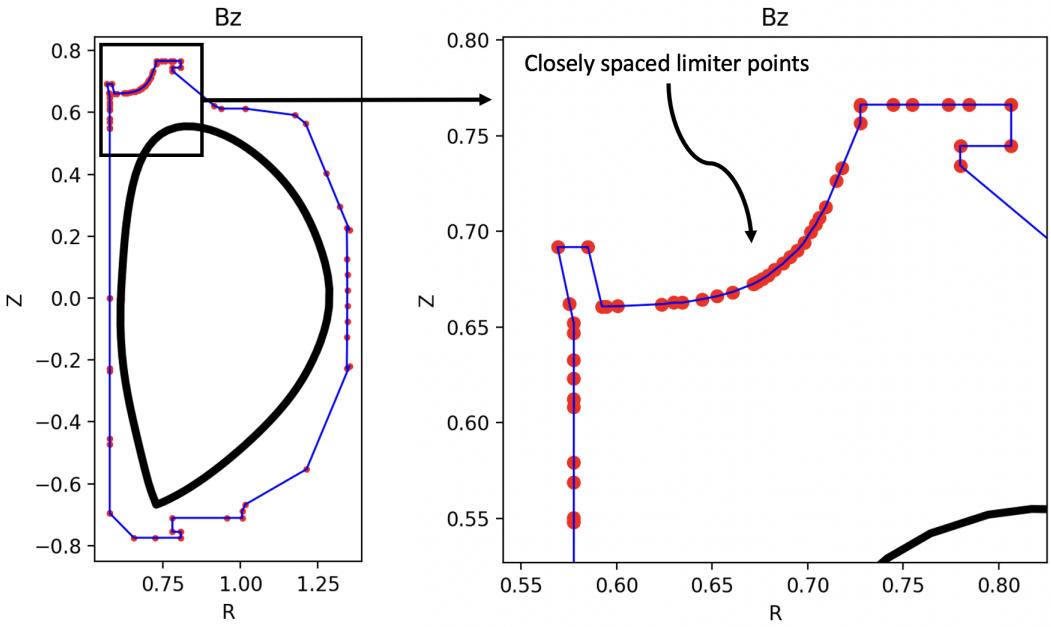


Figure 8.3: Limiter points. Zoomed part is where the curvature lies and limiter points lie close to each other.

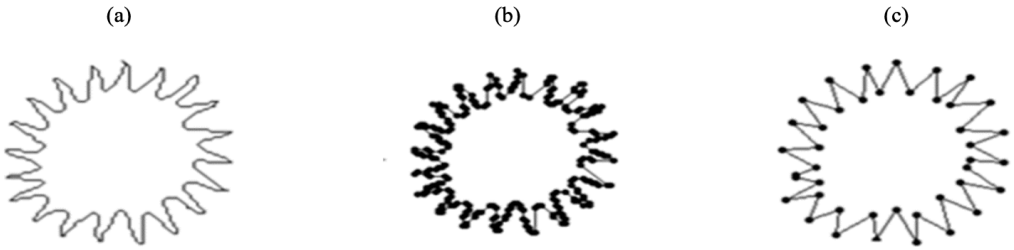


Figure 8.4: a)A digital curve representing the shape of a snowflake, b) Initial set of dominant points, c)Suitable polygonal approximation are shown here. [44]

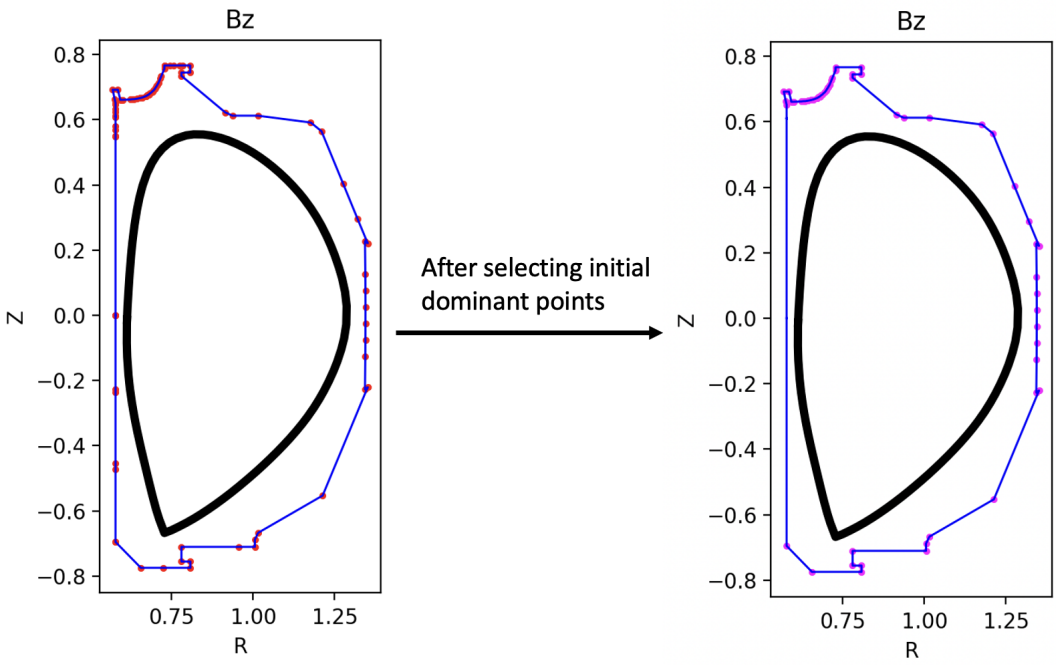


Figure 8.5: Identifying dominant points on limiter points

measure the amount of angle produced by the translated line segment with the x-axis. In order to align the translated line segment with the x-axis, rotate the line segment with a computed amount angle. The actual XY coordinates system and new transformed coordinate systems are displayed in Fig 8.6(a) and Fig 8.6(b).

Now all the work is done in transformed coordinates. In the next step by checking transformed x coordinate of S_k the method compute the significant measure. If the x coordinate of S_k is less than 0 then the significant measure sig(S_k) is computed using eqn. (1).(see Fig. 8.7(a)) if x_k of S_k lies between 0 and the x coordinate of S_{k+1} then the significant measure is computed using eqn. (2) (see Fig. 8.7(b). If the x_k value is greater than x coordinate of S_{k+1} then the significant measure of S_k is computed using eqn. (3) (See Fig. 8.7(c)).

$$sig(S_k) = \sqrt{(S_{x_k} - S_{x_{k-1}})^2 + (S_{y_k} - S_{y_{k-1}})^2} \quad \texttt{eqn. (1)}$$

$$sig(S_k) = \sqrt{(S_{y_k})^2} \quad \texttt{eqn. (2)}$$

$$sig(S_k) = \sqrt{(S_{x_k} - S_{x_{k+1}})^2 + (S_{y_k} - S_{y_{k+1}})^2} \quad \texttt{eqn. (3)}$$

This is how we find the significant measure of a point. This process is repeated until significant measures of all initial dominant points are computed. Then we eliminate the point with least significant measure and update the significant measure of S_{k-1} and S_{k+1} (preceding and exceeding points of the eliminated point). This elimination and updating

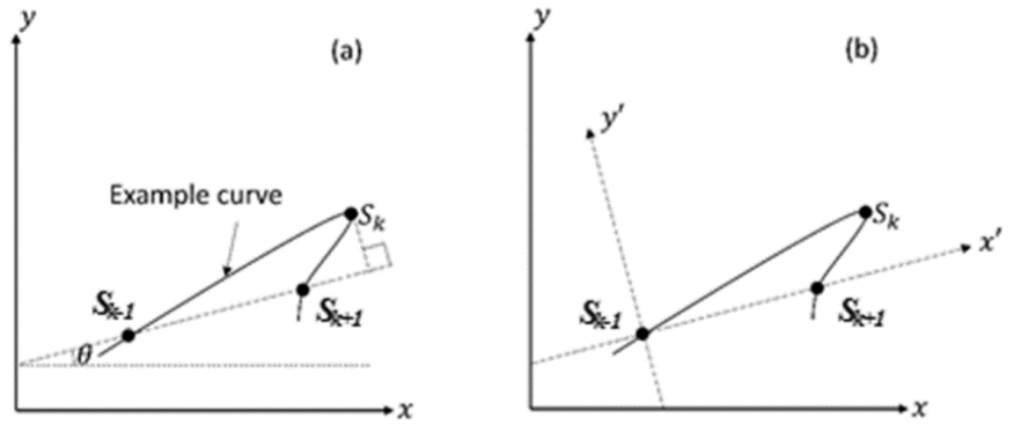


Figure 8.6: Demonstration of the coordinate transform performed. (a) An example curve in the original x-y coordinate system is shown. (b) The transformed x'-y' coordinate system is shown in addition to the original x-y system. [44]

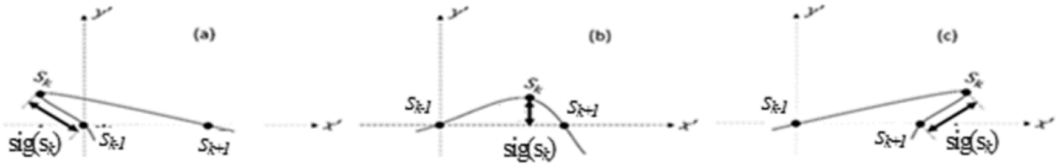


Figure 8.7: Demonstration of computation of significant measure of the point S_k from the line segment S_{k-1} S_{k+1}. [44]

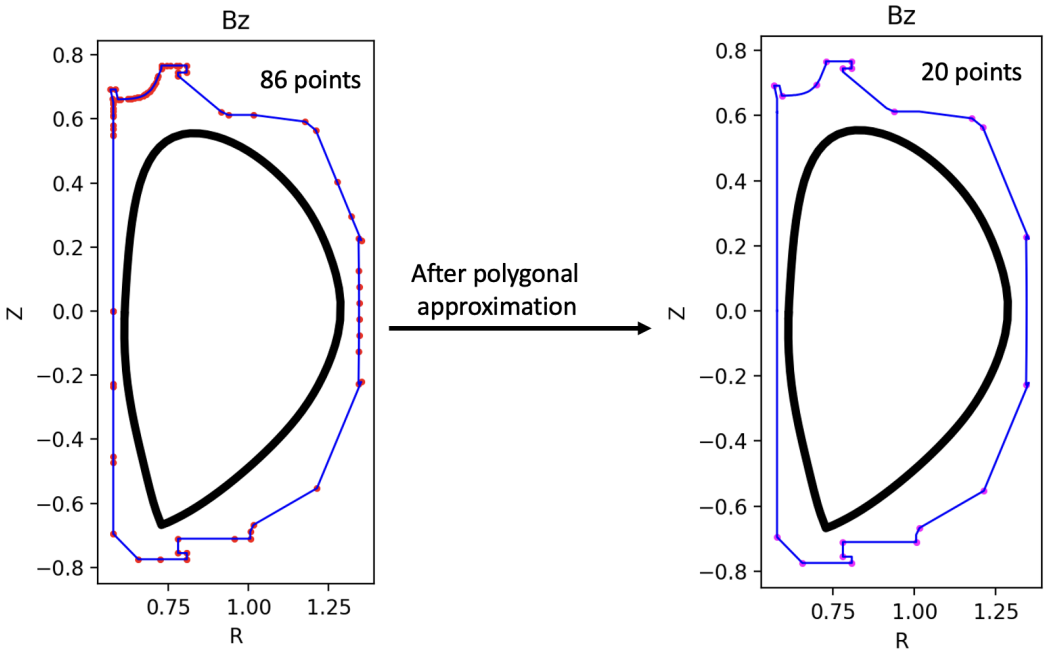


Figure 8.8: Approximating limiter points from 86 to 20 points.

is continued until the remaining points is equal to the desired number of points (number of points in which we need to approximate the polygon into) [44].

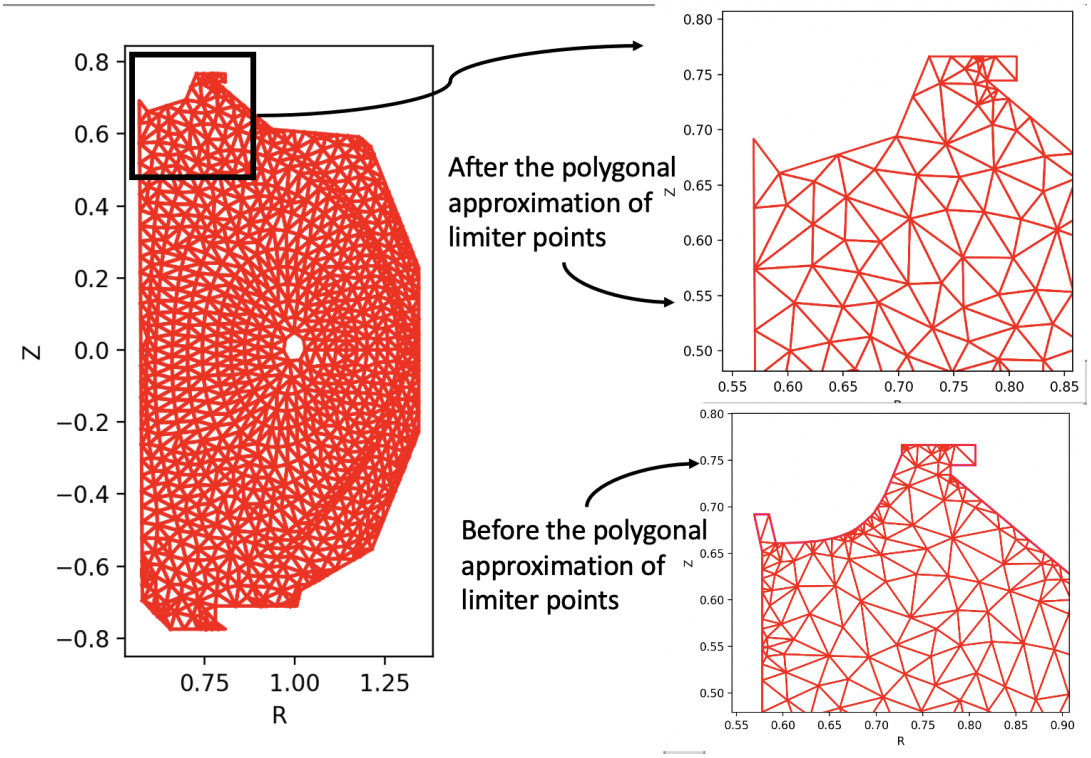
In our case the original limiter points had 86 points and we used the polygonal approximation and reduced it to 20 points, see Fig 8.8.

8.2.3 Resultant mesh and its variance

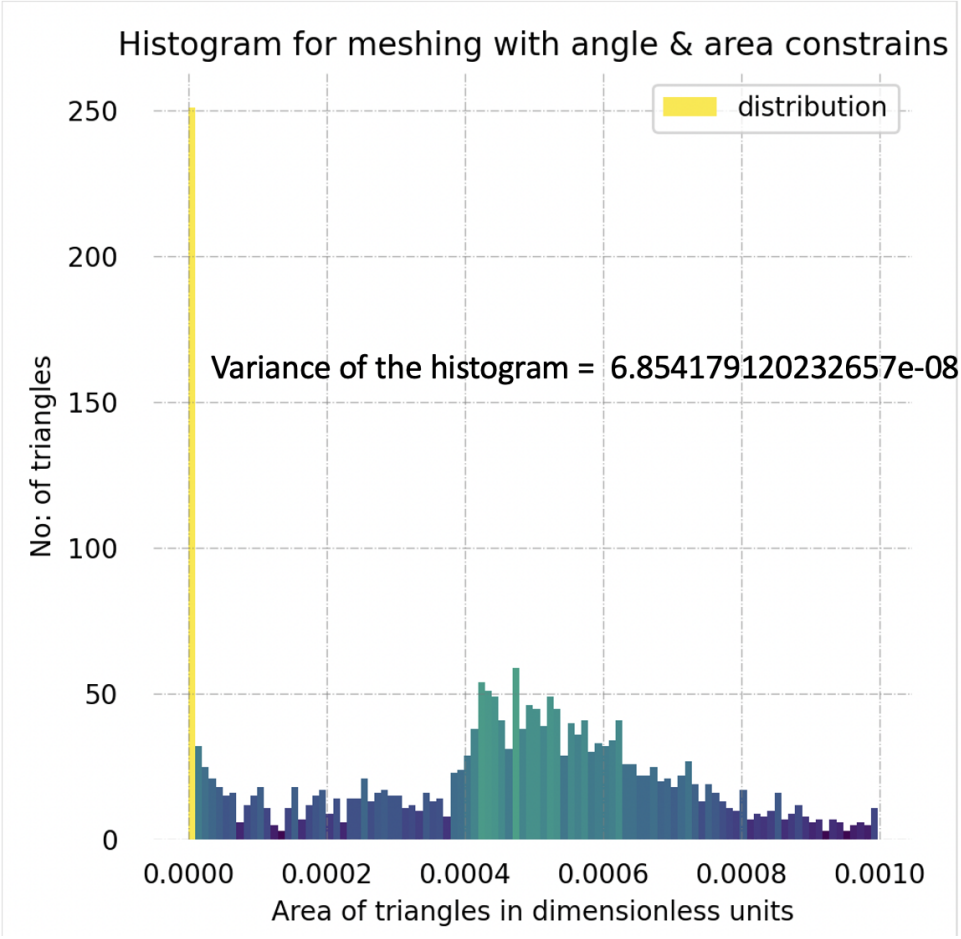
Since we have new limiter points, we use this points instead of our old limiter points and run TRIANGLE on them with exact same constraints implemented in section 8.1 (see Fig 8.9(a)). From the zoomed parts of Fig 8.1(a)and Fig 8.9(a), we can see that by reducing the number of limiter points by polygonal approximation, results in lowering the variance of histogram (see Fig 8.1(b)and Fig8.9(b)). Another important point here to note that (while comparing histograms in Fig 8.1(b)and Fig8.9(b)), we were able to reduce the number of smaller triangles by almost half and more number of triangles near limiter points got area comparable to areas of triangles in core regions (more number of triangles fall in the range 0.0004 and 0.0006).

8.3 Achievements

While introducing the new coordinate system in GTC-X, we could couple the core, private, and SOL regions. So in our thesis, we confirmed this by constructing flux surfaces



(a) triangulation (with area and angle constraints) after approximation of limiter points and the zoomed part of regions show the reduction in number of smaller triangles near limiter points



(b) The corresponding histogram with exact value of variance.

Figure 8.9: triangulation with area and angle constraints at all and its histogram.

and match with experimentally obtained results. We were also able to construct field-aligned grids, such that grids are obtained in all the parts inside limiter points. We then used TRIANGLE to construct meshes on these grids and showed different ways to customize them. We also proposed a few ways to optimize the meshes which meet our needs. All the proposed optimization techniques were followed by their implementation in our system. Then we were able to show that these techniques can optimize the mesh for any toroidal tokamak system.

8.4 Further work

After the successful optimization of mesh, we need to number these obtained triangles such that for a given point in a 2D poloidal cross-section, we should be able to get the triangle/triangle number in which the point lies. To get the triangle number, we need to check the given point with all the triangles and find in which triangle the point lies. However, to check this huge amount of triangles for just one particle and the case of millions of particles becomes extremely expensive in terms of computational work. So we use a neural network and train it with the numbering in the triangles. This way neural network will be able to tell the triangle or a nearby triangle of a point without checking all the triangles. The ordering of the triangle affects how well the training is done for the neural network. So the ordering should be such that the triangles should be continuous.

Bibliography

[1] J. P. Palutikof et al In M. L. Parry O. F. Canziani. "Climate Change 2007: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change". In: *IPCC* (2007), pp. 7–22. URL: https%3A%2F%2Fwww.ipcc.ch%2Freport%2Far4%2Fwg2%2F&usg=AOvVaw1LpfzwVxTneV-Y_uRR0WWx.

[2] John Cook et al. "Consensus on consensus: a synthesis of consensus estimates on human-caused global warming". In: *Environmental Research Letters* 11.4 (Apr. 2016), p. 048002. DOI: 10.1088/1748-9326/11/4/048002. URL: https://doi.org/10.1088/1748-9326/11/4/048002.

[3] Erika Böhm-Vitense. *Introduction to stellar astrophysics. Volume 3. Stellar structure and evolution.* Vol. 3. 1992.

[4] J D Huba. *NRL PLASMA FORMULARY Supported by The Office of Naval Research*. Washington, DC: Naval Research Laboratory, 2013, pp. 1–71. ISBN: 9781234126070. URL: http://wwwppd.nrl.navy.mil/nrlformulary/.

[5] J. Wesson and D.J. Campbell. *Tokamaks*. International series of monographs on physics. Clarendon Press, 2004. ISBN: 9780198509226. URL: https://books.google.co.in/books?id=iPlAwZI6HIYC.

[6] Dominik Michels. "Development of a high-performance gyrokinetic turbulence code for the edge and scrape-off layer of magnetic confinement fusion devices". PhD thesis. Fakultät für Informatik der Technischen Universität München, 2021.

[7] Jef Ongena et al. "Magnetic-confinement fusion". In: *Nature Physics* 12 (May 2016), pp. 398–410. DOI: 10.1038/nphys3745.

[8] Jeffrey P. Freidberg. *Plasma Physics and Fusion Energy*. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511755705.

[9] Jeffrey P. Freidberg. *Ideal MHD*. Cambridge University Press, 2014. DOI: 10.1017/CBO9780511795046.

[10] M. Z. Tokar. "Scrape-off layer modeling with kinetic or diffusion description of charge-exchange atoms". In: *Physics of Plasmas* 23.12 (2016), p. 122512. DOI: 10.1063/1.4972538. eprint: https://doi.org/10.1063/1.4972538. URL: https://doi.org/10.1063/1.4972538.

[11] A. J. Brizard and T. S. Hahm. "Foundations of nonlinear gyrokinetic theory". In: *Rev. Mod. Phys.* 79 (2 Apr. 2007), pp. 421–468. DOI: 10.1103/RevModPhys.79.421. URL: https://link.aps.org/doi/10.1103/RevModPhys.79.421.

[12] T T Ribeiro and B Scott. "Gyrofluid turbulence studies of the effect of the poloidal position of an axisymmetric Debye sheath". In: *Plasma Physics and Controlled Fusion* 50.5 (Apr. 2008), p. 055007. DOI: 10.1088/0741-3335/50/5/055007. URL: https://doi.org/10.1088/0741-3335/50/5/055007.

[13] P Ricci et al. "Simulation of plasma turbulence in scrape-off layer conditions: the GBS code, simulation results and code validation". In: *Plasma Physics and Controlled Fusion* 54.12 (Nov. 2012), p. 124047. DOI: 10.1088/0741-3335/54/12/124047. URL: https://doi.org/10.1088/0741-3335/54/12/124047.

[14] Andreas Stegmeir et al. "The field line map approach for simulations of magnetically confined plasmas". In: *Computer Physics Communications* 198 (May 2015). DOI: 10.1016/j.cpc.2015.09.016.

[15] P. Tamain et al. "The TOKAM3X code for edge turbulence fluid simulations of tokamak plasmas in versatile magnetic geometries". In: *Journal of Computational Physics* 321 (2016), pp. 606–623. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.05.038. URL: https://www.sciencedirect.com/science/article/pii/S0021999116301838.

[16] B. D. Dudson and J. Leddy. "Hermes: global plasma edge fluid turbulence simulations". In: *Plasma Physics and Controlled Fusion* 59.5, 054010 (May 2017), p. 054010. DOI: 10.1088/1361-6587/aa63d2. arXiv: 1609.03360 [physics.plasm-ph].

[17] Ben Zhu, Manaure Francisquez, and Barrett N. Rogers. "GDB: A global 3D two-fluid model of plasma turbulence and transport in the tokamak edge". In: *Computer Physics Communications* 232 (2018), pp. 46–58. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2018.06.002. URL: https://www.sciencedirect.com/science/article/pii/S001046551830208X.

[18] S. I. Braginskii. "Transport Processes in a Plasma". In: *Reviews of Plasma Physics* 1 (Jan. 1965), p. 205.

[19] A. Zeiler, J. F. Drake, and B. Rogers. "Nonlinear reduced Braginskii equations with ion thermal dynamics in toroidal plasma". In: *Physics of Plasmas* 4.6 (1997), pp. 2134–2138. DOI: 10.1063/1.872368. eprint: https://doi.org/10.1063/1.872368. URL: https://doi.org/10.1063/1.872368.

[20] Gregory W. Hammett and Francis W. Perkins. "Fluid moment models for Landau damping with application to the ion-temperature-gradient instability". In: *Phys. Rev. Lett.* 64 (25 June 1990), pp. 3019–3022. DOI: 10.1103/PhysRevLett.64.3019. URL: https://link.aps.org/doi/10.1103/PhysRevLett.64.3019.

[21] O. V. Batishchev et al. "Kinetic effects in tokamak scrape-off layer plasmas". In: *Physics of Plasmas* 4.5 (1997), pp. 1672–1680. DOI: 10.1063/1.872280. eprint: https://doi.org/10.1063/1.872280. URL: https://doi.org/10.1063/1.872280.

[22] F Jenko and W Dorland. "Nonlinear electromagnetic gyrokinetic simulations of tokamak plasmas". In: *Plasma Physics and Controlled Fusion* 43.12A (Nov. 2001), A141–A150. DOI: 10.1088/0741-3335/43/12a/310. URL: https://doi.org/10.1088/0741-3335/43/12a/310.

[23] X. Q. Xu R. H. Cohen. *Progress in Kinetic Simulation of Edge Plasmas*. Vol. 48. 2008.

[24] W. W. Lee. "Gyrokinetic approach in particle simulation". In: *The Physics of Fluids* 26.2 (1983), pp. 556–562. DOI: 10.1063/1.864140. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.864140. URL: https://aip.scitation.org/doi/abs/10.1063/1.864140.

[25] A.M. Dimits and W.W. Lee. "Partially Linearized Algorithms in Gyrokinetic Particle Simulation". In: *Journal of Computational Physics* 107.2 (1993), pp. 309–323. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1993.1146. URL: https://www.sciencedirect.com/science/article/pii/S0021999183711460.

[26] S. E. Parker and W. W. Lee. "A fully nonlinear characteristic method for gyrokinetic simulation". In: *Physics of Fluids B: Plasma Physics* 5.1 (1993), pp. 77–86. DOI: 10.1063/1.860870. eprint: https://doi.org/10.1063/1.860870. URL: https://doi.org/10.1063/1.860870.

[27] C.S. Chang et al. "Gyrokinetic projection of the divertor heat-flux width from present tokamaks to ITER". In: *Nuclear Fusion* 57.11 (Aug. 2017), p. 116023. DOI: 10.1088/1741-4326/aa7efb. URL: https://doi.org/10.1088/1741-4326/aa7efb.

[28] W. M. Nevins et al. "Discrete particle noise in particle-in-cell simulations of plasma microturbulence". In: *Physics of Plasmas* 12.12 (2005), p. 122305. DOI: 10.1063/1.2118729. eprint: https://doi.org/10.1063/1.2118729. URL: https://doi.org/10.1063/1.2118729.

[29] John A. Krommes. "Nonequilibrium gyrokinetic fluctuation theory and sampling noise in gyrokinetic particle-in-cell simulations". In: *Physics of Plasmas* 14.9 (2007), p. 090501. DOI: 10.1063/1.2759879. eprint: https://doi.org/10.1063/1.2759879. URL: https://doi.org/10.1063/1.2759879.

[30] George J. Wilkie and William Dorland. "Fundamental form of the electrostatic f-PIC algorithm and discovery of a converged numerical instability". In: *Physics of Plasmas* 23.5 (2016), p. 052111. DOI: 10.1063/1.4948493. eprint: https://doi.org/10.1063/1.4948493. URL: https://doi.org/10.1063/1.4948493.

[31] A Staniforth and J Cote. "Semi-Lagrangian integration schemes for atmospheric models - A review". In: *Monthly Weather Review; (United States)* 119 (Sept. 1991). ISSN: 0027-0644. DOI: 10.1175/1520-0493(1991)119<2206:SLISFA>2.0.CO;2. URL: https://www.osti.gov/biblio/6059770.

[32] A. K. Stegmeir. "GRILLIX: A 3D turbulence code for magnetic fusion devices based on a field line map". PhD thesis. Ph.D. thesis, Technische Universit at Mu nchen, 2015.

[33] Z. Lin et al. "Turbulent Transport Reduction by Zonal Flows: Massively Parallel Simulations". In: *Science* 281.5384 (1998), pp. 1835–1837. DOI: 10.1126/science.281.5384.1835. eprint: https://www.science.org/doi/pdf/10.1126/science.281.5384.1835. URL: https://www.science.org/doi/abs/10.1126/science.281.5384.1835.

[34] Yong Xiao et al. "Gyrokinetic particle simulation of microturbulence for general magnetic geometry and experimental profiles". In: *Physics of Plasmas* 22.2 (2015), p. 022516. DOI: 10.1063/1.4908275. eprint: https://doi.org/10.1063/1.4908275. URL: https://doi.org/10.1063/1.4908275.

[35] Mark W. Beall, Joe Walsh, and Mark S. Shephard. "A Comparison of Techniques for Geometry Access Related to Mesh Generation". In: *Eng. with Comput.* 20.3 (Sept. 2004), pp. 210–221. ISSN: 0177-0667. DOI: 10.1007/s00366-004-0289-z. URL: https://doi.org/10.1007/s00366-004-0289-z.

[36] Hoffmann CM. *Geometric and solid modeling*. San Mateo: Morgan Kaufmann, 1989.

[37] Mark S. Shephard and Marcel K. Georges. "Reliability of automatic 3D mesh generation". In: *Computer Methods in Applied Mechanics and Engineering* 101.1 (1992), pp. 443–462. ISSN: 0045-7825. DOI: https://doi.org/10.1016/0045-7825(92)90033-G. URL: https://www.sciencedirect.com/science/article/pii/004578259290033G.

[38] Fan Zhang et al. "Mesh Generation for Confined Fusion Plasma Simulation". In: *Eng. with Comput.* 32.2 (Apr. 2016), pp. 285–293. ISSN: 0177-0667. DOI: 10.1007/s00366-015-0417-y. URL: https://doi.org/10.1007/s00366-015-0417-y.

[39] Mark W Beall and Mark S Shephard. "A general topology-based mesh data structure". In: *International Journal for Numerical Methods in Engineering* 40.9 (1997), pp. 1573–1596.

[40] Rao V Garimella. "Mesh data structure selection for mesh generation and FEA applications". In: *International journal for numerical methods in engineering* 55.4 (2002), pp. 451–478.

[41] Waldemar Celes, Glaucio Paulino, and Rodrigo Espinha. "A compact adjacency-based topological data structure for finite element mesh representation". In: *International Journal for Numerical Methods in Engineering* 64 (Nov. 2005), pp. 1529–1556. DOI: 10.1002/nme.1440.

[42] Jonathan Richard Shewchuk. "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator". In: *Applied Computational Geometry: Towards Geometric Engineering*. Ed. by Ming C. Lin and Dinesh Manocha. Vol. 1148. Lecture Notes in Computer Science. From the First ACM Workshop on Applied Computational Geometry. Springer-Verlag, May 1996, pp. 203–222.

[43] S. De et al. "Kinetic particle simulations in a global toroidal geometry". In: *Physics of Plasmas* 26.8 (2019), p. 082507. DOI: 10.1063/1.5108684. eprint: https://doi.org/10.1063/1.5108684. URL: https://doi.org/10.1063/1.5108684.

[44] Mangayarkarasi Ramaiah and Dilip Prasad. "Polygonal approximation of digital planar curve using novel significant measure". In: (Dec. 2018).