Exploring turbulence and filaments in Tokamak exhaust via simulations

Sanfilippo Andrea - CID 06034359
Venturato Giancarlo - CID 06033934
Project supervisor Dr. Kingham, Robert J.
Project assessor Dr. Andrew, Yasmin

***Abstract*—Magnetic confinement fusion (MCF) offers a very promising route to clean and sustainable energy by harnessing nuclear fusion. Tokamaks, which confine plasma using strong magnetic fields, play the most important role in this endeavour. However, instabilities in the scrape-off layer (SOL) caused by edge-localized modes (ELMs) generate turbulent plasma filaments that heavily challenge the stability of the confinement and the longevity of the material. Understanding these filament dynamics is essential to the advancement of fusion reactor designs and tech. Our report presents the development and subsequent implementation of a 2D turbulent filament solver based on drift-reduced plasma fluid equations to investigate filament behaviour in the SOL. The model involves many physical processes, including $E \times B$ advection, magnetic curvature effects, and dissipative forces through the diffusion phenomenon. Our simulations managed to validate the theoretical predictions, thus demonstrating the radial propagation of filaments, their deformation, and the limited work of the dissipative effects. The solver's results align with other experimental findings, supporting the use of sheath dissipation closures for SOL studies [Nie14]. Our findings have possible implications for the enhancement of plasma confinement and the mitigation of ELM related damage, providing a way to optimize the Tokamak's performance. Future work could include extending the model to three-dimensional simulations and validate findings with experimental data from operating fusion devices.**

I. CONTENTS

II. Introduction

MAGNETIC confinement fusion (MCF) is the most promising approach to achieving clean and sustainable energy through the use of nuclear fusion [Mil22]. Central to this effort are Tokamaks, toroidal devices that confine high-temperature plasma via the use of strong magnetic fields. The plasma in a Tokamak is divided into two different regions: the core, where magnetic field lines are closed and confined to toroidal surfaces (thus streamlining the flux of plasma inside), and the "scrape-off layer" (SOL) (see Figure 6), which lies beyond the core and is characterized by open magnetic field lines and a generally turbulent behaviour of plasma. The magnetic field lines in the SOL spiral outward and eventually collide with the walls of the structure. The SOL plays a critical role in regulating how particles and thermal energy are exhausted from the plasma to the walls, thus ensuring the stability of the confinement system.

Under ideal and streamlined conditions, all of this exhaust process in the SOL is steady and manageable, however, instabilities at the interface between the core plasma and the SOL cause significant trouble to the modellization. These instabilities, better known as edge-localized modes (ELMs), periodically inject bursts of particles (and thus energy) from the core into the SOL. ELMs manifest as filaments, or localized regions of higher density and temperature plasma that are elongated following the magnetic field lines. These filaments propagate radially outward throughout the SOL in a turbulent manner and interact in complex ways [DZ11]. To understand and to control the dynamics of these filaments is certainly one of the most pressing challenges in the field of magnetic confinement fusion research. Their behavior has significant implications for material erosion, plasma performance, and the overall viability of future and extant fusion reactors.

Scope of the project

This project focuses on the development of a basic turbulent filament solver to study the dynamics of these filaments (or blobs, as we will also call them throughout the report) in a simplified 2D geometry. The solver will be based on the drift-reduced plasma fluid equations, which describe the evolution of important properties, such as density and vorticity, in a plane perpendicular to the magnetic field lines. By modelling the behaviour of individual blobs, our project aims to discover how the different terms in the equations affect the evolution of these blobs in a 2D plane, in order to better understand the rich plasma dynamics that occur inside the SOL [Nie14]. The project mainly revolves around the set of three equations[1]:

$$\frac{dn}{dt} = \frac{n\phi}{L_\parallel} - \frac{n-n_0}{L_\parallel} + ng\frac{\partial\phi}{\partial y} - g\frac{\partial n}{\partial y} + D_n\nabla_\perp^2 n$$

$$\frac{d\Omega}{dt} = \frac{\phi}{L_\parallel} - \frac{g}{n}\frac{\partial n}{\partial y} + \mu_i\nabla_\perp^2\Omega$$

$$\Omega = \nabla^2{}_\perp\Phi$$

The general scope of this project is in part theoretical and in part computational. On the theoretical side, it involves the understanding of the complex physics behind the simplified plasma model to ensure the correct application of the reduced plasma equations and to capture the essential tenets of blob dynamics. On the computational side, it requires the implementation of a numerical solver for the drift-reduced fluid equations, the validation of the code using analytical solutions in tractable limits, and the comparison of the results with the theoretical expectations. This dual approach tries to provide a sturdy framework for better investigating the behaviour of the plasma along the edge of a Tokamak.

Geometry

We employed a simplified geometry to facilitate the computational modeling while retaining most key features of blob dynamics. We used a 2D plane perpendicular to the magnetic field lines, capturing the cross-sectional evolution of the filament as it propagates radially outward. This choice reduces computational complexity while still allowing for a detailed exploration of the underlying physics. The initial conditions included a blob with his density represented by a Gaussian over a constant background to represent a single ELM filament, and we created a solver to track its subsequent evolution as it interacts with the surrounding plasma and the change in Electrical field and Potential.

[1]The equation set is gonna be explained in more details in the Discussion section

III. Discussion of the methods and topics investigated

THE analysis we performed started with a broad understanding of the 3D model, which is an electrostatic drift-fluid framework designed to simulate the dynamics of plasma filaments in the scrape-off layer (SOL) of magnetic confinement devices [Nie14] [Eas14]. It has been devised to capture essential physical processes while making a series of simplifying assumptions to focus on the dominant effects. With these premises we are able to reduce the necessary equations to these four:

$$\frac{dn}{dt} = -\nabla_{\parallel}(nV) + ng\frac{\partial\phi}{\partial y} - g\frac{\partial n}{\partial y} + D_n\nabla_{\perp}^2 n + S_n$$
$$\frac{d\Omega}{dt} = -U\nabla_{\parallel}\Omega + \frac{1}{n}\nabla_{\parallel}J_{\parallel} - \frac{g}{n}\frac{\partial n}{\partial y} + \mu_i\nabla_{\perp}^2\Omega$$
$$\frac{dU}{dt} = -U\nabla_{\parallel}U - \nabla_{\parallel}\phi - \frac{\mu_{\parallel}}{n}J_{\parallel} - \frac{S_n U}{n}$$
$$\frac{dV}{dt} = -V\nabla_{\parallel}V + \mu_{\parallel}\nabla_{\parallel}\phi - \frac{\mu_{\parallel}}{n}\nabla_{\parallel}n - \frac{\mu_{\parallel}}{n}J_{\parallel} - \frac{S_n V}{n}$$

The model describes the evolution in time of many plasma variables, such as density (n), vorticity (Ω), and parallel ion (U) and electron (V) velocities, under the influence of electromagnetic forces, curvature effects, and dissipation. The goal is mainly to balance physical realism with computational feasibility to study filamentary motion. The model makes some important assumptions about the physics that are extremely important for the understanding of the subject:

1) **Plasma Composition:**
   - The plasma consists of singly charged cold ions and isothermal electrons.
   - Ion temperature is assumed negligible, while electron temperature (T_e) remains constant.
2) **Geometry:**
   - A slab geometry is used, with a uniform magnetic field aligned in the z-direction ($B = B\hat{z}$).
   - Magnetic curvature and gradients are incorporated as additional terms in the governing equations.
3) **Normalization:**
   - Time is normalized to the ion gyrofrequency ($\Omega_i = eB/m_i$).
   - Lengths are normalized[2] to the hybrid gyroradius ($\rho_s = c_s/\Omega_i$), where $c_s = \sqrt{T_e/m_i}$ is the ion-acoustic speed.
   - Plasma density (n), electrostatic potential (ϕ), and velocities are normalized to characteristic SOL values.
4) **Collisional Effects:**
   - Dissipation, like perpendicular viscosity (μ_i) and density diffusion (D_n), are modeled specifically for SOL conditions.
   - Parallel collisional drag is included for ion and electron velocities via a simplified collision frequency.
5) **Electrostatic Limit:**
   - Our model assumes the electrostatic approximation and thus neglects variations in the magnetic field. This is only valid for regimes where magnetic perturbations are negligible, which we assume ours to be.
6) **Boussinesq Approximation:**
   - Density perturbations are assumed negligible relative to the background density thus simplifying current continuity calculations [Kra22] [Nie14].
7) **Perpendicular Dynamics:**
   - Perpendicular advection, magnetic curvature effects and diffusion are included, but higher-order corrections like gyroviscosity are neglected.
8) **Parallel Transport:**
   - Parallel dynamics are simplified using a closure using sheath boundary conditions ignoring higher-order effects like thermal transport.

The model solves for four variables: plasma density (n), vorticity (Ω), parallel ion velocity (U), and parallel electron velocity (V) that are governed by evolution equations. The set of differential equations tries to balance different forces that are results of the changing environmental conditions:

- Advection (driven by $E \times B$ motion),
- Parallel transport (via currents and electric fields),

[2]more on normalization in the apposite section, III

- Magnetic curvature effects,
- Collisional dissipation.

Sheath boundary conditions at the parallel edges ($z = \pm L_\parallel$) allow us to capture interactions with components that are facing the plasma:

- **Bohm sheath criterion:** in primis ions enter the sheath at the sound speed: $U = \pm 1$.
- **Electron flux condition:** in secundis electrons follow a Boltzmann-like relation: $V = \pm \exp(-\phi)$.

Dimensionless parameters are also used to characterize physical effects:

- **Curvature parameter:** $\gamma = \frac{2\rho_s}{R_c}$, where R_c is the magnetic field's radius of curvature.
- **Diffusion coefficient:** D_n, representing perpendicular particle diffusion because of collisions.
- **Viscosity:** μ_i, for perpendicular ion viscosity.
- **Collision frequency:** $\eta_\parallel$, for parallel electron-ion drag.

2D reduction of the model

To reduce the complexity of the full 3D model, certain assumptions about the parallel dynamics of the plasma filaments are made. This results in two-dimensional closures, which simplify the system by integrating out the parallel dimension. The two main closures discussed are the sheath dissipation closure and the vorticity advection closure. Each one of them makes distinct assumptions about the parallel dynamics and their role in blob behaviour.

The former assumes the filament is fully connected to the sheath, meaning it extends along the entire parallel length of the field lines from one sheath to the other. This leads to a series of primary assumptions:

- **Negligible parallel gradients:** The variation in the parallel direction of density (n) and electrostatic potential (ϕ) are assumed negligible.
- **Sheath current contribution:** Parallel currents are closed through the sheath. The dynamics are integrally averaged along the parallel direction, and similar effects are incorporated via sheath boundary conditions.

From these assumptions, we can derive the governing 2D equations for density and vorticity:

$$\frac{dn}{dt} = \frac{n\phi}{L_\parallel} - \frac{n - n_0}{L_\parallel} + n\gamma\frac{\partial\phi}{\partial y} - \gamma\frac{\partial n}{\partial y} + D_n\nabla_\perp^2 n$$

$$\frac{d\Omega}{dt} = \frac{\phi}{L_\parallel} - \frac{\gamma}{n}\frac{\partial n}{\partial y} + \mu_i\nabla_\perp^2\Omega$$

Here:

- $L_\parallel$ is the parallel connection length.
- n_0 is the background density.
- γ represents magnetic curvature effects.
- D_n and μ_i are diffusion terms, describing collisional dissipation.

The sheath dissipation model predicts that the filament's radial velocity depends on its perpendicular size ($d_\perp$). For small filaments only ($d_\perp < d^*$), polarization currents dominate, leading to a velocity scaling as $v_b \propto \sqrt{d_\perp}$. Instead, for larger blobs ($d_\perp > d^*$), sheath currents dominate, causing the velocity to scale inversely with size, $v_b \propto 1/d_\perp^2$. The critical size d^* is given by:

$$d^* = \left(\frac{\gamma L_\parallel^2}{2}\frac{\delta n}{n_0 + \delta n}\right)^{1/5}$$

The vorticity advection closure is a different approach that assumes the parallel dynamics are dominated by parallel gradients instead of sheath currents. Its main features are, in contrast:

- **Negligible parallel currents:** It assumes parallel currents ($J_\parallel$) are sufficiently small, which is very often valid in cases with high resistivity near the target.
- **Inclusion of parallel gradients:** This closure retains the effects of parallel density gradients even in its simplified representation

The resulting 2D equations are:

$$\frac{dn}{dt} = -\frac{n - n_0}{2L_b} + n\gamma\frac{\partial\phi}{\partial y} - \gamma\frac{\partial n}{\partial y} + D_n\nabla_\perp^2 n$$

$$\frac{d\Omega}{dt} = -\frac{\Omega}{2L_b} - \frac{\gamma}{n}\frac{\partial n}{\partial y} + \mu_i\nabla_\perp^2\Omega$$

Here, L_b represents the parallel length scale of the filament. Unlike the sheath dissipation closure, the vorticity advection model predicts the radial velocity of the filament scales as $v_b \propto \sqrt{d_\perp}$ for all sizes.

Both models offer complementary insights but are suited to different physical scenarios. The sheath dissipation closure captures the role of sheath currents in filament dynamics, while the vorticity advection closure is applicable in conditions where parallel currents are suppressed [Nie14]. In our analysis we preferred the use of the first method, since the impact of the scaling of the filament velocity isn't particularly relevant to our simulation given the rather limited timescales used. The final form of our equations is thus:

$$\frac{dn}{dt} = \frac{n\phi}{L_{\parallel}} - \frac{n - n_0}{L_{\parallel}} + ng\frac{\partial \phi}{\partial y} - g\frac{\partial n}{\partial y} + D_n \nabla_{\perp}^2 n$$
$$\frac{d\Omega}{dt} = \frac{\phi}{L_{\parallel}} - \frac{g}{n}\frac{\partial n}{\partial y} + \mu_i \nabla_{\perp}^2 \Omega$$
$$\Omega = \nabla^2{}_{\perp}\Phi$$

The term γ has been renominated g to better denote his work in the evolution of the filaments, as it acts similarly to a gravitational acceleration (The blob has negligile mass but is susceptible to electromagnetic forces). Here, the variables and constants are defined as:

- n: Plasma density
- ϕ: Electrostatic potential
- Ω: Vorticity ($\Omega = \nabla_{\perp}^2 \phi$)
- $L_{\parallel}$: Parallel connection length
- n_0: Background density
- g: Magnetic curvature drive term
- D_n: Diffusion coefficient
- μ_i: Ion viscosity
- $\nabla_{\perp}^2$: Perpendicular Laplacian
- d/dt: Total differential in time

Total differential: The total differential is central to the understanding of the 2D equations in the context of plasma blobs dynamics. The equations describe the evolution of density (n) and vorticity (Ω) over time and the total differential helps to express how the variables change as a result of both local and non-local contributions.

In general, the total differential of a quantity f (e.g., n or Ω) with respect to time is written as:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f$$

where:

- $\frac{\partial f}{\partial t}$ is the partial time derivative, describing the local rate of change of f.
- $\mathbf{v} \cdot \nabla f$ represents the advective term, accounting for the transport of f due to the flow velocity of the plasma $\mathbf{v}$.

The equations involve the total differentials $\frac{dn}{dt}$ and $\frac{d\Omega}{dt}$, expressed as:

$$\frac{dn}{dt} = \frac{\partial n}{\partial t} + \mathbf{v}_E \cdot \nabla n$$
$$\frac{d\Omega}{dt} = \frac{\partial \Omega}{\partial t} + \mathbf{v}_E \cdot \nabla \Omega$$

where $\mathbf{v}_E = \mathbf{E} \times \mathbf{B}/B^2$ is the $E \times B$ drift velocity that describes the advection of plasma by the electrostatic field $\mathbf{E} = -\nabla \phi$.

The total differential is essential for our simulations because it allows for:

1) **the separation of local and advective changes:** In the equations, $\frac{\partial n}{\partial t}$ and $\frac{\partial \Omega}{\partial t}$ capture the local time-dependent behavior of density and vorticity, while the advective terms, $\mathbf{v}_E \cdot \nabla n$ and $\mathbf{v}_E \cdot \nabla \Omega$, represent how these quantities are transported spatially by the $E \times B$ flow.
2) **the coupling of effects across the plasma:** By including the advective term, the total differential allows us to make sure that the evolution of density and vorticity depends not only on their local behavior but also on gradients across the plasma in order to capture the influence of non-strictly-local processes.
3) **the unification of the framework for physical effects:** The total differential provides a compact form that incorporates all these different physical processes, such as advection, dissipation, and curvature effects, into a single mathematical framework that ensures that the equations account for how plasma filaments evolve in both time and space.
4) **the facilitation of numerical simulation:** Last but not least, the total differential is particularly useful for tracking the evolution of quantities like n and Ω in simulations as it breaks down the changes into discrete, easily computable components (local updates and transport effects).

In the context of our equation set, the total differential highlights that the evolution of density and vorticity results from:

- Local effects, such as curvature-driven forces (γ) and dissipation (D_n, μ_i).
- Transport due to $E \times B$ flows, which redistribute density and vorticity across the perpendicular plane.

This combination ensures that the equations accurately capture the behavior of plasma filaments in the scrape-off layer, where advection and local dynamics are tightly coupled.

The following is the derivation of the total differential in the context of the parameters of our equations and assumptions:

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \underline{\nabla}_E \cdot \underline{\nabla} = \frac{\partial}{\partial t} - \frac{1}{|B|}\left(E_y \frac{\partial}{\partial x} - E_x \frac{\partial}{\partial y}\right)$$

$$\underline{\nabla}_E = \frac{\hat{b} \times \nabla\phi}{|B|} \quad \hat{b} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \nabla\phi = \begin{pmatrix} E_x \\ E_y \\ 0 \end{pmatrix}$$

$$|B|\underline{\nabla}_E = \hat{b} \times \nabla\phi = \begin{pmatrix} -E_y \\ E_x \\ 0 \end{pmatrix}$$

$$\underline{\nabla}_E \cdot \underline{\nabla} = \frac{1}{|B|}\left(-E_y \frac{\partial}{\partial x} + E_x \frac{\partial}{\partial y}\right)$$

As we will discuss later, we initialized the Electric field as having no E_x component, thus greatly simplifying the computational load.

Derivation of the constants and natural units: Our equations describe the evolution of density (n) and vorticity (Ω) in the context of the sheath dissipation closure. The constants appearing in these equations arise from the basic assumptions of the model and the physical processes governing plasma dynamics. We can thus outline the steps to derive these constants, that depend on:

- The parallel connection length $L_{\|}$, representing the distance between the sheath boundaries.
- The curvature of the magnetic field, parameterized by γ or g.
- Dissipative effects, like perpendicular diffusion (D_n) and viscosity (μ_i).

Our differential equation set is derived by integrating the 3D governing equations for density and vorticity along the parallel direction and applying the following assumptions:

1) The filament is sheath-connected, meaning parallel gradients in n and ϕ are negligible.
2) Sheath boundary conditions govern the parallel current ($J_{\|}$) In our chosen plane.
3) Dissipation occurs mainly in the perpendicular plane and is modeled by D_n (density diffusion) and μ_i (viscosity damping).
4) Magnetic curvature effects (γ or g) drive polarization currents and affect density and vorticity.

The term $\frac{\phi}{L_{\|}}$ accounts for parallel electric fields that are derived from the sheath boundary conditions. The constant $L_{\|}$, which represents the characteristic parallel lenght scale, appears due to the averaging of parallel effects in the sheath dissipation model because parallel currents are closed through the sheath.

The parameter γ or g encapsulates the effects of magnetic field curvature and gradients. It arises from the diamagnetic current density term, which depends on the magnetic field. For a slab geometry, g is defined as:

$$g = \frac{2\rho_s}{R_c}$$

where:

- $\rho_s = c_s/\Omega_i$ is the hybrid gyroradius.
- R_c is the radius of curvature of the magnetic field.

This parameter quantifies the strength of curvature effects, which drive interchange instabilities and influence vorticity and density both.

The perpendicular diffusion of density is represented by the term $D_n \nabla_{\perp}^2 n$. The constant D_n depends on collisional processes and is given by:

$$D_n = \frac{\nu_{ei}}{\Omega_i}$$

where:

- ν_{ei} is the electron-ion collision frequency.
- Ω_i is the ion gyrofrequency.

The viscous dissipation of vorticity is governed by the term $\mu_i \nabla_{\perp}^2 \Omega$. The perpendicular viscosity constant μ_i is given by:

$$\mu_i = \frac{\mu}{n_0}$$

where:

- μ is the dynamic viscosity of the plasma.
- n_0 is the background density.

The term $\frac{n-n_0}{L_\parallel}$ comes from the assumption that the density perturbation relaxes toward the uniform background density n_0. This is reflected in the shape of the initial blob in our simulation, that assumes a gaussian profile, tapering off towards the edge and approaching the background density Summarizing, the constants are:

$$n_0 = 0.8 \times 10^{13} cm^{-3}$$
$$g = \frac{2q_s}{R_c} = 2.44 \times 10^{-3}$$
$$\mu_i = 0.04$$
$$L_\parallel = 5467.5$$
$$D_n = 0.0015$$

These constants are either provided directly by the reference paper or calculated using its informations, as it reports data obtained directly from the Mega Ampere Spherical Tokamak (MAST) [MF11]. These constants use their natural units derived from the Bohm normalization to ease computational time:

- Time and length scales are normalized to the ion gyro-frequency ($\Omega_i = \frac{eB}{m_i}$) and the hybrid gyro-radius ($q_s = \frac{c_s}{\Omega_i}$), respectively.
- The electrostatic potential (ϕ) is normalized to $\frac{T_e}{e}$, where T_e is the electron temperature and e is the elementary charge.
- The vorticity Ω is normalized to $\frac{T_e}{eq_s^2}$.
- Plasma density is normalized to a characteristic background SOL density (n_{SOL}).

Computational methods

After analyzing the theoretical implications of our equations and assumptions and calculating the constants needed for the numerical implementation, we began the search for computational methods that could aid us in the creation of the simulation [Owe].

Iterative methods for linear systems: Iterative methods provide a practical alternative to direct methods for solving linear systems of equations of the form $A \cdot \mathbf{x} = \mathbf{b}$. These methods are particularly advantageous when the matrix A is large and sparse, as they avoid the computational expense and memory usage of direct matrix inversion. The main idea is to approximate A with a simpler matrix B such that:

$$A \approx B, \quad \text{where } A = B + S$$

Here, S represents a residual matrix, accounting for the difference between A and B. Using this method, the system can be solved iteratively, starting from an initial guess $\mathbf{x}^{(0)}$ and refining the solution incrementally.

The iterative process is governed by the equation:

$$\mathbf{x}^{(i+1)} = B^{-1}\mathbf{b} - G\mathbf{x}^{(i)}, \quad \text{where } G = B^{-1}S$$

The matrix G, known as the update matrix, determines how the solution evolves in each iteration. The convergence of the iterative method depends on the properties of this matrix, specifically, the spectral radius $\rho(G)$, defined as the largest absolute value of the eigenvalues of G, must satisfy the condition:

$$\rho(G) < 1$$

This makes sure that the error diminishes with each iteration. The propagation can be expressed as:

$$\varepsilon^{(i+1)} = -G\varepsilon^{(i)}$$

where $\varepsilon^{(i)} = \mathbf{x}^{(i)} - \mathbf{x}$ is the error at the i-th iteration.

Iterative methods offer several advantages over direct methods, for example in using large matrices, as they are computationally less demanding. Direct methods typically require $\mathcal{O}(N^3)$ operations, whereas iterative methods normally scale as $\mathcal{O}(N^2 \cdot N_{\text{iter}})$, where N_{iter} is the number of iterations required to arrive at convergence. This makes these methods particularly well-suited for sparse matrices, like the one we need to analyze for our simulation. Iterative methods are also less susceptible to rounding errors because they rely on successive refinements rather than a long, single computation involving the entire matrix. Finally, an iterative method must be terminated based on a predefined stopping condition, which typically involves assessing the convergence of the solution through the relative change, which is the fractional change in successive approximations of the solution:

$$\varepsilon^{(i)} = \frac{\|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\|}{\|\mathbf{x}^{(i)}\|}$$

The iteration stops when $\varepsilon^{(i)}$ falls below a specified tolerance level. The efficiency of these kinds of methods depends heavily on the choice of the approximation matrix B. A common approach is to decompose A into three components:

$$A = D + T_L + T_U$$

where D is the diagonal matrix containing the diagonal elements of A, T_L contains the elements below the diagonal, and T_U contains the elements above the diagonal. This decomposition allows flexibility in defining B and consequently the update matrix G.

Jacobi Method: The Iterative method we chose for the simulation is called Jacobi Method, as it is one of the simplest but still reliable. Here, the approximation matrix B is chosen as the diagonal matrix D, leading to:

$$B = D, \quad S = T_L + T_U$$

The iterative formula becomes:

$$\mathbf{x}^{(i+1)} = D^{-1}\mathbf{b} - D^{-1}(T_L + T_U)\mathbf{x}^{(i)}$$

This method is computationally straightforward since D^{-1} is easy to compute (as D is a diagonal matrix). The initial guess for the solution is:

$$\mathbf{x}^{(0)} = D^{-1}\mathbf{b}$$

The Jacobi method is guaranteed to converge if the matrix A is strictly diagonally dominant, meaning:

$$|A_{ii}| > \sum_{j \neq i} |A_{ij}| \quad \forall i$$

While the method may also converge for non-diagonally dominant matrices, this depends on the eigenvalues of the associated update matrix G.

Sparse systems (such as the one we need to analyze), where most of the elements of the matrix are zero, are particularly well-suited to this method because these systems can very often be manipulated into a diagonally dominant form using row pivoting.

Elliptic PDE: To tackle the elliptic partial differential equations (PDEs) present in our main set of equations, we employed a finite difference method, the fundamental principle of which involves discretizing the domain into a 2D lattice (grid of points) and approximating derivatives using finite difference techniques. For starters, we can consider the baseline elliptic PDE (Laplace's equation):

$$\nabla^2 u(x, y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Via discretizing the problem, we can construct a system of simultaneous equations that are represented in matrix form as:

$$A \cdot \mathbf{u} = \mathbf{b}$$

where $\mathbf{u}$ is the vector of unknowns, A is the coefficient matrix derived from the approximation, and $\mathbf{b}$ encodes the boundary conditions.

In order to construct the grid we had to impose some limitations: The lattice is defined within a rectangular domain $0 \leq x \leq L_x$ and $0 \leq y \leq L_y$. Let N_x and N_y denote the number of grid points in the x- and y-directions respectively, including the boundaries. The spacing in each direction is a uniform h, such that:

$$x_i = ih, \quad i = 0, 1, \ldots, N_x - 1, \quad L_x = (N_x - 1)h$$

$$y_j = jh, \quad j = 0, 1, \ldots, N_y - 1, \quad L_y = (N_y - 1)h$$

The method we used involves the finite difference approximations of the second derivatives:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j} \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}$$

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{i,j} \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2}$$

Combining these, the finite difference approximation for the Laplacian becomes:

$$\nabla^2 u_{i,j} \approx \frac{u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1} - 4u_{i,j}}{h^2}$$

which represents the coupling between a grid point and its four immediate neighbors in the x- and y-directions.

The discretized Laplacian can be visualized as a "finite difference stencil" or pictorial operator:

$$\nabla^2 u_{i,j} = \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} u_{i,j}$$

Solving the Poisson equation: The finite difference (FD) matrix equation derived from discretizing elliptic PDEs, such as the previously mentioned Laplace equation, has to be solved iteratively through the Jacobi method. The Jacobi update formula for the k-th iteration is:

$$\mathbf{u}^{(k+1)} = D^{-1}\mathbf{b} - D^{-1}(T_L + T_U)\mathbf{u}^{(k)}$$

In this specific case where we have our Laplacian, the inverse of D simplifies to:

$$D^{-1} = -\frac{1}{4}I$$

Thus, the zero order guess for the solution is:

$$\mathbf{u}^{(0)} = -\frac{1}{4}\mathbf{b}$$

and the iterative update becomes:

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(0)} + \frac{1}{4}\left[(T_L + T_U)\mathbf{u}^{(k)}\right]$$

The update rule can be expressed pictorially as:

$$u_{i,j}^{(k+1)} = u_{i,j}^{(0)} + \frac{1}{4}\begin{bmatrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{bmatrix} u_{i,j}^{(k)}$$

In our case, we used the extended case of the **Poisson equation** in order to calculate $u^{(k+1)}$

$$u_{i,j}^{(k+1)} = u_{i,j}^{(0)} + \frac{1}{4}\begin{bmatrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{bmatrix} u_{i,j}^{(k)} - \frac{h^2}{4}\rho_{i,j}$$

As stated in the previous segment, this formula averages the values of the four nearest neighbors for each internal grid point during each iteration.

The boundary conditions are also implicitly handled by including their contributions in $\mathbf{b}$. For example, when updating an internal grid point near the edge, the values from the boundary automatically contribute via the FD stencil. The algorithm for solving the FD system then sets initial guesses for all grid points, including boundary values, applies the pictorial operator to each internal grid point to compute $u_{i,j}^{(k+1)}$ and finally repeats the process until the solution converges within a predefined tolerance.

Runge-Kutta method: The Runge-Kutta (RK) methods are a family of explicit, single-step approaches widely used for solving ordinary differential equations (ODEs). These methods work by generalizing the multi-step concept via the use of multiple estimates of the derivative that are computed at different points within the interval, to determine an average. The family of RK methods includes members of increasing order and we used the fourth order one (RK4) which is a particularly popular variant due to its balance between accuracy and computational ease. It obtains fourth-order accuracy by combining four gradient evaluations each timestep, that are carefully weighted to ensure that their Taylor series expansion matches up to terms of $\mathcal{O}(\Delta t^4)$.

The RK4 process can be expressed as:

$$\begin{aligned} f_a &= f(t_n, u_n) \\ f_b &= f\left(t_n + \frac{\Delta t}{2}, u_n + \frac{f_a \Delta t}{2}\right) \\ f_c &= f\left(t_n + \frac{\Delta t}{2}, u_n + \frac{f_b \Delta t}{2}\right) \\ f_d &= f\left(t_n + \Delta t, u_n + f_c \Delta t\right) \end{aligned}$$

where f_a, f_b, f_c, f_d represent the function evaluations at different points within the timestep.

The final solution is thus updated using a weighted average:

$$u_{n+1} = u_n + \frac{\Delta t}{6}\left(f_a + 2f_b + 2f_c + f_d\right)$$

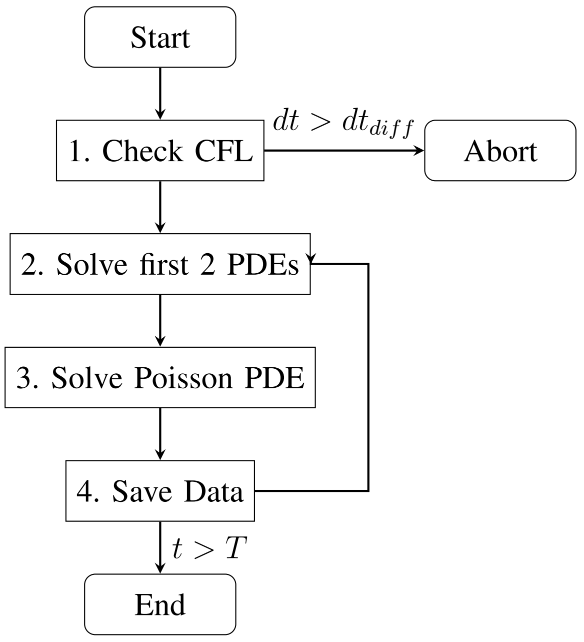
The RK4 method is particularly useful to our cause because:

- It does not require storage of previous timesteps, making it computationally less heavy.
- It avoids the starting issues that are present with other multi-step methods, as each step is computed independently of every other.
- The method can be adapted for use with larger step sizes while still maintaining acceptable accuracy, due to its fourth-order nature.

The stages of RK4 progressively refine the gradient guess, starting with an initial Euler estimate and then iterating through midpoint and endpoint evaluations. The weighting scheme then ensures that intermediate errors cancel out effectively.

Effective implementation

We have implemented a solver using periodic boundary condition, a grid domain of $L_x \times L_y = 20 \times 10$ with a resolution of 200×200, a time domain $T = 0.1$ and $N_t = 10^4$ time-steps. The solver takes advantage of all the algorithms mentioned above. This next diagram summarizes the steps of the algorithm:



Equations and corresponding flowchart steps

Check CFL: The diffusion equation's CFL conditions ensure stability in the time-stepping scheme, whose condition is given by:

$$dt_{diff\_n} = \frac{dx^2}{4D}, \quad dt_{diff\_\Omega} = \frac{dx^2}{4\mu}$$

Check: $dt \leq dt_{diff\_n}$ and $dt \leq dt_{diff\_\Omega}$

If the time step exceeds the CFL condition, the simulation halts with an error.

Solve first 2 PDEs: The spatial and temporal derivatives used in the code are:

- Laplace operator for diffusion $\nabla^2 n_{i,j}$:

$$\frac{n_{i+1,j} - 2n_{i,j} + n_{i-1,j}}{dx^2} + \frac{n_{i,j+1} - 2n_{i,j} + n_{i,j-1}}{dy^2}$$

- Laplace operator for diffusion $\nabla^2 \phi_{i,j}$:

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{dx^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{dy^2}$$

- Central difference for derivatives:

$$\left.\frac{\partial n}{\partial y}\right|_{i,j} = \frac{n_{i,j+1} - n_{i,j-1}}{2dy}$$
$$\left.\frac{\partial \phi}{\partial y}\right|_{i,j} = \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2dy}$$

- Drag term, used for implementing the effect of the total differential

$$\text{dg}(f)_{i,j} = \left.\frac{-\frac{\partial \phi}{\partial y} \cdot \frac{\partial f}{\partial x} + \frac{\partial \phi}{\partial x} \cdot \frac{\partial f}{\partial y}}{B}\right|_{i,j}$$

The 4th-order Runge-Kutta method is used to numerically integrate the equations for n and Ω:

$$f(n_{i,j}, \Omega_{i,j}) = \left.\frac{dn}{dt}\right|_{i,j} = \text{dg}(n)_{i,j} - \frac{n_{i,j} \cdot \phi_{i,j}}{L_\parallel} +$$
$$+ g \cdot n_{i,j} \cdot \left.\frac{\partial \phi}{\partial y}\right|_{i,j} - g \cdot \left.\frac{\partial n}{\partial y}\right|_{i,j} + D \cdot \nabla^2 n_{i,j}$$
$$g(n_{i,j}, \Omega_{i,j}) = \left.\frac{d\Omega}{dt}\right|_{i,j} =$$
$$\text{dg}(\Omega)_{i,j} + \frac{\phi_{i,j}}{L_\parallel} + \frac{n_{i,j}}{g} \cdot \left.\frac{\partial n}{\partial y}\right|_{i,j} + \mu \cdot \nabla^2 \Omega_{i,j}$$

Following these steps:

- First step:
$k_1^n = f(n_{i,j}, \Omega_{i,j}), \quad k_1^\Omega = g(n_{i,j}, \Omega_{i,j})$
- Second step:
$$k_2^n = f\left(n_{i,j} + \frac{dt}{2} \cdot k_1^n, \Omega_{i,j} + \frac{dt}{2} \cdot k_1^\Omega\right)$$
$$k_2^\Omega = g\left(n_{i,j} + \frac{dt}{2} \cdot k_1^n, \Omega_{i,j} + \frac{dt}{2} \cdot k_1^\Omega\right)$$
- Third step:
$$k_3^n = f\left(n_{i,j} + \frac{dt}{2} \cdot k_2^n, \Omega_{i,j} + \frac{dt}{2} \cdot k_2^\Omega\right)$$
$$k_3^\Omega = g\left(n_{i,j} + \frac{dt}{2} \cdot k_2^n, \Omega_{i,j} + \frac{dt}{2} \cdot k_2^\Omega\right)$$
- Fourth step:
$$k_4^n = f\left(n_{i,j} + dt \cdot k_3^n, \Omega_{i,j} + dt \cdot k_3^\Omega\right)$$
$$k_4^\Omega = g\left(n_{i,j} + dt \cdot k_3^n, \Omega_{i,j} + dt \cdot k_3^\Omega\right)$$

The updated values are then written as:

$$n_{i,j}^{n+1} = n_{i,j}^{n} + \frac{dt}{6}\left(k_1^n + 2k_2^n + 2k_3^n + k_4^n\right)$$

$$\Omega_{i,j}^{n+1} = \Omega_{i,j}^{n} + \frac{dt}{6}\left(k_1^\Omega + 2k_2^\Omega + 2k_3^\Omega + k_4^\Omega\right)$$

Solve Poisson PDE: The potential ϕ is updated iteratively using the Jacobi method:

$$\phi_{i,j}^{(k+1)} = \frac{\frac{\phi_{i+1,j}^k + \phi_{i-1,j}^k}{dx^2} + \frac{\phi_{i,j+1}^k + \phi_{i,j-1}^k}{dy^2} + \Omega_{i,j}}{2\left(\frac{1}{dx^2} + \frac{1}{dy^2}\right)}.$$

until $\| \frac{\Phi^{k+1} - \Phi^k}{\Phi^k} \|_\infty < \epsilon_{tol}$[3]

Save data: The values n, ϕ, and Ω are finally saved for the current t.

Initialization of simulation values

The simulation initializes the density $n_{i,j}$, the potential $\phi_{i,j}$, and the vorticity $\Omega_{i,j}$ on a periodic grid. The initialization follows these steps:

1) Density : The density $n_{i,j}$ is set to a Gaussian distribution of height 2 over a background of 1 (n_{SOL}) centered at $(x_0, y_0) = (L_x/2, L_y/2)$ with standard deviations $\sigma_x = L_x/20$ and $\sigma_y = L_y/10$. For each grid point (i, j), the formula is:

$$n_{i,j} = 1 + \exp\left(-\frac{(x_i - x_0)^2}{2\sigma_x^2} - \frac{(y_j - y_0)^2}{2\sigma_y^2}\right)$$

where $x_i = i \cdot \Delta x$ and $y_j = j \cdot \Delta y$. Periodic boundary conditions are applied using modular indexing to ensure consistency at the edges of the grid.

2) Potential: The potential $\phi_{i,j}$ is initialized under the assumption that the vorticity is minimal at $t = 0$, leading to $\frac{\partial \Omega}{\partial t} = 0$. For each grid point (i, j), the formula is:

$$\phi_{i,j} = -\frac{L_k g}{n_{i,j}} \cdot \frac{n_{i,j+1} - n_{i,j-1}}{2dy}$$

3) Vorticity: The vorticity $\Omega_{i,j}$ is computed from the potential $\phi_{i,j}$ using the Laplacian:

$$\Omega_{i,j} = \nabla^2 \phi_{i,j}$$

where the vorticity is evaluated numerically as explained in the previous section.

[3] ϵ_{tol} is estimated as the biggest truncation error, so it is dependent on the resolution of the grid.

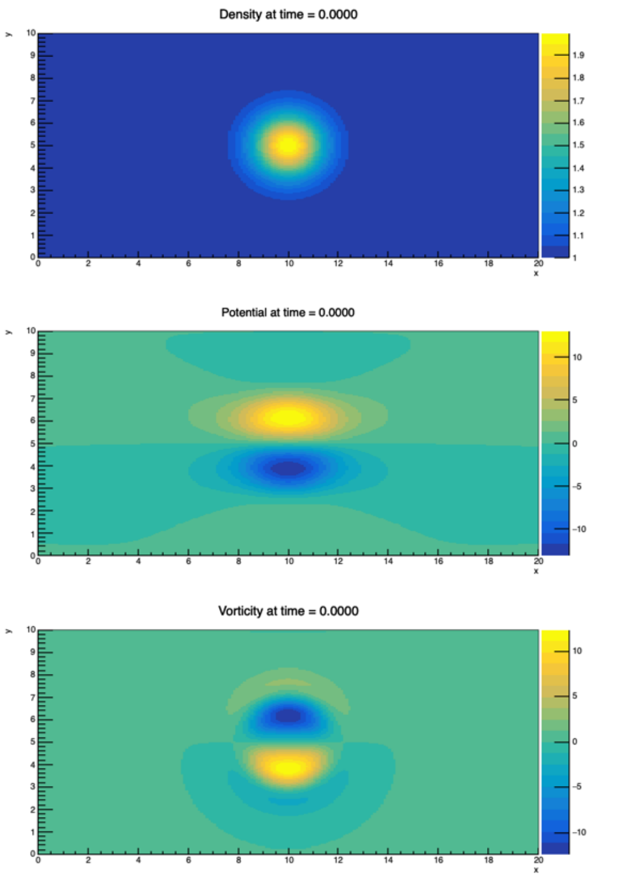


Figure 1: Initialization of variables under a constant upward pointing magnetic field and E_y. The upper image shows density n, the middle one shows the potential Φ and the last one shows the vorticity Ω. We use a color-coded graduate scale whose values are shown on the right graduated bar.

In our reference paper [Nie14], the initialisation is done by running the code until reaching a stability point and then recording the evolution values. To verify that the potential tends to become the one we initialized, we can try to add some instability, for example, by rotating 60° counterclockwise the initial electric field, and then integrating to find the electric potential at t=0 (see Figure 2 and 3 for reference). Comparing the animation 4 and 5 we can see that the potential evolves very fast to the one we set, thus our way of initializing the potential (by assuming that in the beginning $\frac{\partial \Omega}{\partial t} = 0$) is indeed a correct choice, as it's a point of stability.

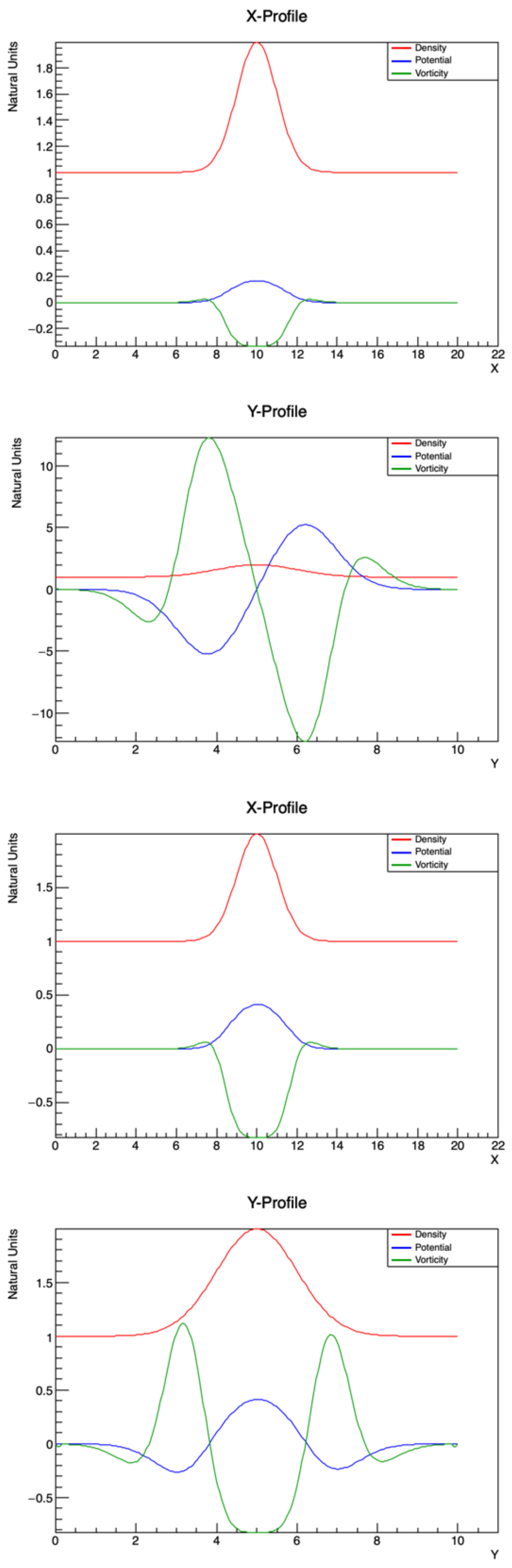


Figure 2: The initialized profiles of $n_{i,j}$, $\phi_{i,j}$, and $\Omega_{i,j}$ are plotted to verify coherence with the theoretical model before starting the simulation. The first two graphs refer to the main simulation, the last two refer to Figure 3, a secondary simulation in which the initial electric field was rotated 60° counterclockwise

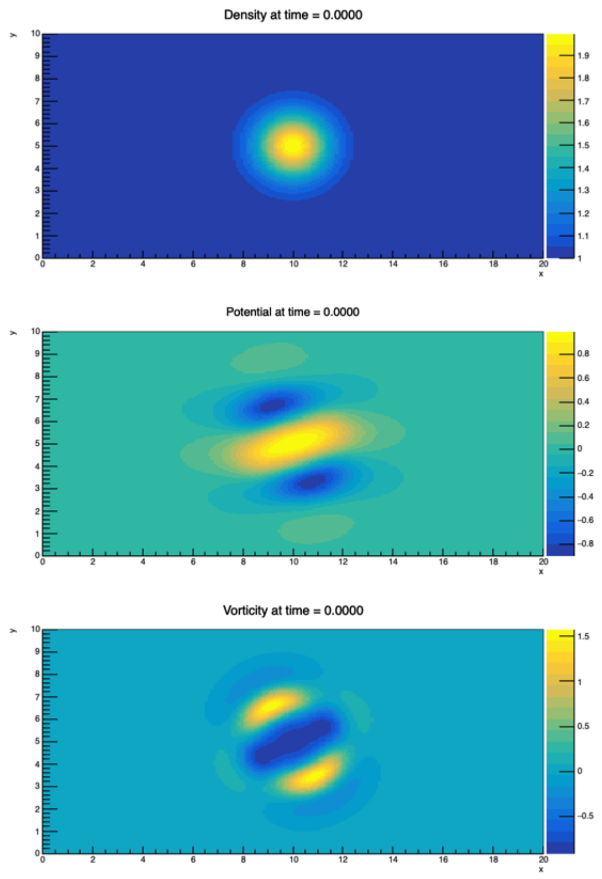


Figure 3: Initialisation after a rotation of 60° of the initial electric field. The upper image shows density n, the middle one shows the potential Φ and the last one shows the vorticity Ω. We use a color-coded graduate scale whose values are shown on the right graduated bar.

IV. Presentation of findings

THIS section highlights the outcomes of our computational simulation on the dynamics of plasma blobs in the simplified two-dimensional geometry. Our primary objective was to achieve a sound visualization of the evolution of a filament inside the SOL, and we obtained such a result using our theoretical and computational approaches.

Successful simulation framework

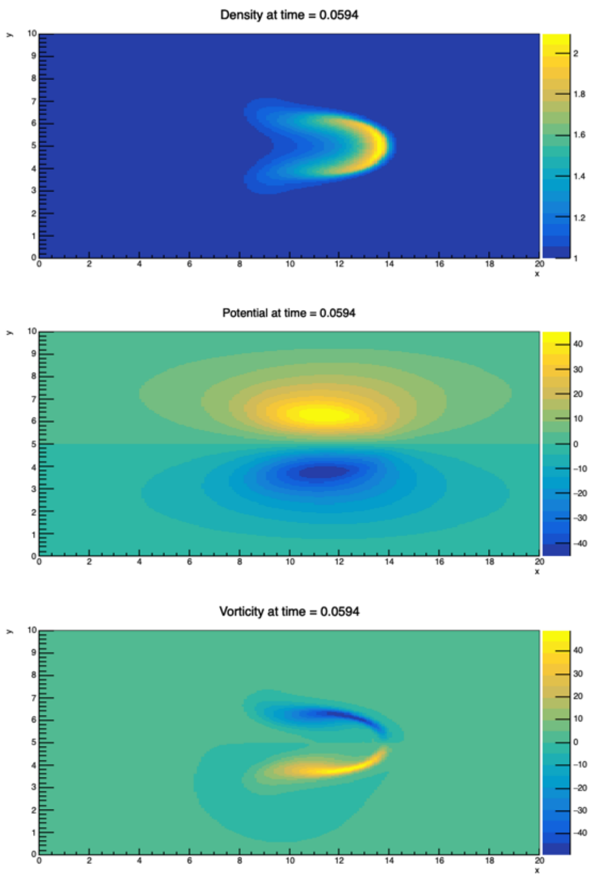


Figure 4: See Animation.
Main simulation result at frame 90/150: Single gaussian blob in a constant upward pointing magnetic field B and a downward pointing electric field E_y. The upper figure represents the evolution of plasma density, as highlighted by the bar opposite to the y axys, the middle figure represents the electric potential and the lower figure the vorticity. The simulation has all the equation terms, including the diffusion ones. Every axys is in its own natural units to better show the evolution of the characteristics.

After our main simulation was completed, we decided to further investigate other possibilities for the initial conditions, such as smaller and bigger blobs and even a different initial electric field. These secundary endeavors allowed us to explore the issue in a more in-depth and less streamlined way. We also further ensured the solidity and reliability of our main code.

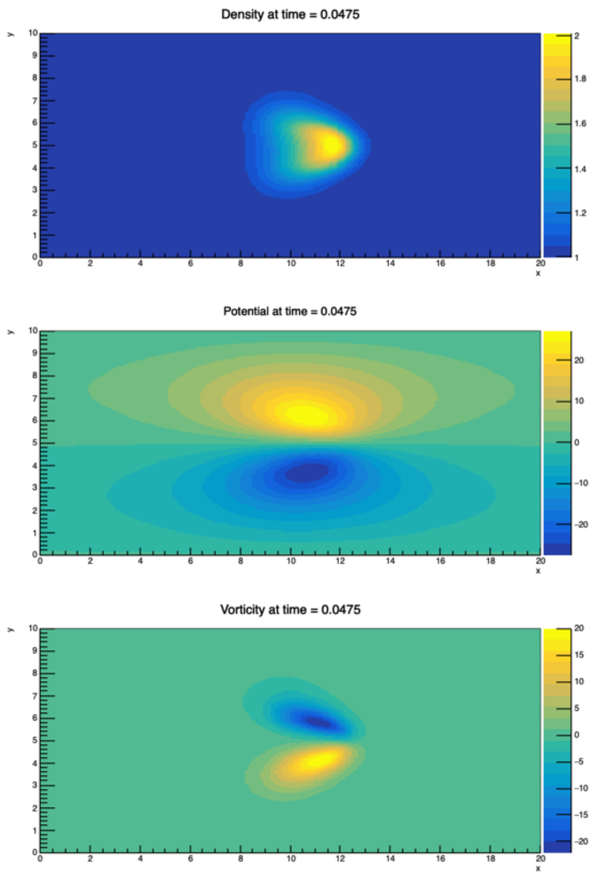


Figure 5: See Animation.
Further simulation result at frame 70/150: Single gaussian blob in a constant upward pointing magnetic field B and a rotated (by $\pi/3$) electric field E_y. The upper figure represents the evolution of plasma density, as highlighted by the bar opposite to the y axys, the middle figure represents the electric potential and the lower figure the vorticity. The simulation has all the equation terms, including the diffusion ones. Every axys is in its own natural units to better show the evolution of the characteristics. This was an intriguing perspective to analize to show a different configuration of the initial conditions.

The simulation produces the expected theoretical behaviour for a gaussian blob. We can notice how the center of density progressively migrates towards the x direction, in accordance with the predicted $E \times B$ motion. As such, we can focus our attention to the aspects that interest our research:

- **Radial propagation:** All filaments exhibited a radial outward motion driven by $\mathbf{E} \times \mathbf{B}$ advection, with observed velocities in alignment with predictions from our theory.
- **Dissipative effects:** The role of perpendicular viscosity and diffusion was qualitatively assessed, thus showing that their impact on filament dissipation and shape retention was rather limited. Both diffusion terms accounted for very minor changes and overall small effect.
- **Electrostatic potential dynamics:** The distribution of electrostatic potential ϕ evolved consistently with expectations, showing a clear coupling relation to density and vorticity, just as expected from the governing equations.
- **Blob deformation:** Under the influence of magnetic curvature and dissipative forces, filaments experienced elongation and deformation, mostly resulting from the difference in pull between the center of density and the periferal region.

V. Discussion of Results

The results obtained from our computational simulations offer interesting insights into the rich dynamics of plasma filaments in the SOL of tokamak devices.

Limitations and sources of error

Despite the robustness of our simulation framework, certain limitations have influenced the interpretation and generalization of results:

- **Dimensionality Constraints**: The use of a two-dimensional model greatly simplifies the computational process but causes the exclusion of critical three-dimensional effects, in particular those associated with parallel transport along magnetic field lines. If we were to start with different 2D closure assumptions we could have gotten slightly (or even greatly) different simulation outputs.
- **Assumption of electrostatic conditions**: By assuming negligible magnetic perturbations, the model restricts itself to particularly idealized kinds of plasma. This may not fully capture dynamics in highly turbulent regimes.
- **Simplified properties of plasma**: The assumption of isothermal electrons and negligible ion temperature gradients introduces unavoidable inaccuracies.
- **Boundary conditions**: Since periodic boundary conditions are an artifact of our code to simulate a continuous environment, the eventual presence of density near the edge of our simulation's grid can bring to results that are not properly physical and, while still numerically and computationally stable, could bring to wrong real-world implications.

Possible implications for magnetic confinement fusion

The findings from our study, whilst certainly self contained to the scope of the project, can also carry implications and real world application for the design of magnetic confinement fusion devices:

- **ELM mitigation strategies**: A good understanding of blob dynamics can infer new approaches to control edge-localized modes, potentially reducing the heat flux on components that are in contact with the plasma to prolong their lifespan.
- **Enhanced plasma confinement**: our insights into dissipation mechanisms could guide the optimization of tokamak geometries and structures and allow us to find operational parameters to improve the efficiency of the confinement.
- **Validation of modeling techniques**: The consistency between our simulation results and the theoretical predictions could allow the use of sheath dissipation closure in describing localized SOL dynamics, thus establishing a foundation for future computational studies, even in a 3D environment

Possible future directions

Several avenues for future research could emerge from our study:

- **3D Simulation development**: The most obvious would be simply extending the model to include parallel dynamics and toroidal geometry, which would provide a more comprehensive understanding of filament behavior, especially because it would be under more realistic tokamak conditions.
- **Incorporation of kinetic effects**: Including non-fluid effects, such as particle drifts and kinetic instabilities, would enhance the model's applicability to a wider range of plasma parameters. This is however the most difficult possible future development, since it would require a complete overhaul of the base model.
- **Experimental validation**: Comparing simulation results with experimental data from operating tokamaks would certainly enhance our confidence in the model and provide further opportunities for refinement.

VI. Conclusions

OUR project successfully achieved the initial aim of developing a basic turbulent filament solver by leveraging a simplified 2D geometry and solving drift-reduced plasma fluid equations. We were able to accurately simulate the behavior of individual blobs under the influence of magnetic curvature and dissipative forces. Our results demonstrate the theoretical expectations of filament motion, including and not limited to radial propagation due to $E \times B$ advection, deformation influenced by magnetic curvature, and the limited role of dissipative effects. Our findings also validated the feasibility of using a sheath dissipation closure for localized SOL dynamics and provided useful insights that could aid in the design of new magnetic confinement fusion devices. While our work achieved its stated goals, it also highlighted certain limitations that are intrinsec to the model's assumptions, like the simplification to 2D geometry, electrostatic conditions, and especially the isothermal electron properties. These self imposed constraints point to opportunities for further exploration that could include the extension of the model to three-dimensional simulations, the incorporation of kinetic effects, and the validation of results against experimental data from operational tokamaks like JET or ITER. The only real obstacle to the full success of this project is the lack of possible validations outside the multiple trials and the theoretical coherence of our simulations. Summarizing, this project tried to lay a solid enough foundation for further studies in the subject of plasma filament dynamics coadiuvated with clear pathways for advancing the understanding and control of turbulence in magnetic confinement fusion systems.

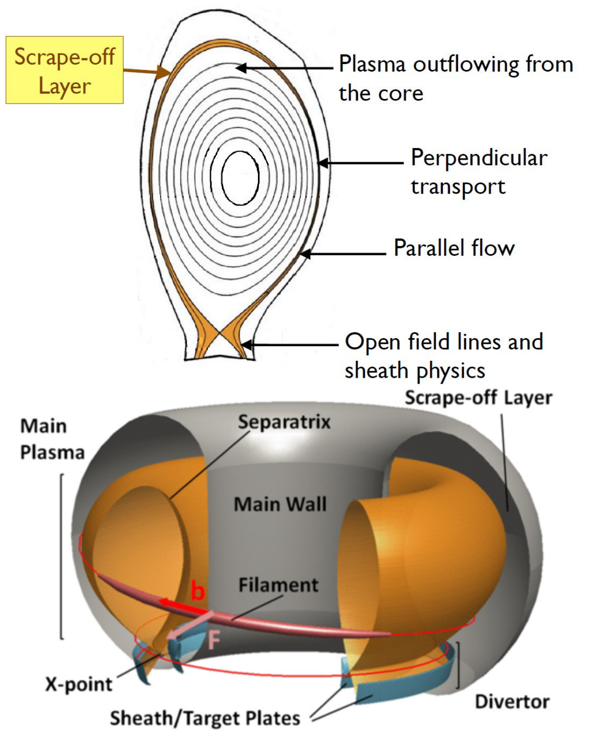


Figure 6: Schematics of a Tokamak, highlighting the location of the scrape-off-layer (SOL)

VII. Aknowledgements

We wish to thank Dr Kingham, Robert J. for the help and the guidance during the course of this project and Dr Andrew, Yasmin for the useful insights given during the Project's Viva.

Appendix

The code used for the simulation can be downloaded here.
Examples of further simulations performed:

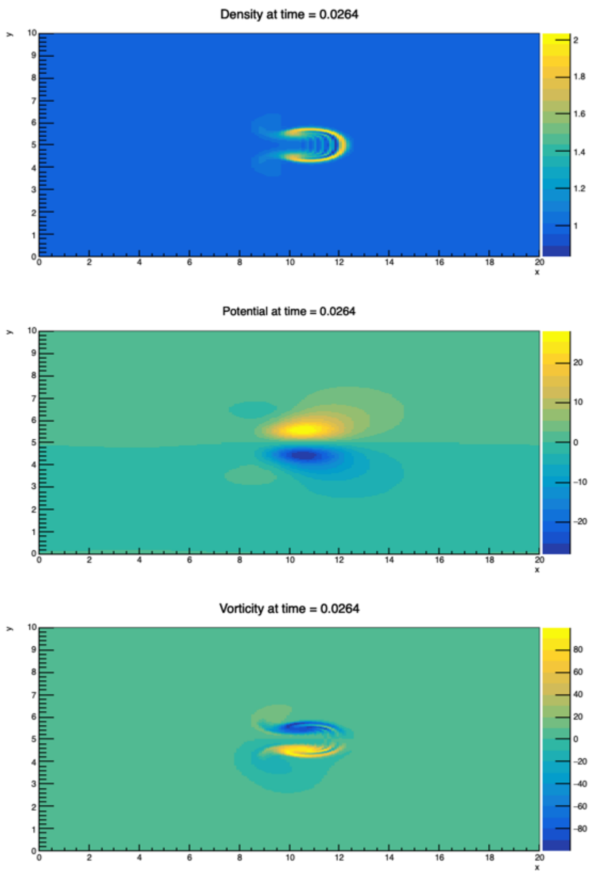


Figure 7: See Animation.
Initialisation after changing the parameters of the gaussian blob, essentially halving its extension. Frame 40/150. The upper image shows density n, the middle one shows the potential Φ and the last one shows the vorticity Ω. We use a color-coded graduate scale whose values are shown on the right graduated bar.

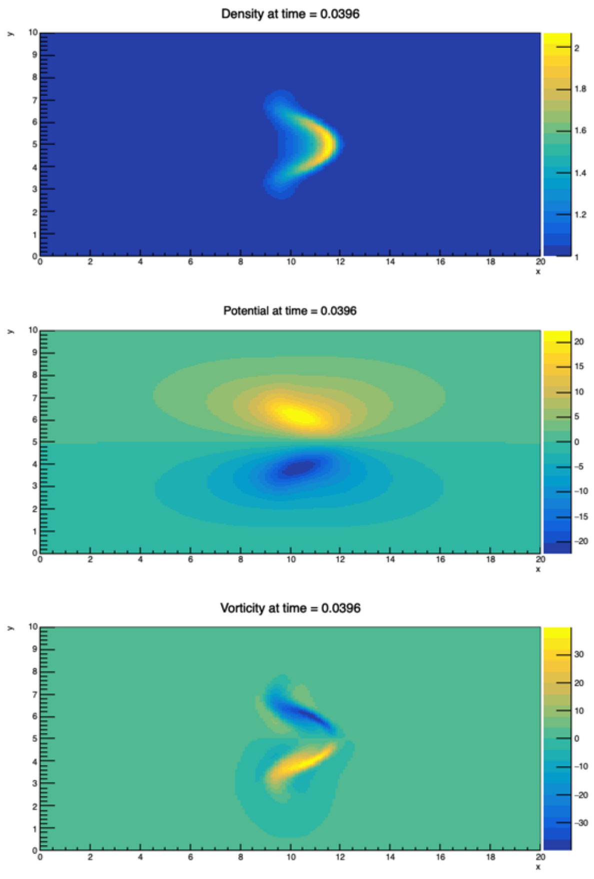


Figure 8: See Animation.
Initialisation after changing the parameters of the gaussian blob, creating a different, elliptic shape. Frame 40/150. The upper image shows density n, the middle one shows the potential Φ and the last one shows the vorticity Ω. We use a color-coded graduate scale whose values are shown on the right graduated bar.

BIBLIOGRAPHY

[DZ11] J. R. Myra D. A. D'Ippolito and S. J. Zweben. "Convective transport by intermittent blob-filaments: Comparison of theory and experiment". In: (2011). DOI: https://www.doi.org/10.1063/1.3594609.

[MF11] F Militello and W Fundamensk. "Multi-machine comparison of drift fluid dimensionless parameters. (English)". In: (2011). DOI: https://doi.org/10.1088/0741-3335/53/9/095002.

[Eas14] L. Easy. "Three dimensional simulations of plasma filaments in the scrape off layer: A comparison with models of reduced dimensionality". In: (2014). DOI: https://doi.org/10.1063/1.4904207.

[Nie14] L. Easy ; F. Militello; J. Omotani; B. Dudson; E. Havlíčková; P. Tamain; V. Naulin; A. H. Nielsen. "Three dimensional simulations of plasma filaments in the scrape off layer: A comparison with models of reduced dimensionality. (English)". In: (2014). DOI: https://doi.org/10.1063/1.4904207.

[Kra22] S. I. Krasheninnikov. "Fascinating physics at the edge of magnetic fusion devices". In: (2022). DOI: https://doi.org/10.1088/1361-6587/ac9b8e.

[Mil22] F. Militello. *Boundary Plasma Physics – An Accessible Guide to Transport, Detachment, and Divertor Design*. Springer Series on Atomic, Optical, and Plasma Physics (SSAOPP, volume 123). Springer, 2022. DOI: https://doi.org/10.1007/978-3-031-17339-4.

[Owe] Mark Scott; James Owen. "Notes from the Computational Physics course, Imperial College London, 2024-2025". URL: https://bb.imperial.ac.uk/ultra/courses/_42715_1/cl/outline.