Gyrofluid Simulations of Filaments in Tokamak Edge Plasmas

Mr. Adam Dempsey B.Sc.

A dissertation submitted in fulfilment of the
requirements for the award of
Doctor of Philosophy (Ph.D.)

presented to the

School of Physical Sciences
Dublin City University

Supervisors:
Prof. Miles Turner
Dr. Huw Leggate
Dr. Ben Dudson (LLNL, USA)

Jan 2025

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed .....

Adam Dempsey
Student ID:12354706
Date: 8th January, 2025

Contents

List of Figures

List of Tables

Abbreviations

BOUT++ BOUndary Turbulence in C++

ELM edge localised mode

FLR finite Larmor radius

GEM electromagnetic gyrofluid model

GMRES generalised minimal residual

GPI gas puff imaging

H-mode high confinement mode

LCFS last closed flux surface

MAST the Mega Ampere Spherical Tokamak

MCF magnetic confinement fusion

MHD magnetohydrodynamics

ODE ordinary differential equation

PID proportional–integral–derivative

RMS root-mean-square

SOL scrape-off-layer

VFP Vlasov-Focker-Plank

WENO Weighted Essentially Non-Oscillatory

Acknowledgements

Thanks are owed to a great many people, without whom this thesis would not have been possible. Firstly of course, to my primary supervisors Dr. Huw Leggate and Prof. Miles Turner go my thanks for their patience, guidance and support, throughout my studies. Secondly the York Plasma Institute and Fusion CDT programme were invaluable and a better start to my doctoral studies couldn't be wished for. My thanks go to all the staff at the YPI for their excellent programme. Particularly I would like to thank Dr. Ben Dudson for his supervision during my time there, he is a font of knowledge and without his supervision I would have been thoroughly lost.

Personally I would like to thank my fellow students, both at DCU and the YPI, for their fruitful discussions over coffee and many late nights in the office. Particularly I'd like to mention Aoife, David, and, Rhys, the most frequent, but I hope willing, participants in such discussions.

The support of my friends and family remained unwavering throughout my studies. They are all owed more thanks than I can find words here to express. My friends have stuck with me through a time when my focus was more on derivations and simulations than meetups and events. To a friend that always made things more manageable when they seemed insurmountable, Nicole, I can't thank you enough. I couldn't have done it without you. To the JAMTAMNs (you know who you all are!), thanks for all the good times, long may they continue. And thanks of course for the unfailing support. Finally, I would like to thank my parents John and Sharon who have always supported my interest in science and never failed to encourage me to pursue whatever path I chose.

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 and 2019–2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

We acknowledge the CINECA award under the ISCRA initiative, for the availability of high performance computing resources and support.

Abstract

Gyrofluid Simulations of Filaments in Tokamak Edge Plasmas

Adam Dempsey

Cross field transport in the edge of magnetically confined plasmas is known to be turbulent in nature, specifically the transport is composed largely of plasma filaments. Understanding and quantifying this transport is of key importance when considering particle and heat loads for future devices.

Filament velocity scaling laws, derived using linearised drift-fluid models, and reproduced through fluid simulations are of key importance when considering filament propagation. They are also used in statistical models of the edge that relate filamentary fluctuations to mean scrape-off-layer (SOL) profiles. These same velocity scaling laws are reproduced herein both computationally and theoretically with a deviation identified and explained in the limit of small filament widths. Filament simulations have been carried out using a gyrofluid model known as GEM (electromagnetic gyrofluid model), implemented in the BOUT++ (BOUndary Turbulence in C++) framework, both with the inclusion of finite Larmor radius (FLR) effects and also with the gyroaverage operators taken in the limit of small Larmor radius to resemble drift-fluid models. Simulations were carried out over a range of filament sizes and a simplified slab geometry was employed with parameters chosen to represent typical conditions in the edge of the Mega Ampere Spherical Tokamak (MAST). Good agreement with the sheath-limited velocity scaling relation is found in both the drift-fluid limit and when FLR effects are included. Agreement is also found with the inertial limit when simulations with GEM are carried out in the drift-fluid limit. However, a deviation from the inertial velocity scaling relation is observed when FLR effects are included. This deviation is explained through linearisation of the underlying equations. Finally, edge profiles and radial heat flux decay length are interpreted through radially averaged profiles of self-consistent turbulent simulations that include both the core and the SOL.

1 Introduction & Motivation

1.1 Energy demand in the 21st century

The world population is projected to reach over 10 billion by 2100 [6]. This population growth as well as the industrialisation of developing countries [7] will lead to increased energy demands worldwide as shown in fig. 1. On a shorter timescale an average global temperature increase of 1.5 °C above pre-industrial levels due to anthropogenic climate change is widely agreed to be a foregone conclusion. It seems unlikely that even the Paris climate agreement goal of limiting warming to 2 °C above pre-industrial levels will be met without significant effort and societal change [8]. In fact on the current trajectory a warming of 3 °C is expected [9]. The implications of such a rise in global mean temperature are varied and are not only environmental in nature but will also affect agricultural and economic systems. Likely effects of climate change on the current trajectory include but are not limited to: ocean acidification [10] (which threatens marine biota), vastly reduced insect populations leading to reduced pollination [11] and increased frequency of drought events [12]. If these devastating effects are to be avoided then global greenhouse gas emissions must be reduced on a relatively short timescale.

The technologies that are currently available with the potential to ameliorate the problem of greenhouse gas emission each suffer from similar problems, namely intermittency, geographical availability and political issues such as public distaste.

Solar and wind power both inherently suffer from the problem of intermittency. Energy is not continuously available from these sources. Setting aside the problem of nighttime energy generation solar power generation is affected by cloud cover. Wind energy generation is naturally affected by wind speed. These factors lead to solar and wind power being unreliable sources of energy that are too intermittent to meet peak demand. The problem of reliability could be solved with suitable storage and energy transport solutions however, such solutions remain to be found. It has been suggested that solar, water and wind energy could meet the world's energy needs [13]. However, such suggestions typically require close control of the world's energy demands.

Hydroelectric, tidal and geothermal energy are all geographically limited.

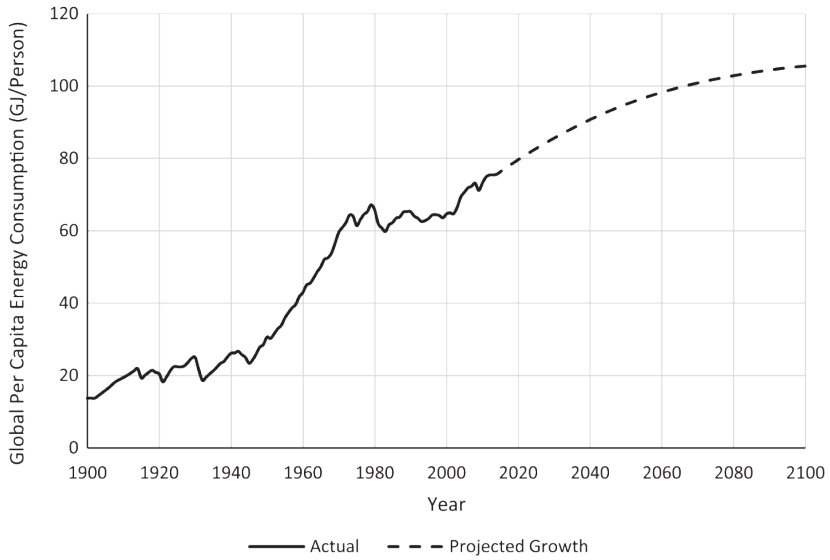

Figure 1: Predicted growth in global energy production (reproduced with permission from [1])

Where they are available they can contribute strongly to the energy generation in that region however, room for growth of these energy supplies is limited.

Nuclear fission, while often not thought of as green, could provide non-intermittent, reliable, zero-carbon energy. That is not to suggest that fission power is without problems. Long-lived radioisotopes must be managed almost indefinitely. There is also the problem of nuclear proliferation. Fissile materials required for the production of nuclear weapons could be obtained by nations with nuclear energy capabilities, which may be problematic. Beyond these pitfalls is the large problem that is the public appetite, or lack thereof for nuclear power.

1.2 Nuclear Fusion

Nuclear fusion offers an alternative to current energy technologies. The basic principle is that light nuclei can, under the right conditions, join together to form a larger nucleus, releasing energy in the process. The conditions required for nuclear fusion vary depending on the reaction in question. A table containing the cross-section as a function of energy is shown in fig. 2. It is clear from this figure that the most promising cross-section is that of the deuterium-tritium reaction. It is this reaction that is typically targeted by fusion research.

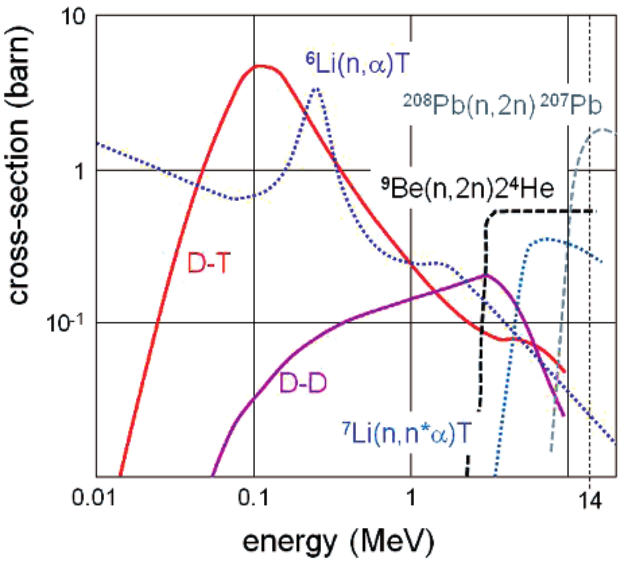

Figure 2: Fusion cross-section as a function of energy. The two cross-sections most often of interest in the context of magnetic confinement fusion (MCF) are the deuterium-tritium (D-T) cross-section (red line) and the deuterium-deuterium (D-D) cross-section (purple line) (reproduced with permission from [2])

This reaction is shown below:

$$^{2}\mathrm{D} + {}^{3}\mathrm{T} \longrightarrow {}^{4}\mathrm{He} + \mathrm{n}^{1} + 17.6\,\mathrm{MeV} \tag{1}$$

A deuterium nucleus (D) and tritium nucleus (T) fuse releasing an alpha particle (He) and an energetic neutron (n). Another benefit of this reaction is the abundance of deuterium in natural water with a ratio of $\frac{\mathrm{D}}{\mathrm{H}} \approx 10^{-4}$. Tritium while much less abundant can be bred from a lithium isotope (Li) with the following reactions:

$$^{6}_{3}\mathrm{Li} + {}^{1}_{0}\mathrm{n} \longrightarrow {}^{4}_{2}\mathrm{He} + {}^{3}_{1}\mathrm{T} + 4.8\,\mathrm{MeV} \tag{2}$$

$$^{7}_{3}\mathrm{Li} + {}^{1}_{0}\mathrm{n(fast)} \longrightarrow {}^{3}_{2}\mathrm{He} + {}^{3}_{1}\mathrm{T} + {}^{1}_{0}\mathrm{n(slow)} \tag{3}$$

These reactions would allow tritium to be generated from lithium by using the neutrons produced by the D-T reaction (eq. (1)). This abundance of energy dense fuel, combined with a lack of long-lived radioisotopes, makes fusion an appealing energy generation scheme.

However, it is clear from fig. 2, where the D-T cross-section peaks around

100 keV, that very high temperatures are required to make the D-T fusion reaction favourable. There are multiple approaches to achieving the kind of temperatures required for fusion to occur, but the scheme that will be focused on here is that of magnetic confinement fusion (MCF).

1.2.1 Magnetic Confinement Fusion

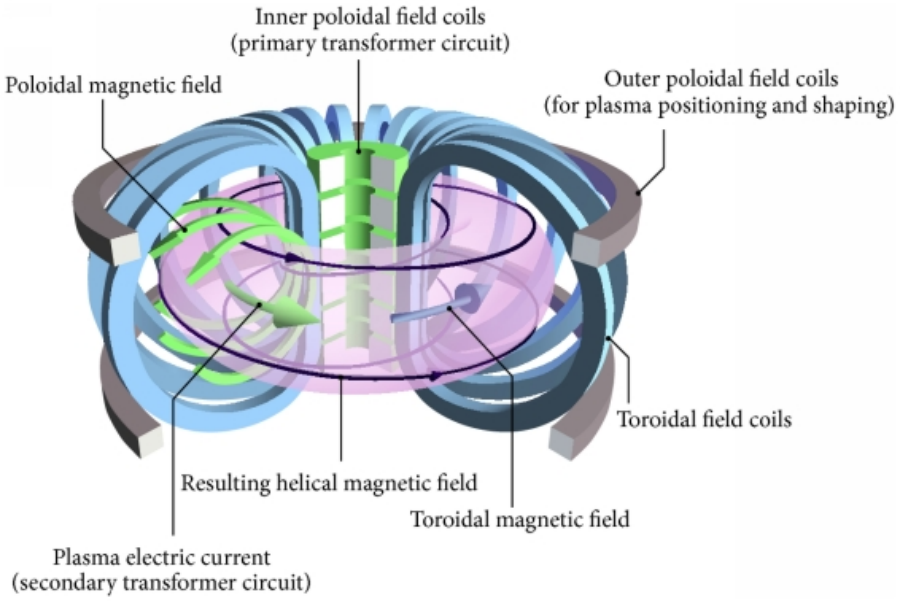

Figure 3: Schematic of the most basic components in a typical tokamak (image reproduced with permission from [3])

Magnetic confinement fusion, as the name suggests, aims to confine a fusion plasma using magnetic fields. There are two main types of magnetic confinement device, namely the tokamak and the stellarator. A stellarator employs a strongly shaped magnetic field to confine a plasma and does not rely on a central solenoid. In a stellarator the magnetic configuration is set by a rotational transform such that the net curvature drift found in tokamaks is not present and an equilibrium can be found between the plasma pressure and magnetic forces [14]. This rotational transform makes the engineering and construction of a stellarator more challenging than that of a tokamak since non-axisymmetric solenoidal coils are required in a stellarator. The alternative approach taken when designing a tokamak, is to create a rotational transform using a poloidal field generated by a toroidal electric current which

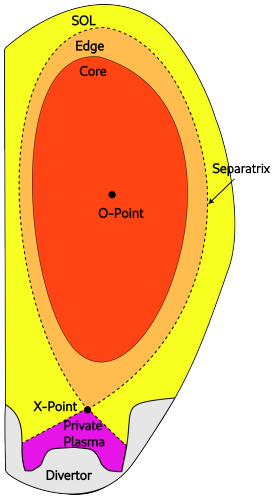

Figure 4: Typical tokamak single null geometry. The separatrix denotes the last closed flux surface.

is driven using a central coil. A secondary benefit of driving such a current is that it facilitates Ohmic heating. The design of a stellarator minimises current driven instabilities due to the absence of a toroidal plasma current. Stellarators also facilitate longer continuous operation since they lack the need to ramp the current in a central solenoid to generate the toroidal current needed in tokamaks. However, the axisymmetric nature of a tokamak reduces the difficulty of construction. The goal of either approach is to reach an ignition state where the fusion reactions are sustained by alpha-heating (heating from the generation of fast alpha particles from the fusion process). A figure of merit for how close a tokamak is to ignition is the fusion triple product [15], $nT\tau_E$, where n is the plasma density, T is the plasma temperature and τ_E is the energy confinement time. This criterion illustrates the appeal of larger plasma devices, and hence larger plasma volumes, as increasing the distance required for energy to reach the plasma edge increases the energy confinement time. Charged particles in a tokamak are confined on the helical field lines produced from the toroidal current and toroidal field coils (as illustrated in fig. 3). These field lines form closed surfaces of equal flux. Ideally a plasma would be perfectly confined on these 'flux surfaces' however, this is not the case. Particle drifts and turbulence lead to plasma moving across these flux surfaces. The region where the magnetic field lines transition from closing on themselves to closing on a material surface is known as the edge. The transition occurs at the separatrix and the region outside the separatrix is the SOL.

Field lines in the SOL close on plates designed to handle high heat fluxes, this region is known as the divertor. A cross-section illustrating this geometry is shown in fig. 4. It is in the scrape-off-layer (SOL) region that the exhaust of particles and energy occurs and is the region of interest for this work.

1.2.2 Sheath formation

When considering a plasma that is confined in a vacuum chamber (such as in a tokamak) one must consider the interaction between the plasma and the vessel walls. The Bohm criterion first published in [16, 17] is the natural starting point when considering sheath dynamics or sheath formation. In the description to follow we will consider a 1D case where a plasma meets a planar electrode.

When ions and electrons hit the plate they recombine and are lost. Due to their greater thermal speed, electrons are lost more quickly than ions which generates a positive space charge in the region near the plate. It then follows that a positive potential is formed in the plasma with respect to the wall. The variation between the plasma potential and the wall potential will be confined to a region on the order of a few Debye lengths in thickness, due to the well-known Debye shielding. This region is known as the sheath and equalises ion and electron fluxes by accelerating ions and repelling electrons.

The Bohm sheath derivation outlined below follows Chen [18]. We are addressing here the 1D steady state problem and assume the potential decreases monotonically with x towards the plate.

Conservation of energy requires:

$$\frac{1}{2}mu^2 = \frac{1}{2}mu_0^2 - e\phi(x) \tag{4}$$

$$u = \left(u_0^2 - \frac{2e\phi}{m}\right)^{\frac{1}{2}} \tag{5}$$

Where u is the ion velocity, m is the ion mass, e is the electron charge, ϕ is the potential and u_0 is the ion velocity at the sheath entrance. Ion continuity

then requires:

$$n_0 u_0 = n_i(x)u(x) \tag{6}$$

$$n_i(x) = n_0\left(1 - \frac{2e\phi}{m}\right)^{\frac{1}{2}} \tag{7}$$

Where n_0 is the ion density at the sheath entrance. Electrons are assumed to be in steady state and to follow the Boltzmann relation. This simplifies the behaviour of electrons in the sheath region by treating them as a Maxwellian population that responds instantaneously to changes in the potential. Such a simplification is valid when the following conditions are met. Electrons must be in local thermal equilibrium, meaning that their distribution function is Maxwellian, and they have a well-defined temperature T_e. They must also have a much higher mobility compared to ions which allows them to respond instantaneously to changes in potential. When these conditions are met we can write an expression for the electron density as follows.

$$n_e(x) = n_0 \exp\left(\frac{e\phi}{k_B T_e}\right) \tag{8}$$

Where k_B is the Boltzmann constant and T_e is the electron temperature. We can then solve for the potential using Poisson's equation

$$\nabla^2\phi = -\frac{\rho}{\epsilon} \tag{9}$$

$$\tag{10}$$

Which in our 1D case becomes:

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = \frac{e(n_e - n_i)}{\epsilon_0} \tag{11}$$

Then substituting in our expressions for the ion and electron densities:

$$\nabla^2\phi = \frac{en_0}{\epsilon_0}\left[\exp\left(\frac{e\phi}{k_B T_e}\right) - \left(1 - \frac{2e\phi}{m}\right)^{\frac{1}{2}}\right] \tag{12}$$

We then switch to the following dimensionless quantities which normalise the system to the electron temperature, the Debye length and the ion sound speed

respectively:

$$\chi \equiv -\frac{e\phi}{k_b T_e} \qquad \xi \equiv \frac{x}{\lambda_D} = x\left(\frac{n_0 e^2}{\epsilon_0 k_b T_e}\right)^{\frac{1}{2}} \qquad \mathcal{M} \equiv \frac{u_0}{(k_b T_e/m)^{\frac{1}{2}}} \tag{13}$$

Equation 12 then becomes:

$$\frac{\mathrm{d}^2\chi}{\mathrm{d}\xi^2} = \left(1 + \frac{2\chi}{\mathcal{M}}\right)^{-\frac{1}{2}} - \exp(-\chi) \tag{14}$$

We then multiply by $\frac{\mathrm{d}\chi}{\mathrm{d}\xi}$ and integrate with respect to ξ:

$$\int_0^\xi \frac{\mathrm{d}\chi}{\mathrm{d}\xi}\frac{\mathrm{d}^2\chi}{\mathrm{d}\xi^2}d\xi = \int_0^\xi \left(1 + \frac{2\chi}{\mathcal{M}^2}\right)^{-\frac{1}{2}}\frac{\mathrm{d}\chi}{\mathrm{d}\xi}d\xi - \int_0^\xi \exp(-\chi)\frac{\mathrm{d}\chi}{\mathrm{d}\xi}d\xi \tag{15}$$

Then identifying that the following simplification holds for the left-hand side, which allows us to apply the fundamental theorem of calculus:

$$\frac{\mathrm{d}\chi}{\mathrm{d}\xi}\frac{\mathrm{d}^2\chi}{\mathrm{d}\xi^2} = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{\mathrm{d}\chi}{\mathrm{d}\xi}\right)^2 \tag{16}$$

Since $\chi = 0$ at $\xi = 0$ the integration yields:

$$\frac{1}{2}\left(\left(\frac{\mathrm{d}\chi}{\mathrm{d}\xi}\right)^2 - \left(\frac{\mathrm{d}\chi_0}{\mathrm{d}\xi}\right)^2\right) = \mathcal{M}^2\left[\left(1 + \frac{2\chi}{\mathcal{M}^2}\right)^{\frac{1}{2}} - 1\right] + \exp(-\chi) - 1 \tag{17}$$

A second integration could be done numerically to solve for χ. However, since the left-hand side of eq. (17) is a sum of squared (real) terms the right-hand side of eq. (17) must be positive. We then Taylor expand the right-hand terms about $\chi = 0$ which yields:

$$\frac{1}{2}\chi^2\left(-\frac{1}{\mathcal{M}^2} + 1\right) > 0 \tag{18}$$

$$\mathcal{M}^2 > 1 \implies u_0 > \left(\frac{k_B T_e}{m}\right)^{\frac{1}{2}} \tag{19}$$

Where we have finally arrived at the Bohm criterion. The ion velocity at the sheath entrance must be greater than the ion sound speed in order for a stable sheath to form.

1.3 Filaments

The exhaust of particles and heat without damage to reactor components is of utmost importance for current and future fusion experiments. This exhaust is governed largely by radial transport in the SOL. Such transport is largely attributed to a type of plasma structure referred to interchangeably as blobs or filaments. The nomenclature arises from their structure, they are identified as field-aligned, radially elongated structures that are localised in the drift plane. Their localisation in the drift plane leads to a 'blob' of enhanced density while their radial elongation resembles a 'filament' aligned to the **B** field. Seemingly coherent structures were observed in the SOL quite some time ago using fast cameras [19] and probe arrays [20, 21]. These structures (now labelled as filaments) continue to be studied experimentally with probe arrays [22], fast cameras [23, 24], Li-Beam spectroscopy and gas puff imaging (GPI) [25], among other techniques. These techniques all show that transport in the edge is dominated by coherent large amplitude structures (blobs) and that in some cases blobs accounted for 50% of the particle transport in the SOL [26]. Transport arising from blobs is not diffusive in nature, rather the radial transport associated with blobs is dominated by $\mathbf{E} \times \mathbf{B}$ advection. This advective transport across field lines enhances particle and heat transport into the far SOL. This can increase unwanted interactions with the first wall, limiters, diagnostics, and antennas [27, 28]. They are also of interest when considering material limits [29], especially for next generation devices as heat fluxes approach material limits. Furthermore, Edge localised mode (ELM) filaments in a high confinement mode (H-mode) plasma can constitute significant heat fluxes and erode the divertor inhomogeneously [30]. As such, they strongly affect the plasma dynamics in the SOL, it is therefore important to understand filament dynamics if an understanding of SOL is sought.

First however, a basic definition of a filament should be established. Such a definition is provided by D'Ippolito in his review paper [31] The definition is as follows:

1. A filament should have a monopole density distribution with a peak value much greater than the background plasma;

2. It should be aligned parallel to the magnetic field, and its parallel vari-

ation should be much weaker than its variation in the perpendicular plane;

3. It should have a dominant $\mathbf{E} \times \mathbf{B}$ velocity in the direction of a charge polarizing force and a corresponding dipole structure in potential and vorticity.

1.3.1 Blob generation

The generation of blobs is a topic of active research in the fusion community however, theory and simulation indicate that filaments are born from a non-linear saturation of edge turbulence or other instabilities including but not limited to the interchange [28, 32], drift wave [33, 34, 35], and cooperative elliptic [36] instabilities. It is expected that in tokamaks the dominant instabilities will be the curvature-driven sheath interchange mode [37] and the resistive X-point mode [38] in the region of low B-field, also known as the region of bad curvature.

In order to study the generation of blobs, simulations that self-consistently solve a set of governing equations with a realistic 3D geometry [39, 40] or more commonly a reduced 2D geometry [41, 42] are often employed. These simulations are difficult in part due to numerical difficulties, in part due to computational expense, and in the case of 2D simulations difficulties arise from choosing an appropriate closure for the dynamics aligned to the magnetic field. The choice of this closure will necessarily affect the dynamics of the system. However, when one hopes to study self-consistent blob generation such an impact on the dynamics is not ideal. Moreover, difficulties arise due to the turbulent nature of such a simulation. Dissipation constants must be chosen without it being clear a priori what values are suitable. The problem is also multiscale and energy that cascades to the highest frequency supported by the simulation must be handled appropriately, this problem is naturally linked to the issue of choosing dissipation constants.

1.3.2 Blob transport mechanisms

Regardless of the instability that leads to the generation of filaments their dynamics thereafter are likely to follow the same basic principles in a tokamak.

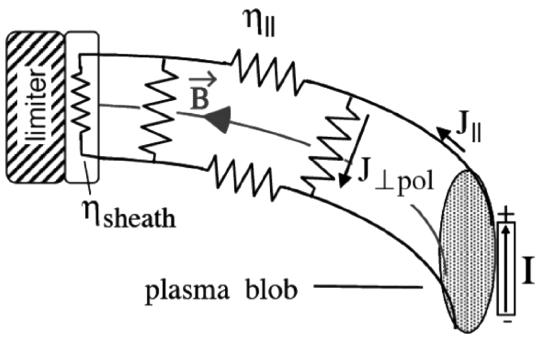

Figure 5: Blob equivalent circuit (reproduced with permission [4]) The dynamics of filaments are strongly influenced by the dominant current path. Magnetic curvature drives a charge separation which drives a current. This current can close either along field lines with a parallel resistivity $\eta_{\|}$ and through the sheath $J_{\|}$ with a resistivity η_{sheath} or through polarisation currents $J_{\perp pol}$

This hypothesis is supported by filament interaction studies in which simulations are initialised with multiple filaments present. Such studies [43] suggest that blobs do not interact until in very proximity and even then that they do not interact strongly and can therefore generally be treated independently of one another. The hypothesis is tested and supported by a filament interaction investigation presented in section 3.4.

It is on this basis that many studies [44, 45, 46] of filaments begin by assuming the filament has already appeared, thereby dispensing with the problem of generating filaments self-consistently. This is useful also because it allows blob parameters such as their density, temperature, perpendicular extent, and parallel profile to be varied independently and systematically.

In any case the basic blob dynamics suggested by Krasheninnikov [28] and Garcia [47] can be understood by considering a balance of currents as illustrated as an equivalent circuit in fig. 5. The current in a blob can be generated by the curvature drive. This drive arises from the fact that particles confined to curved magnetic field lines experience a centrifugal force perpendicular to the **B** field. The outward centrifugal force is then given by

$$\mathbf{F}_{\mathcal{K}} = \frac{m v_{\|}^2}{R_c} \hat{\mathbf{r}} \tag{20}$$

Where R_c is the radius of curvature. This force naturally leads to a drift given

by:

$$\mathbf{v}_{\mathcal{K}} = \frac{1}{q}\frac{\mathbf{F}_{\mathcal{K}} \times \mathbf{B}}{B^2} \tag{21}$$

$$= \frac{mv_{\parallel}^2}{q}\frac{\mathbf{R}_c \times \mathbf{B}}{R_c^2 B^2} \tag{22}$$

This drift is charge dependent and hence leads to a current as electrons and ions drift in opposite directions. There are two primary current paths this current can close through. The current can close either through sheath currents (which is known as the sheath limited regime) or through polarisation currents (which is known as the inertial regime). In the case of currents closing mainly through the sheath as is the case with large filaments, then the model given by Krasheninnikov [28] applies. This model predicts an inverse square relationship between filament width and peak filament velocity. In the case of currents closing through polarisation currents, as is the case for small filaments, then the Garcia model [47]. This model predicts a square root relationship between filament width and peak filament velocity. Both the inertial and sheath limited regime are captured in Walkden's description [48] which is as follows where superscript e and o represent the even and odd spatial components of a quantity.

When interpreting these equations it is useful to note that an even function multiplied by an even function produces an odd function, an odd function multiplied by an odd function produces an odd function and an even function multiplied by an odd function produces an odd function. Note also that even order differential operators retain the parity of the function to which they are applied, and odd order operators reverse the parity. In the Cartesian coordinate system used here, where a slab geometry is assumed for simplicity x is the radial direction, z is the direction parallel to the magnetic field and y is the binormal direction.

$$\frac{\partial \Omega^e}{\partial t} + \mathbf{v}_E^o \cdot \nabla \Omega^o + \mathbf{v}_E^e \cdot \nabla \Omega^e = \frac{1}{n}\nabla_{\parallel} J_{\parallel}^e \tag{23}$$

$$\frac{\partial \Omega^o}{\partial t} + \mathbf{v}_E^o \cdot \nabla \Omega^e + \mathbf{v}_E^e \cdot \nabla \Omega^o = \frac{g}{n}\frac{\partial nT}{\partial y} + \frac{1}{n}\nabla_{\parallel} J_{\parallel}^o \tag{24}$$

Where n is the plasma density, Ω is the plasma vorticity, T is the electron temperature, $J_{\parallel}$ is the parallel plasma current, g is the approximated effective

gravity from the curvature drive. $\mathbf{v_E}$ is the $\mathbf{E} \times \mathbf{B}$ velocity given by:

$$\mathbf{v}_E^e = \mathbf{b} \times \nabla \phi^o \tag{25}$$

$$\mathbf{v}_E^o = \mathbf{b} \times \nabla \phi^e \tag{26}$$

When linearised sheath boundary conditions are applied to eqs. (23) and (24) and the equations are integrated along the magnetic field lines then one obtains the following equations where $L_\parallel$ is the parallel connection length and V_f is the floating potential:

$$\frac{\partial \Omega^e}{\partial t} + \mathbf{v}_E^o \cdot \nabla \Omega^o + \mathbf{v}_E^e \cdot \nabla \Omega^e = \frac{1}{L_\parallel \sqrt{T}} \left(T V_f + \phi^e \right) \tag{27}$$

$$\frac{\partial \Omega^o}{\partial t} + \mathbf{v}_E^o \cdot \nabla \Omega^e + \mathbf{v}_E^e \cdot \nabla \Omega^o = \frac{g}{n} \frac{\partial nT}{\partial y} + \frac{\phi^o}{L_\parallel \sqrt{T}} \tag{28}$$

Next the interaction between ϕ^e and ϕ^o is assumed small. The gradients are also estimated as $\nabla_\perp \approx 1/\delta_\perp$ where $\delta_\perp$ is the filament characteristic width. The vorticity and potential are related with the following expression:

$$\Omega = \nabla_\perp^2 \phi \tag{29}$$

The gradient is then evaluated using our above approximation which results in the following:

$$\Omega = \frac{\phi}{\delta^2} \tag{30}$$

Similarly for the $\mathbf{E} \times \mathbf{B}$ velocity where the odd parity component of the potential generates the radial velocity:

$$\mathbf{v}_R = \mathbf{v}_E^e = \frac{\phi^o}{\delta_\perp} \tag{31}$$

$$\mathbf{v}_E^o = \frac{\phi^e}{\delta_\perp} \tag{32}$$

Stationary solutions $\left(\frac{\partial}{\partial t} \to 0 \right)$ are finally sought for eq. (28) where either the sheath current the inertial term is retained, assuming that the peak velocity occurs at the same time as the peak of the odd component of vorticity. Given

the relation of the radial velocity to the odd component of the potential and hence the odd component of the vorticity this assumption seems reasonable. First the inertial term is assumed to balance the drive term:

$$\mathbf{v}_E^e \cdot \nabla \Omega^o \approx \frac{g}{n} \frac{\partial nT}{\partial y} \tag{33}$$

Then substituting in our approximated expressions for the radial velocity and vorticity.

$$\frac{\phi^o}{\delta_\perp} \frac{1}{\delta_\perp} \frac{\phi^o}{\delta^2} \approx \frac{g}{n} \frac{nT}{\delta_\perp} \tag{34}$$

Next we group terms and identity the approximated form of our radial velocity.

$$\frac{\phi^{o2}}{\delta_\perp^4} \approx \frac{g}{n} \frac{nT}{\delta_\perp} \tag{35}$$

$$\frac{\mathbf{v}_R^2}{\delta_\perp^2} \approx \frac{g}{n} \frac{nT}{\delta_\perp} \tag{36}$$

$$\implies \mathbf{v}_R \approx \sqrt{\delta_\perp \frac{gnT}{n}} \tag{37}$$

Finally, the density and temperature fields are linearised, and we see the max radial velocity in the inertially limited case $\mathbf{v}_I$ is predicted to scale according to $\mathbf{v}_{\max} \propto \delta_\perp^{0.5}$:

$$\mathbf{v}_I \approx \sqrt{\delta_\perp g \frac{\delta p}{n_0 + \delta n}} \tag{38}$$

Now we take the case where the sheath current balances the drive term:

$$\frac{g}{n} \frac{\partial nT}{\partial y} \approx \frac{\phi^o}{L_\parallel \sqrt{T}} \tag{39}$$

Rearranging for the potential then identifying our linearised radial velocity and substituting yields:

$$\phi^o \approx \frac{g}{n} \frac{\partial nT}{\partial y} L_\parallel \sqrt{T} \tag{40}$$

$$\implies \mathbf{v}_{Sh} \approx \frac{g L_\parallel \delta p \sqrt{T_0 + \delta T}}{(n_0 + \delta n)\, \delta_\perp^2} \tag{41}$$

And we see that in the sheath limited regime an inverse square relationship

($\mathbf{v}_{\max} \propto \delta_{\perp}^{-2}$) is predicted between blob size and max radial blob velocity $\mathbf{v}_{sh}$. Finally, when $\mathbf{v}_{Sh}$ and $\mathbf{v}_I$ are equated one finds an expression for the fundamental filament size δ^*[49, 50] where both regimes exhibit the same velocity:

$$v_I \approx v_{Sh} \implies \delta^* = \left(g L_{\parallel}^2 \delta p \frac{T_0 + \delta T}{n_o + \delta n} \right)^{1/5} \tag{42}$$

These well-known velocity scaling predictions have been observed in drift fluid simulations [45]. Divergence from the inertial limit in gyrofluid simulations is discussed in section 5.

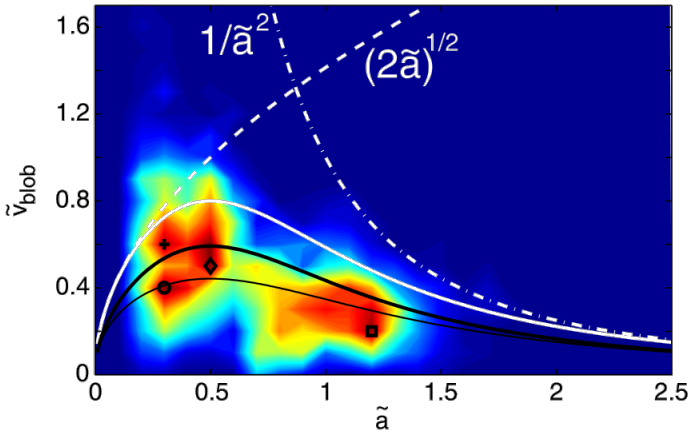

Figure 6: Normalised filament radial velocity $\tilde{v}_{blob}$ plotted against normalised filament width $\tilde{a}$. The dashed and dash-dotted lines show the analytical scaling predicted for the inertial and the sheath-limited regime respectively. The 2D probability distribution on top of which the analytical results are plotted was measured experimentally at TORPEX. Figure reproduced from [5].

Experimental results verifying these velocity scaling laws are sparse due to the difficulty of diagnosing filaments of the relevant sizes. The scaling relations are asymptotic which means that both very large and very small filaments must be diagnosed to verify them. To measure the inertial limit very small filaments must be observed, which itself is challenging. To measure the sheath limit very large filaments must be measured which in reality are limited by the size of the vessel and so can't be arbitrarily large, they also have a tendency to break up into smaller filaments. However, Theiler et al. [5] measured blob probability as a function of filament velocity and filament width with a probe method at Torpex. This probability function reproduced the filament velocity scaling laws and is shown in fig. 6 and shows reasonable agreement with the scaling

relations presented above.

1.4 Kinetic & Fluid Theory

When considering the dynamics of a plasma system one may initially consider a kinetic approach. This approach is appealing because the most fundamental description of a plasma is on the scale of its constituent particles. In a magnetic field charged particles will experience a Lorentz force and obey the following equation of motion[51].

$$m_i \frac{\partial \mathbf{v_i}}{\partial t} = q_i \left(\mathbf{E} + \mathbf{v_i} \times \mathbf{B} \right) \tag{43}$$

Where $\mathbf{v_i}$, m_i, q_i are the particle velocity, mass, and charge respectively. $\mathbf{E}$ and $\mathbf{B}$ are the electric and magnetic fields experienced by the particle. The problem with this approach lies mainly in its computational expense. Even for sparse plasmas a great number of particles are required to be simulated. So many that for large 3D problems this approach is largely infeasible.

Alternatively one could formulate a statistical approach. One could consider the distribution function $f(\mathbf{x}, \mathbf{v}, t)$ which describes the probability density for a particle to exist at position $\mathbf{x}$ with velocity $\mathbf{v}$ at time t [15]. In order to conserve particle number the following expression must hold.

$$\frac{\partial f}{\partial t} = -\nabla \cdot (\dot{\mathbf{x}} f) - \nabla_v \cdot (\dot{\mathbf{v}} f) \tag{44}$$

Which can be written in advective form as:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + \mathbf{a} \cdot \nabla_v f = 0 \tag{45}$$

Where $\mathbf{a}$ is the acceleration. When the acceleration is provided by the Lorentz force one arrives at the Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f + (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_v f = 0 \tag{46}$$

When multiple plasma species are considered and collisional terms $\mathcal{C}[f_i, f_j]$ are added where the subscript is the species label and $\mathcal{C}[f_i, f_j] = -\mathcal{C}[f_j, f_i]$ one obtains the Vlasov-Focker-Plank (VFP) equation (eq. (47)):

$$\frac{\partial f_i}{\partial t} + \mathbf{v}_i \cdot \nabla f_i + \frac{Z_i e}{m} (\mathbf{E} + \mathbf{v}_i \times \mathbf{B}) \cdot \nabla_v f_i = \mathcal{C}[f_i, f_j] \tag{47}$$

The electric and magnetic fields $\mathbf{E}$ and $\mathbf{B}$ can be calculated using Gauss's law and Ampere's law where ρ and $\mathbf{J}$ are the charge and current densities.

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon} \tag{48}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} \tag{49}$$

Numerical solutions of this equation are still typically difficult to find due to the 6 dimensional nature of the problem. The drift-kinetic and gyrokinetic equations reduce this dimensionality to 5 but are still typically prohibitively intensive. It is also from eq. (47) that the widely used drift ordered fluid equations are derived.

1.4.1 Drift-reduced fluid models

Because of the aforementioned difficulty of solving the VFP equation and because the electromagnetic field equations depend on the lowest two moments it may seem appealing to take velocity moments of eq. (47). This procedure is exactly the procedure that produces drift-reduced fluid equations and magnetohydrodynamics (MHD) equations. The difference is in the closure method.

In the derivation of the MHD and drift-fluid equations, a perturbed Maxwell-Boltzmann velocity distribution is assumed. In general the n^{th} fluid moment of eq. (47) is obtained from:

$$\int \mathbf{v}_i^n \frac{\partial f_i}{\partial t} d^3\mathbf{v}_i + \int \mathbf{v}_i^{n+1} \cdot \nabla f_i d^3\mathbf{v}_i + \int \mathbf{v}_i^n \frac{Z_i e}{m} (\mathbf{E} + \mathbf{v}_i \times \mathbf{B}) \cdot \nabla_v f_i d^3\mathbf{v}_i = \int \mathbf{v}_i^n \sum_j \mathcal{C}[f_i, f_j] d^3\mathbf{v}_i \quad (50)$$

The difficulty with this approach of course is that each moment depends on the next. This evaluation of moments is necessarily truncated at some point by applying an appropriately chosen closure. The closure chosen in the case of drift-fluid equations is typically the Braginskii closure [52] which holds for collisional plasmas. The larger problem is that of the gyroviscous cancellation and neglect of FLR effects elsewhere in the derivation. The gyroviscous cancellation is an approximation made in the derivation of drift-fluid equations where an advective momentum term is cancelled with a gyroviscous term in the pressure tensor. This cancellation is only valid for $k_\perp \rho_s \ll 1$, where $k_\perp$ is the perpendicular wavenumber of the feature or wave of interest, which is not the case for small filaments or drift waves. Gyrofluidics bypasses the need for FLR approximations of the fluid equations by accounting for FLR effects at the gyrokinetic level.

1.5 Gyrokinetic & Gyrofluid Theory

The basis of gyrokinetic theory lies in three parts [53];

1. The gyrokinetic Vlasov equation
2. A set of gyrokinetic Maxwell equations
3. An energy conservation expression for the gyrokinetic Vlasov-Maxwell system which includes coupling terms

This is analogous to the kinetic description but rather than considering particle motions the motion of rings of charges are considered.

Gyrofluid equations are an alternative description of plasma dynamics to the fluid descriptions provided by MHD, drift-fluid equations or the Braginskii equations. Each of these formulations has strengths and weaknesses and a corresponding regime of applicability. The strength of gyrofluid equations is

their ability to incorporate FLR effects to arbitrary order. This comes at the expense however, of complicating the collisional terms.

It can be shown from the Lorentz force ($\mathbf{F} = q\mathbf{E} + \mathbf{v} \times \mathbf{B}$) that a charged particle moving with a velocity component perpendicular to a magnetic field (in the absence of an electric field) will exhibit a fast gyromotion perpendicular to the magnetic field. This gyromotion typically occurs on a timescale much faster than the drift motion of the particle.

To illustrate this point let us consider the equations of motion in 3D Cartesian coordinates for a charged particle moving with some arbitrary velocity $\mathbf{v} = v_x\hat{\mathbf{x}} + v_y\hat{\mathbf{y}} + v_x\hat{\mathbf{z}}+$ in a uniform magnetic field which is aligned to the $\hat{\mathbf{z}}$ axis and the absence of an electric field. Then setting up the equations for each velocity component we obtain:

$$\begin{cases} \dfrac{dv_x}{dt} = \dfrac{qB}{m} v_y \\ \dfrac{dv_y}{dt} = -\dfrac{qB}{m} v_x \\ \dfrac{dv_z}{dt} = 0 \end{cases}$$

Then integrating these coupled differential equations once to arrive at expressions for the velocity we obtain the following which describe uniform circular motion perpendicular to the magnetic field:

$$\begin{cases} v_x(t) = v_{x0}\cos(\omega t) - v_{y0}\sin(\omega t) \\ v_y(t) = v_{x0}\sin(\omega t) + v_{y0}\cos(\omega t) \\ v_z(t) = v_{z0} \end{cases}$$

Where we have defined $\omega = \dfrac{qB}{m}$ which is also known as the cyclotron frequency. Defining a perpendicular velocity $v_\perp = \sqrt{v_{x0}^2 + v_{y0}^2}$ and a phase constant $\phi = \tan^{-1}\left(\dfrac{v_{y0}}{v_{x0}}\right)$, not to be confused with the electric potential, we can rewrite these equations in a simpler form:

$$\begin{cases} v_x(t) = v_\perp \cos(\omega t + \phi) \\ v_y(t) = v_\perp \sin(\omega t + \phi) \\ v_z(t) = v_{z0} \end{cases}$$

Integrating again to obtain expressions for the particle positions we obtain:

$$\begin{cases} x(t) = \rho_s \sin(\omega t + \phi) + x_c \\ y(t) = -\rho_s \cos(\omega t + \phi) + y_c \\ z(t) = z_0 + v_{z0} t \end{cases}$$

Where we have identified the Larmor radius ρ_s and the guiding centre position, the location about which the particle orbits, (x_c, y_c):

$$\rho_s = \frac{v_\perp}{\omega} \tag{51}$$

$$x_c = x_0 - \rho_s \sin\phi \tag{52}$$

$$y_c = y_0 + \rho_s \cos\phi \tag{53}$$

The fast gyromotion that we have shown the basis for above and the comparatively slower drift motion that arises in the presence of an electric field can be decoupled. Kinetic equations typically track the particle motion as gyromotion is executed. Gyrokinetic equations average over the gyromotion and effectively track the motion of the charge of the particles distributed spatially over one Larmor orbit. This reduces the dimensionality of the problem from six dimensions to five. Conventional fluid models typically approximate the ion Larmor radius to be small compared to the size of the system. They may thereafter attempt to reintroduce FLR effects through the use of a gyroviscous stress tensor as in the Braginskii equations. The issue that arises with this formulation is that FLR effects are maintained only to first order in terms of an expansion parameter chosen in either a 'fast dynamics' or 'slow dynamics' ordering [54]. This leads to a loss of generality.

Gyrofluid formulations avoid this loss of generality by initially considering not the particle distribution function but the gyrocentre distribution function and then accounting for this with gyroaveraging operators when considering fields [55]. This approach allows the gyroviscous cancellation required in the derivation of a Braginskii fluid to be avoided. It also leads to a more accurate description of FLR effects but the gyroaveraging operation must be carried out with care.

There are two main branches of modern gyrofluidics. These branches are

delta-F and Full-F modelling. Both branches have a similar starting point whereby velocity moments are taken of a gyrokinetic Vlasov-Maxwell system. They differ, however in the approximations they make in order to render the procedure of taking these velocity moments tractable. Models within each type of gyrofluidics will also differ by what kinetic effects (if any) they include. Some models for instance will include effects such as Landau damping [56] or phase mixing [57], others attempt to include collisional effects [58, 59].

1.6 Guiding centre models

The initial attempts made in the field of gyrofluidics came in the form of corrections to guiding centre models. The typical approach was to begin with a continuity equation and the Poisson equation.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\rho) = 0 \qquad \mathbf{v} = -\nabla\phi \times \mathbf{e_z} \tag{54}$$

$$\nabla^2 \phi = -\rho \tag{55}$$

Where ρ is the charge density and $\mathbf{v}$ is the $\mathbf{E} \times \mathbf{B}$ drift velocity. Gyroaveraging is then introduced by considering the charge of a particle to be distributed on a ring of radius r_L equal to one Larmor orbit about the guiding centre. When two species are considered (electrons and ions) and gyroaveraging is ignored for electrons due to their small Larmor radius, which is smaller than the ion Larmor radius by a factor of $\frac{m_i}{m_e}$. This assumption is justified not just by the smallness ratio but also by the fact that the turbulence spectrum we are concerned with only extends as far as the ion gyroradius. When this assumption is taken the system becomes:

$$\frac{\partial n_i}{\partial t} + \nabla \cdot (n_i \bar{\mathbf{v}}) = 0 \tag{56}$$

$$\frac{\partial n_e}{\partial t} + \nabla \cdot (n_e \mathbf{v}) = 0 \tag{57}$$

$$\nabla^2 \phi = -(\bar{n_i} - n_e) \tag{58}$$

$$\mathbf{v} = -\nabla\phi \times \mathbf{e_z}, \qquad \bar{\mathbf{v}} = G * \mathbf{v}, \qquad \bar{n_i} = G * n_i$$

Where G is a convolution integral operator in configuration space, or simply a multiplication in wave-number space. Such a multiplication in wave-number space was convenient for early gyrofluid studies as spectral methods were common [60, 61, 62] at the time which meant that the evolving quantities were defined in wave-number space, hence removing the need for additional Fourier transforms.

1.7 Delta-F models

A more rigorous approach to deriving a set of gyrofluid equations is to take moments of the gyrokinetic description. In doing so one can guarantee conservation of energy and momentum. It also allows higher moments to be derived than can be accessed using the guiding centre correction method discussed above. It should be noted however, that such a derivation is non-trivial and that various approaches have produced gyrofluid equations with varying degrees of success.

The derivation of a delta-f or full-f gyrofluid model hinges on the gyrokinetic Vlasov equation. The difference between the delta-f approach and the full-f approach is whether the distribution function is assumed to be a perturbed Maxwellian ($f = F^M + \delta f$) or not. Such an assumption renders the derivation of the gyrofluid moment equations more tractable but is not always strictly correct.

In the case of a delta-f model then once the gyrokinetic Vlasov equation is field-aligned, linearised and combined with a polarisation equation (to enforce quasineutrality) and sometimes an induction equation (to include electromagnetic effects) velocity moments of the system can be taken (typically in guiding centre space rather than particle space) to arrive at delta-f gyrofluid moment equations.

There have been multiple derivations of such models [63, 64, 58, 59] the latter of which were largely energetic corrections to the former by way of a more rigorous derivation procedure. Some delta-f derivations also approximate the electron moment equations by neglecting FLR effects for electrons due to their much smaller Larmor radius [65].

This hierarchy of moment equations is then closed, often by including Landau damping in the heat flux moments. One key advantage of such a

closure is the avoidance of the typical high collision frequency closure applied to drift fluid equations. Such an assumption while valid in highly collisional plasmas is questionable in less collisional plasmas, for example, low density high temperature plasmas. Finally, dissipative terms can be added to the moment equations in an ad-hoc fashion. Ideally these terms would be added to the gyrokinetic Vlasov equation, however the derivation of a self-consistent gyrokinetic collision operator that can be used when deriving gyrofluid equations is an open question. Instead, these terms are added in such a way that they agree with the drift fluid equations in the zero FLR limit. Particle and energy source terms are then added to the equations, the form of these terms is chosen to match the scenario being modelled. For example, a decaying exponential particle source could be chosen to attempt to capture particle recycling at the divertor.

1.7.1 GEM Model Equations

The moment equations chosen for this study, the GEM gyrofluid equations, to which density and temperature sources have been added, are presented below. A detailed derivation of these equations is found in [59], but a rudimentary introduction to the moment equaitons will be provided here. This model falls into the category of delta-f and was chosen in part due to its inclusion of collisional effects and a benchmark against a gyrokinetic code that seemed promising [66].

However, after working with and becoming familiar with the model the validity of certain approximations made may be questionable when considering simulations of the edge and filaments. Specifically the linearisation that is intrinsic in a delta-F model and the Boussinesq approximation that follows may be unsuitable for the study of filaments. In order to avoid such approximations however, one must employ a full-F model the complexity of which is typically much greater than the model presented here. The dissipation-free moment equations and field equations of the model used for the simulations that follow are shown below. The following advective and parallel gradient operators are used where $\hat{\mathbf{x}}$ is the radial direction, $\hat{\mathbf{z}}$ is the binormal direction (perpendicular to both $\mathbf{B}$ and $\mathbf{x}$) and f and g are arbitrary functions.

$$[f, g] = \left(\frac{\partial f}{\partial x}\frac{\partial g}{\partial z} - \frac{\partial g}{\partial x}\frac{\partial f}{\partial z} \right) \tag{59}$$

$$\nabla_{\parallel} f = \mathbf{b} \cdot \nabla f - \beta_e \left[A_G, f \right] \tag{60}$$

Where $\mathbf{b}$ is the direction parallel to the magnetic field and β_e is the electron beta, the ratio of the electron pressure to the magnetic pressure. A_G is the gyroaveraged electromagnetic potential which we define in eq. (75). The curvature operator $\mathcal{K}$ which captures the effects of magnetic gradients and magnetic curvature is defined as:

$$\mathcal{K}(f) = \nabla \cdot \left(\frac{1}{B}(\mathbf{b} \times \nabla f) \right) \tag{61}$$

We simplify this operator for use in our slab geometry in section 2.3.5

The subscript z is the species subscript in the following equations. And the

following constants are defined where a_z, τ_z and μ_z are the background charge density, temperature/charge, and mass/charge ratios:

$$a_z = \frac{Zn_z}{n_e}, \qquad \tau_z = \frac{T_z}{ZT_e}, \qquad \mu_z = \frac{m_z}{Zm_D} \tag{62}$$

The equation for density n_z is then:

$$\begin{aligned}\frac{\mathrm{d}n_z}{\mathrm{d}t} =& -\left[\phi_G, n_z\right] - \left[\Omega_G, T_{z\perp}\right] - B\nabla_\parallel \frac{u_{z\parallel}}{B} + \beta_e\left[\chi_G, q_{z\perp}\right] \\ &+ \mathcal{K}\left(\tau_z \frac{p_{z\parallel} + p_{z\perp}}{2} + \phi_G + \frac{\Omega_G}{2}\right) + S_{n_z}\end{aligned} \tag{63}$$

Where the bracket terms $[\phi_G, n_z]$, $[\Phi_G, T_z]$, and $[\chi_G, q_{z\perp}]$, represent the $\mathbf{E}\times\mathbf{B}$ advection of the species density n_z where the latter two terms arise from FLR corrections. The $B\nabla_\parallel \frac{u_{z\parallel}}{B}$ term represents the linear parallel divergence, $\mathcal{K}$ is the curvature operator defined above. S_n is a density source term. We systematically vary this source term in section 4.1 to calculate stable background solutions which we use to initialise isolated filament simulations.

The equation for parallel velocity $u_\parallel$ and gyroaveraged parallel electromagnetic potential $A_G = \Gamma_1 A_\parallel$ is:

$$\begin{aligned}\beta_e \frac{\partial A_G}{\partial t} + \mu_z \frac{\mathrm{d}u_{z\parallel}}{\mathrm{d}t} =& -\mu_z\left[\phi_G, u_{z\parallel}\right] - \mu_z\left[\Omega_G, q_{z\perp}\right] - \nabla_\parallel(\phi_G + \tau_z p_{z\parallel}) \\ &+ \beta_e\left[\chi_G, (\Omega_G + \tau_z T_{z\perp})\right] - \left(\Omega_G + \tau_z T_{z\perp} - \tau_z T_{z\parallel}\right)\nabla_\parallel \log B \\ &+ \tau_z\mu_z\mathcal{K}\left(\frac{4u_{z\parallel} + 2q_{z\parallel} + q_{z\perp}}{2}\right) - \mu_z\frac{S_{n_z}u_{z\parallel}}{n_z}\end{aligned} \tag{64}$$

The bracket terms, curvature and parallel gradient terms take their usual form. $\left(\Omega_G + \tau_z T_{z\perp} - \tau_z T_{z\parallel}\right)\nabla_\parallel \log B$ Is a magnetic pumping term arising due to the combinations of parallel flow and conduction, magnetic moment conservation at the gyrokinetic level, and the parallel gradient in the strength of the magnetic field. $\mu_z\frac{S_{n_z}u_{z\parallel}}{n_z}$ is a term added to preserve momentum corresponding to the density source in eq. (63).

The equation for parallel temperature, the temperature associated with the

motion of particles along magnetic field lines $T_\parallel$ is:

$$\begin{aligned}\frac{1}{2}\frac{\mathrm{d}T_{z\parallel}}{\mathrm{d}t} =& -\frac{1}{2}\left[\phi_G, T_{z\parallel}\right] - B\nabla_\parallel\frac{u_{z\parallel}+q_{z\parallel}}{B} + \beta_e\left[\chi_G, q_{z\perp}\right] \\ &- \left(u_{z\parallel}+q_{z\perp}\right)\nabla_\parallel \log B + \mathcal{K}\left(\tau_z\frac{p_{z\parallel}+2T_{z\parallel}}{2} + \frac{\phi_G}{2}\right) + S_{T_{z\parallel}}\end{aligned} \tag{65}$$

The bracket terms, curvature and parallel divergence terms take their usual form. $\left(u_{z\parallel}+q_{z\perp}\right)\nabla_\parallel \log B$ Is a magnetic pumping term arising for the same reasons as that in eq. (64). $S_{T_{z\parallel}}$ is a heat source for the parallel temperature.

The equation for perpendicular temperature, the temperature associated with the motion of particles perpendicular to magnetic field lines $T_\perp$ is:

$$\begin{aligned}\frac{\mathrm{d}T_{z\perp}}{\mathrm{d}t} =& -\left[\phi_G, u_{z\perp}\right] - \left[\Omega_G, (n_z+2T_{z\perp})\right] + B\nabla_\parallel\frac{q_{z\perp}}{B} \\ &- \beta_e\left[\chi_G, (u_{z\parallel}+2q_{z\perp})\right] - \left(u_{z\parallel}+q_{z\perp}\right)\nabla_\parallel \log B \\ &+ \mathcal{K}\left(\tau_z\frac{p_{z\perp}+3T_{z\perp}}{2} + \frac{\phi_G+4\Omega_G}{2}\right) + S_{T_{z\perp}}\end{aligned} \tag{66}$$

Where the bracket terms, curvature and parallel divergence terms take their usual form. The magnetic pumping term this time takes the form $\left(u_{z\parallel}+q_{z\perp}\right)\nabla_\parallel \log B$. The term $S_{T_{z\perp}}$ is a heat source for the perpendicular temperature which we later vary systematically to obtain stable background profiles.

The equation for parallel heat flux $q_\parallel$ arising from the parallel temperature $T_\parallel$ is then:

$$\mu_z\frac{\mathrm{d}q_{z\parallel}}{\mathrm{d}t} = -\mu_z\left[\phi_G, q_{z\parallel}\right] - \frac{3}{2}\tau_z\nabla_\parallel T_{z\parallel} + \tau_z\mu_z\mathcal{K}\left(\frac{3u_{z\parallel}+8q_{z\parallel}}{2}\right) \tag{67}$$

Where again the bracket terms, curvature and parallel gradient terms take their usual form.

And finally the equation for the gyroaveraged, parallel electromagnetic field $\chi_G = \Gamma_2 A_\parallel$ and parallel heat flux $q_{z\perp}$ arising from the perpendicular temper-

ature is given by:

$$\begin{aligned}\beta_e \frac{\partial \chi_G}{\partial t} + \mu_z \frac{\mathrm{d}q_{z\perp}}{\mathrm{d}t} =& -\mu_z \left[\phi_G, q_{z\perp}\right] - \mu_z \left[\Omega_G, (u_{z\parallel} + 2q_{z\perp})\right] - \tau_z \nabla_\parallel T_{z\perp} \\ &+ \beta_e \left[\chi_G, (\phi_G + \tau_z p_{z\parallel})\right] + \beta_e \left[\chi_G, 2(\Omega_G + \tau_z T_{z\perp})\right] \\ &- \left(\Omega_G + \tau_z T_{z\perp} - \tau_z T_{z\parallel}\right) \nabla_\parallel \log B + \tau_z \mu_z \mathcal{K} \left(\frac{u_{z\parallel} + 6q_{z\perp}}{2}\right)\end{aligned} \tag{68}$$

Where again the bracket terms, curvature and parallel gradient terms take their usual form. The magnetic pumping term enters as $\left(\Omega_G + \tau_z T_{z\perp} - \tau_z T_{z\parallel}\right) \nabla_\parallel \log B$.

The equations below are the polarisation eq. (69) and induction eq. (70) equations respectively. These equations are inverted to find the electrostatic ϕ and electromagnetic $A_\parallel$ potentials. In the case of the electromagnetic potential terms from eqs. (64) and (68) enter into the inversion.

$$\sum_z a_z \left[\Gamma_1 n_z + \Gamma_2 T_{z\perp} + \frac{\Gamma_0 - 1}{\tau_z}\phi\right] = 0 \tag{69}$$

$$\nabla_\perp^2 A_\parallel + \sum_z a_z \left[\Gamma_1 u_{z\parallel} + \Gamma_2 q_{z\perp}\right] = 0 \tag{70}$$

The pressure terms eq. (74) are linearised as shown below, taken from [63], which follows from the delta-f ordering and normalisation taken in the derivation of Dorland's model in [63] and in GEM. $n = n_0 + \tilde{n}$ and n is normalised to n_0 and likewise for $T_\parallel$ and $T_\perp$. We start with an expression for pressure in dimensional units $p = nT$. We then linearise

$$p_0 + \delta p = (n_0 + \delta n)(T_0 + \delta T) \tag{71}$$

$$p_0 + \delta p = n_0 T_0 + n_0 \delta T + T_0 \delta n + \delta n \delta T \tag{72}$$

Next we drop the term that is a perturbation times a perturbation, and normalise

$$\frac{p_0}{p_0} + \frac{\delta p}{p_0} = \frac{n_0 T_0}{n_0 T_0} + \frac{n_0}{n_0}\frac{\delta T}{T_0} + \frac{T_0}{T_0}\frac{\delta n}{n_0} \tag{73}$$

At which point we arrive at the normalised, linearised pressure.

$$p_{z\parallel} = n_z + T_{z\parallel} \qquad p_{z\perp} = n_z + T_{z\perp} \tag{74}$$

The terms below in eqs. (75) and (76) are the gyroaverage operators. These terms correspond to a multiplication of Fourier coefficients in wavenumber space by a Bessel function of order zero [63]. This operation is trivially implemented in spectral codes such as [61] since the Fourier coefficients of evolving fields are immediately available. However, the operation must be adapted for use in a finite-difference or finite-volume code. This typically comes in the form of a Padé approximation such as in [67, 68, 69, 59, 70, 71]. With the application of such an approximation the gyroaveraging operators take the form of Laplacian inversions.

$$\phi_G = \Gamma_1 \phi \qquad A_G = \Gamma_1 A_\parallel \tag{75}$$

$$\Omega_G = \Gamma_2 \phi \qquad \chi_G = \Gamma_2 A_\parallel \tag{76}$$

Where the Padé approximated gyroaverage operators are given in [72] as:

$$\Gamma_0 \left(b_z\right) \to \left(1 - \rho_z^2 \nabla_\perp^2\right)^{-1} \tag{77}$$

$$\Gamma_1 \left(b_z\right) \to \left(1 - \frac{1}{2}\rho_z^2 \nabla_\perp^2\right)^{-1} \tag{78}$$

$$\Gamma_2 \left(b_z\right) \to \frac{1}{2}\rho_z^2 \nabla_\perp^2 \left(1 - \frac{1}{2}\rho_z^2 \nabla_\perp^2\right)^{-1} \Gamma_1 \left(b_z\right) \tag{79}$$

Next, collisional dissipation is added in the following form such that they match the collisional Braginskii model [52] in the drift-fluid limit.

$$\beta_e \frac{\partial A_\parallel}{\partial t} + \mu_z \frac{\mathrm{d}}{\mathrm{d}u_{z\parallel}} = \cdots - \mu_e \nu_e \left[\eta J_\parallel + \frac{\alpha_e}{\kappa_e}\left(q_{e\parallel} + q_{e\perp} + \alpha_e J_\parallel\right)\right] \tag{80}$$

$$\frac{1}{2}\frac{\partial T_{z\parallel}}{\partial t} = \cdots - \frac{\nu_z}{3\pi_z}\left(T_{z\parallel} - T_{z\perp}\right) \tag{81}$$

$$\frac{\partial T_{z\perp}}{\partial t} = \cdots + \frac{\nu_z}{3\pi_z}\left(T_{z\parallel} - T_{z\perp}\right) \tag{82}$$

$$\mu_z \frac{\partial q_{z\parallel}}{\partial t} = \cdots - \mu_z \nu_z \frac{5}{2\kappa_e}\left[q_{z\parallel} + 1.28\left(q_{z\parallel - 1.5 q_{z\perp}}\right) + \frac{3}{5}\alpha_z J_\parallel\right] \tag{83}$$

$$\mu_z \frac{\partial q_{z\perp}}{\partial t} = \cdots - \mu_z \nu_z \frac{5}{2\kappa_e}\left[q_{z\perp} + 1.28\left(q_{z\parallel - 1.5 q_{z\perp}}\right) + \frac{2}{5}\alpha_z J_\parallel\right] \tag{84}$$

Where the coefficients η, α, κ, π_e, and ν_z are resistivity, thermoelectric

coupling, thermal conduction, parallel viscosity and collision frequency respectively. For singly electrons the following values are used as per Scott [59]:

$$\alpha_e = 0.71, \qquad \kappa_e = 3.2, \qquad \pi_e = 0.73, \qquad \eta = 0.51 \tag{85}$$

And for singly charged ions:

$$\alpha_i = 0, \qquad \kappa_i = 3.9, \qquad \pi_i = 0.96 \tag{86}$$

The collision frequencies are calculated as:

$$\nu_e = \frac{n_e Z_i^2 e^4 \Lambda_e}{3\epsilon_0^2 m_e^{1/2} (2\pi T_e)^{3/2}}, \qquad \nu_i = \frac{n_e Z_i^4 e^4 \Lambda_i}{12\epsilon_0^2 m_i^{1/2} (\pi T_i)^{3/2}} \tag{87}$$

Where n_e is the electron density, Z_i is the ion charge state, e is the electron charge, $\ln \Lambda_e$ and $\ln \Lambda_i$ are the Coulomb logarithms for electron-ion and ion-ion collisions, ϵ_0 is the permittivity of free space, and T_i and T_e are the ion and electron background temperatures and m_e and m_i are the electron and ion masses.

The NRL plasma formulary provides expressions for Λ_e and Λ_i in *cgs* and eV for deuterium:

$$\Lambda_e = 24 - \ln\left(n_e^{1/2} T_e^{-1}\right) \tag{88}$$

$$\Lambda_i = 23 - \ln\left(n_i^{1/2} T_i^{-3/2}\right) \tag{89}$$

These collisional terms act to resist parallel currents and heat fluxes and also to reduce temperature anisotropy. Finally, numerical diffusive terms are added to each moment equation of the form:

$$\frac{\mathrm{d}F}{\mathrm{d}t} -= (\nu_\perp \nabla_\perp^2 + \nu_\parallel \nabla_\parallel^2) F \tag{90}$$

These terms should in principle be as small as possible as they are added largely for numerical stability. They act to limit the maximum spatial frequency supported by the system. Care should also be taken when setting numerical diffusion coefficients not introduce unwanted viscosity into the sys-

tem. This is discussed in section 4.7

1.8 Full-F models

Another (more recently developed) subset of gyrofluid models is that of the Full-F model. These models avoid the linearisation on which delta-F models depend, instead evolving each moment equation self consistently. This comes at the expense of a more complicated derivation path which typically requires multiple conversions between coordinate spaces. Such conversions are often facilitated by the use of Lie transform methods [73, 71], except the earliest Full-F models [70].

Regardless of which Full-F derivation method is used the underlying principles are the same. The high-frequency cyclotron motion is eliminated at the gyrokinetic level (from the particle Lagrangian). Using the particle Lagrangian a Vlasov-Maxwell system can be derived. Gyrofluid velocity moments of the Vlasov-Maxwell system then yield the moment equations to which closures are applied.

The calculation of potentials also differs between Full-F and delta-F models. In a delta-F model only the background part of the distribution function enters the polarisation and magnetisation terms. In a Full-F model the full distribution is retained everywhere, however a long wavelength approximation is taken in order to make tractable the gyrokinetic Maxwell's equations. This implies that FLR corrections to the polarisation drift are neglected [70]. It could be argued that this approximation is more suitable for edge studies than those made in delta-F models.

It has also been suggested that Full-F models are less well able to handle collisions compared to delta-F models. However, it appears that collisional effects are typically added to delta-F models in an ad-hoc manner or not at all [70]. Part of the problem with adding collisional effects is that in order to do so consistently one must add a collisional term to the gyrokinetic Vlasov-Maxwell system and keep the term when taking velocity moments to arrive at the moment equations. This is a non-trivial task and a long-standing problem of gyrofluidics. As such, it does not seem like the difficulty of including collisions is a deficiency of Full-F models.

A notable advantage of gyrofluid models is that an exact energy conserva-

tion law is satisfied as long as appropriate closures are applied to the higher moments correctly. This is important as it can be shown that neither spurious energy sources nor sinks enter the model.

2 Implementation

In order to solve the gyrofluid equations shown in section 1.7.1 finite difference methods are employed. The moment equations for each plasma species eqs. (63) to (68) must be integrated (typically with an implicit scheme in the case of fluid equations). The polarisation (eq. (69)) and induction (eq. (70)) equations must be inverted. The gyroaverage operators (eqs. (78) and (79)) require inversions for each plasma species and the spatial gradients require finite difference approximations. To achieve these requirements a physics model has been developed in the BOUT++ (BOUndary Turbulence in C++) framework which provides flexible methods for all the above steps, is well optimised for parallel processing. BOUT++ is open source and is available at `https://github.com/boutproject/BOUT-dev`.

2.1 Finite difference methods

In order to evaluate plasma fluid equations spatial derivatives must be taken. However, because the spatial domain has been discretised then so too must the spatial derivatives. Assuming the function f to be differentiated can be described by a Taylor series about the point h one can do the following:

$$f(x+h) = \sum_{i=0}^{\infty} \frac{\partial^i f}{\partial x^i} \frac{h^i}{i!}$$
$$f(x+h) = f(x) + \frac{\partial f(x)}{\partial x} h + \frac{\partial^2 f(x)}{\partial x^2} \frac{h^2}{2} + \dots$$

Then subtracting $f(x)$ from both sides and dividing by h we arrive at an expression for the second order forward difference.

$$\frac{f(x+h) - f(x)}{h} = \frac{\partial f(x)}{\partial x} + \frac{\partial^2 f(x)}{\partial x^2} \frac{h}{2} + \dots$$

Then rearranging for the derivative we aim to approximate:

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+h) - f(x)}{h} - \frac{\partial^2 f(x)}{\partial x^2} \frac{h}{2} - \dots \tag{91}$$

Finally we drop terms beyond the first and discretise:

$$\frac{\partial f_i}{\partial x} = \frac{f_{i+1} - f_i}{h} - \mathcal{O}(h) \tag{92}$$

In this case the truncation error, the error that arises from truncating the series, is of order h, and so our grid spacing h must be chosen to be suitably small that a term on the order of h is negligible. This expression gives the forward difference (where h is the grid spacing) and an expression for the backward difference is obtained in the same manner by expanding $f(x)$ about $-h$ instead of h:

$$\frac{\partial f_i}{\partial x} = \frac{f_i - f_{i-1}}{h} - \mathcal{O}(h) \tag{93}$$

Here we have expressions for the forward and backward differences which can be useful at the start or end of a domain where $f(x+h)$ or $f(x-h)$ may be undefined. However, more commonly used is the central difference. To derive an expression for the central difference we take two Taylor expansions, one about h and the other about $-h$

$$f(x+h) = f(x) + \frac{\partial f(x)}{\partial x}h + \frac{\partial^2 f(x)}{\partial x^2}\frac{h^2}{2} + \frac{\partial^3 f(x)}{\partial x^3}\frac{h^3}{6} + \dots$$
$$f(x-h) = f(x) - \frac{\partial f(x)}{\partial x}h + \frac{\partial^2 f(x)}{\partial x^2}\frac{h^2}{2} - \frac{\partial^3 f(x)}{\partial x^3}\frac{h^3}{6} + \dots$$

Now we subtract one from the other, where even terms in the series cancel to obtain:

$$f(x+h) - f(x-h) = \frac{\partial f(x)}{\partial x}2h + \frac{\partial^3 f(x)}{\partial x^3}\frac{2h^3}{6} + \dots \tag{94}$$

Dividing through by $2h$, truncating the series after the first term, rearranging, and discretizing we arrive at an expression for the second order central difference.

$$\frac{\partial f_i}{\partial x} = \frac{f_{i+1} - f_{i-1}}{2h} - \mathcal{O}(h^2) \tag{95}$$

In the limit of $\lim h \to 0$ these expressions are equivalent since the higher order terms vanish and should give the same result. However, practically a finite,

non-zero value of h must be chosen and the numerical properties of the chosen scheme can impact the solution. Therefore, the choice of derivative scheme is important. Higher order schemes, or more complicated non-oscillatory, flux limiting, or upwinding schemes could be chosen and in general a derivative scheme can be represented with a stencil describing the weighting of each point in the evaluation of the derivative.

A similar argument holds for the time derivatives where h now represents our timestep:

$$f_{i+1} = f_i + \frac{\partial f_i}{\partial t} h + \mathcal{O}(h^2) \tag{96}$$

eq. (96) is just the well-known forward Euler method. Higher order schemes also exist with better accuracy and can be described with stencils. As with spatial derivatives, one could consider the backwards difference:

$$f_{i+1} = f_i + \frac{\partial f_{i+1}}{\partial t} h + \mathcal{O}(h^2) \tag{97}$$

Where eq. (97) is simply the backward Euler method. The complication here is that the new value f_{t+1} can not be directly calculated, instead an implicit equation must be solved by some iterative method such as the conjugate gradient, Newton-Krylov or GMRES (generalised minimal residual) method. However, they typically have better numerical stability than explicit methods, so larger time steps can be taken.

In both of these methods the local truncation error, the error after one step, is $\mathcal{O}(h^2)$. However, the error accumulates with successive steps and so after N steps the global error is $\mathcal{O}(h)$ since $N \propto h^{-1}$.

In practice more complicated multistep methods, with more favourable numerical properties, are typically used, however the expressions above serve to introduce the notion of truncation error and its dependence on grid spacing which we will revisit in sections 3.3 and 4.8.

2.2 The BOUT++ framework

BOUT++ [74] is a framework for writing fluid simulations in 3D curvilinear geometry. It is an open source project with multiple developers working to improve and extend the library. This also facilitates cooperation between different groups towards the common goal of comprehensive, stable plasma physics simulations. It is written in C++ and utilizes MPI and OpenMP.

BOUT++ provides a set of ODE integrators (explicit, implicit and IMEX schemes are available) as well as operators and data types for calculating time derivatives such as algebraic operators and differential operators. There are also Laplacian inversion routines which can be used to solve equations of the form:

$$d\nabla_\perp^2 x + \frac{1}{c_1}(\nabla_\perp c_2)\cdot\nabla_\perp x + ax = b$$

The link between these solvers and the equations to be solved is the 'physics model', a class in BOUT++ in which the equations to be solved are described. In a physics model normalisations are carried out, initial conditions, boundary conditions, the geometry and other options are read in either from grid files, input files or as command line arguments. The time derivatives are then calculated using the operators and inversion routines mentioned above. The time derivatives are then used by the chosen ODE integrator to advance the equations in time at which point the time derivatives are calculated again and the process repeats. This modular design separates the physics from the numerical methods and allows groups working with different equations to mutually benefit from improvements to the library. The separation of physics from numerical implementation also allows faster development of physics models since the physics equations are described concisely and compactly.

To illustrate the methods and flexibility provided by the BOUT++ framework an example of the source code for a physics model is listed here that solves a 1D heat conduction equation:

$$\frac{\partial T}{\partial t} = \nabla_\parallel\left(\chi\partial_\parallel T\right) \tag{98}$$

Listing 1: Source code for a simple heat conduction equation implemented as a BOUT++ physics model

```
#include <bout/physicsmodel.hxx>

class Conduction : public PhysicsModel {

private:
  Field3D T; // Evolving temperature equation only
  BoutReal chi; // Parallel conduction coefficient

protected:
  // This is called once at the start
  int init(bool UNUSED(restarting)) override {

    // Get the options
    auto &options = Options::root()["conduction"];

    // Read from BOUT.inp, setting default to 1.0
    // The doc() provides some documentation in BOUT.settings
    chi = options["chi"].doc("Conduction coefficient").
   withDefault(1.0);

    // Tell BOUT++ to solve T
    SOLVE_FOR(T);
    return 0;
  }

  int rhs(BoutReal UNUSED(time)) override {

    mesh->communicate(T); // Communicate guard cells

    ddt(T) =
        Div_par_K_Grad_par(
            chi, T); // Parallel diffusion Div_{||}( chi *
   Grad_{||}(T) )
    return 0;
  }
};

BOUTMAIN(Conduction);
```

The differencing method (central, upwind, or Weighted Essentially Non-Oscillatory (WENO)) and the order of the differencing scheme (2nd, 3rd or 4th) are chosen at runtime. Initial profiles, constants (in this case χ) and boundary conditions are also chosen at runtime, as is the time integrator.

2.3 The GEM physics model

In order to solve the GEM gyrofluid equations presented in section 1.7.1 they were implemented in a physics model for BOUT++ similarly to the toy model introduced above, albeit with significantly more complexity. The physics model is available at `https://github.com/AdamD94/GEM` and is also named, simply, GEM after the equations it evolves. This physics model was initially based on an implementation that had been intended for core simulations but was found to be unsuitable for filament simulations. It was then significantly rewritten with filament simulations in mind. Staggered grids were added in the parallel direction for flux like terms to avoid checkerboarding. Also, where evolved quantities were known to be non-negative such as the density and temperature of a plasma species then the natural log of that quantity is evolved instead of the quantity itself. This is known to increase the stability of finite-volume and finite-difference simulations, it also allows division by that quantity in terms containing a spatial derivative of the quantity to be avoided since $\frac{\partial \log n}{\partial x} = \frac{1}{n}\frac{\partial n}{\partial x}$.

Within the GEM physics model a class was written to represent a plasma species. This allows the multi-fluid nature of the GEM model to be captured and minimizes code replication or repetition. Where there is a sum over species in the GEM physics model, there is a function to calculate that species' contribution to the sum in the species class.

One final modification that was made for the sake of stability was to apply limits to scalar fields that should remain strictly positive, the density and temperature of each species in this case. A value of 0.1 times the background value was used for all the GEM simulations shown here. Without this modification the linearised parallel gradient terms, which are not multiplied by the variable they are acting on, could try to drive the density negative which is of course unphysical.

Listing of the main source files of the implemented physics model are shown in appendix A, but the source is also available at `https://github.`

com/AdamD94/GEM. The physics model was implemented and extended over the course of the work described herein, initially only the first two moments (eqs. (63) and (64)) were implemented. Once stable filament simulations were obtained with the reduced implementation the higher order moments, temperature and heat flux (eqs. (65) to (68)) were added. A 2D version of the physics model was also developed alongside, and sharing most of its code with, the 3D implementation. For the 2D version parallel closures for momentum and heat flux were added. These closures are discussed in section 3.

In section 2.1 we motivated the kind of discretisations that are required to evolve our physics model equations. Thankfully the BOUT++ framework provides a suite of validated and tested ODE (ordinary differential equation) integrators, differential operators and Laplacian inversion routines. Naturally for the kind of simulation we are concerned with here, the scalability of the code is also important and BOUT++ has been shown to demonstrate good scaling to 1000s of processors [75].

2.3.1 Differential operators

The $\mathbf{E} \times \mathbf{B}$ advection terms described in section 1.7.1 take the form of Poisson brackets and were naturally implemented as such, in the physics model. Because the perpendicular and parallel dynamics in GEM are decoupled no y derivatives enter the bracket operator defined in eq. (59). BOUT++ provides a bracket operator that supports multiple methods, one of which is an Arakawa scheme, which neglects y derivatives, and as such, our brackets were implemented with it. The $\mathbf{E} \times \mathbf{B}$ advection of the species density for example is calculated similarly to how it is shown here: Internally BOUT++ uses a stencil method to calculate the spatial gradients.

```
1 ddt(n) -= bracket(phi1, N, BRACKET_MODE, CELL_CENTRE)
```

The curvature term was approximated as shown in eq. (114) and therefore depends only on the z derivative. The implementation in the physics model was of the form:

```
1 Field3D Species::curvature(const Field3D &fld, CELL_LOC loc_out
    ) const {
2   TRACE("Species::curvature");
3   // Approximating curvature for a flux tube geometry
```

```
  return (g * DDZ(fld, loc_out));
}
```

Similarly, BOUT++ provides operators to calculate parallel gradients and parallel divergences which was used for our parallel divergence and parallel gradient terms.

```
if (evolve_N) {
  ddt(logN) -= Div_par(Upar_stag, CELL_CENTRE);
}
if (evolve_A_Upar) {
  ddt(A_Upar_stag) -= Grad_par(phi1 + tau * Ppar, CELL_YLOW);
}
```

The correspondence between differential operators used in the GEM model equations and the differential operators available in GEM greatly simplified the implementation the RHS of our moment equations.

2.3.2 Numerical considerations

Some modifications were made to the physics model for purely numerical reasons. The first modification was the adoption of staggered grids. The physics model was initially implemented with all values defined on cell centres. This implementation gave rise to a grid scale oscillatory numerical instability in the parallel direction which was a manifestation of the well-known pressure checkerboarding problem. This was remedied by adopting staggered grids as has been done in other filament simulations [76, 77]. The grids for parallel velocity and heat fluxes (eqs. (64), (67) and (68)) were staggered in y. As a result, parallel velocity and heat fluxes were defined on cell faces rather than cell centres. This solved the checkerboarding problem but complicated boundary condition application since boundary conditions for all variables should be applied at the same locations. This complication required the use of interpolation when setting boundary conditions on the staggered grids in the same fashion as in [76].

Another modification which improved stability was evolving the logarithm of our scalar variables rather than the variables themselves. This helped to enforce positivity for our scalar variables for which negative values are unphysical. In addition to this, limiters were added to the code to prevent our

scalar variables falling below a runtime configurable threshold value and taking unphysical values.

2.3.3 Time Integration

BOUT++ implements the Method of Lines (MOL) which means the spatial operators and time integration are treated separately. A set of ODE solvers are available in BOUT++ including implicit, explicit and IMEX schemes. GEM follows STORM3D [76] in its selection of an implicit, variable order, variable time-step, Newton-Krylov Backwards Difference Formula (BDF). GEM uses the CVODE solver which is provided by the SUNDIALS library. This solver uses a sophisticated multistep method with error estimation to integrate our moment equations. The decoupling that BOUT++ provides through the Method of Lines greatly simplified the integration of our moment equations.

2.3.4 Laplacian Inversion

Finally, the polarisation equation, the gyroaverage operators, and in the case of electromagnetic simulations the induction equations must be solved. These equations all take a similar form, but an example here will be given using the polarisation equation. Starting from eq. (69)

$$\sum_z a_z \left[\Gamma_1 n_z + \Gamma_2 T_{z\perp} + \frac{\Gamma_0 - 1}{\tau_z}\phi \right] = 0 \tag{99}$$

Next we take the single ion species case and assume gyroscreening for electrons is negligible due to their small Larmor radius ($\Gamma_0 \approx 1$ for electrons)

$$a_i \left(\Gamma_1 n_i + \Gamma_2 T_{i\perp} + \frac{\Gamma_0 - 1}{\tau_i}\phi \right) + a_e \left(\Gamma_1 n_e + \Gamma_2 T_{e\perp} \right) = 0 \tag{100}$$

Next we substitute the Padé approximated form of Γ_0 and rearrange to isolate $\nabla_\perp^2 \phi$

$$\nabla_\perp^2 \phi = -\frac{\tau_i}{\rho_i^2 a_i} \left(1 - \rho_i^2 \nabla_\perp^2\right) \left(a_i \left(\Gamma_1 n_i + \Gamma_2 T_{i\perp} \right) + a_e \left(\Gamma_1 n_e + \Gamma_2 T_{e\perp} \right) \right) \tag{101}$$

Which is then in a suitable form to be inverted with a Laplacian inversion routine as is done in our physics model.

2.3.5 Simulation Domain, Geometry, & Boundary Conditions

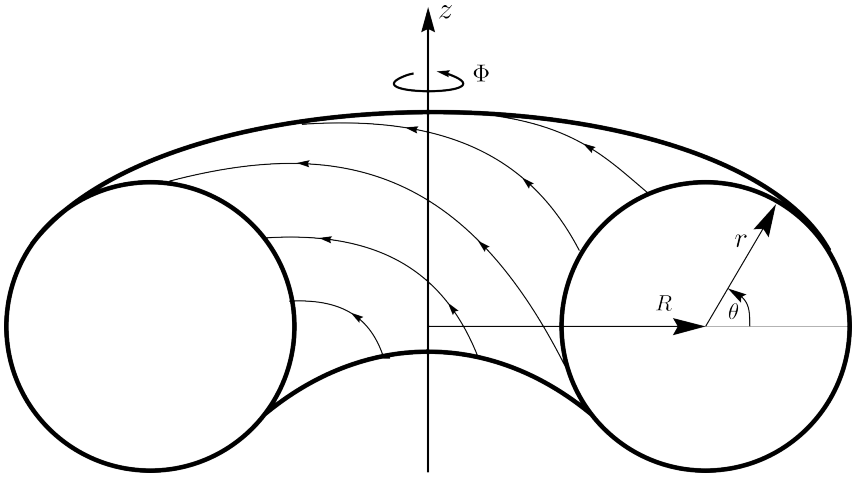

Figure 7: Representative tokamak geometry. Field lines are denoted as the arrowed lines. R is the major radius, (r, ϕ, and ψ) are the radial, poloidal, and toroidal angle respectively

When describing the geometry in a tokamak a toroidal description (r, ϕ, θ) is an obvious initial choice, where θ is the poloidal angle and Φ is the toroidal angle. Alternatively one could use flux coordinates (ψ, θ, Φ) where ψ is the poloidal flux. However, a more appropriate coordinate system is a field-aligned coordinate system. In such a coordinate system the magnetic field lines appear straight. This description facilitates spatial scale separation and efficient simulation of the predominantly field-aligned structures that are found in tokamaks. A further simplification of a field-aligned coordinate system is that of a flux tube. A Cartesian flux tube coordinate system (x, y, z) is a local description aligned to a magnetic field line which is assumed constant $\mathbf{B} = B\hat{\mathbf{y}}$. The $\hat{\mathbf{x}}$ direction is the radial direction and the $\hat{\mathbf{z}}$ direction is the remaining orthogonal direction which by our definition here is always perpendicular to both $\mathbf{B}$ and $\hat{\mathbf{x}}$. The z boundaries are periodic, the x boundaries are zero gradient Neumann boundary conditions and so the filaments we simulate in this domain should not touch either of the x boundaries. We ensure this by initializing the filament away from the inner boundary and making the domain large enough that the simulation finishes before the filament gets close to the outer bound-

ary. The y boundaries are then a mixture of Neumann and Dirichlet boundary conditions. The lower y boundary represents the target/divertor. This means that sheath boundary conditions must be applied, namely that the ion speed at the boundary must be equal to or greater than the Bohm speed.

$$u_{\parallel i}\Big|_{y=l_\parallel} \geq c_s \tag{102}$$

$$u_{\parallel e}\Big|_{y=l_\parallel} = u_{\parallel i} \exp\Big\{V_f - \phi\Big|_{y=l_\parallel}\Big\} \tag{103}$$

Where c_s is the Bohm speed and V_f is the floating potential from basic Bohm sheath theory [78]. Similar to the argument presented above regarding pressure linearisation we must linearise the sheath boundary conditions. Here we take the linearised Debye sheath boundary conditions presented by Ribeiro and Scott[69, 68]

$$u_{i\parallel} = p_{e\parallel} \qquad u_{e\parallel} = u_{i\parallel} - (\phi - V_f T_{e\parallel}) \tag{104}$$

$$q_{e\parallel} = 3T_{e\parallel} \qquad q_{e\perp} = T_{e\perp} \tag{105}$$

$$q_{i\parallel} = 3\tau_i T_{i\parallel} \qquad q_{i\perp} = \tau_i T_{i\perp} \tag{106}$$

The upper y boundary represents the midplane, as such, the boundary conditions are provided by symmetry arguments, namely that the flux like variables take Dirichlet zero value boundary conditions and to the scalar variables Neumann boundary conditions are applied. These symmetry arguments are commonly used for isolated filament simulations and arise from the fact that to set them otherwise would imply a net flow from the upper to the lower divertor.

A sketch of our slab geometry and boundary conditions is shown in fig. 8

Our potential ϕ also requires inner and outer radial x boundary conditions. For our slab simulations we use the background potential calculated from our 1D simulations. In the 1D case it is calculated by looking for steady state solutions with $u_i\| = u_{e\|}$ to equations for velocity as follows starting from eq. (64) for electrons and ions and discarding all the terms except for the

parallel divergence and the momentum conservation term.

$$\frac{\mathrm{d}u_{z\parallel}}{\mathrm{d}t} = -\frac{1}{\mu_z}\nabla_\parallel(\phi_G + \tau_z p_{z\parallel}) - \frac{S u_{z\parallel}}{n_z} \tag{107}$$

Then we take the difference between this equation for electrons and for ions and set the time derivatives to zero as we are looking for a steady state solution. We also take the gyroaverage operators in the drift-fluid limit as we are looking a solution that does not vary in x and z:

$$-\frac{1}{\mu_i}\nabla_\parallel(\phi + \tau_i p_{i\parallel}) + \frac{1}{\mu_e}\nabla_\parallel(\phi + \tau_e p_{e\parallel}) = 0 \tag{108}$$

After rearranging and integrating we see that our potential is defined up to a constant

$$\mu_i\nabla_\parallel(\phi + \tau_e p_{e\parallel}) - \mu_e\nabla_\parallel(\phi + \tau_i p_{i\parallel}) = 0 \tag{109}$$

$$\implies \phi = \frac{\mu_i\tau_e p_{e\parallel} - \mu_e\tau_i p_{i\parallel}}{\mu_e - \mu_i} + C \tag{110}$$

To constrain our potential we set the potential at the sheath entrance equal to the floating potential.

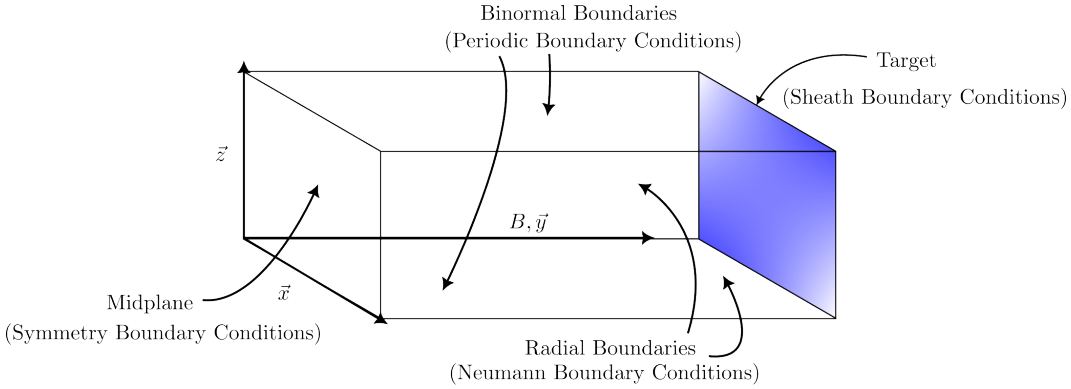

Figure 8: A sketch of the slab geometry used for isolated filament simulations with the boundary conditions listed in the figure. Sheath boundary conditions are applied at the target, symmetry boundary conditions are applied at the midplane, zero gradient Neumann boundary conditions are applied at the radial boundaries and periodic boundary conditions are applied at the z boundaries

In the case of our coupled core-sol simulations there are two regions in our computational domain which are treated differently. In the core region we do not apply sheath boundary conditions or midplane symmetry boundary

conditions at the y boundaries instead the domain is periodic, similar to our z boundaries such that any flux leaving at the lower y boundary enters at the upper y boundary. A sketch of this coupled core-sol geometry is shown in fig. 9

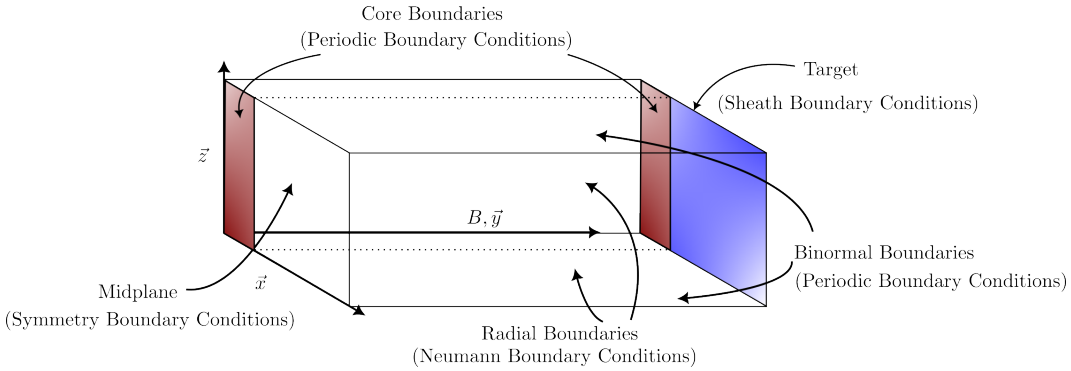

Figure 9: A sketch of the slab geometry used for coupled core-sol simulations with the boundary conditions listed in the figure. In the SOL Sheath boundary conditions are applied at the target and symmetry boundary conditions are applied at the midplane. In the core the y boundaries are periodic, zero gradient Neumann boundary conditions are applied at the radial boundaries and periodic boundary conditions are applied at the z boundaries

With our slab geometries we need to handle the curvature term manually since we have approximated the B field as constant in x and z. To handle the curvature term we approximate it in the same manner as Easy[76, 71] starting from the equation for our curvature operator in eq. (61):

$$\mathcal{K}(f) = \nabla \cdot \left(\frac{1}{B} (\mathbf{b} \times \nabla f) \right) \tag{111}$$

We approximate B in a cylindrical coordinate system (R, Ψ, Z) where R is the radial coordinate, Ψ is the azimuthal coordinate and z is the vertical coordinate

$$\mathbf{B} \approx \frac{B_0 R_0}{R} \tag{112}$$

Then using this approximate form of B in our curvature operator

$$\nabla \cdot \left(\frac{1}{B} (\mathbf{b} \times \nabla f) \right) \approx \frac{2}{R_0 B_0} \frac{\partial f}{\partial z} \tag{113}$$

Therefore to include the effect of magnetic gradients and curvature in our slab

simulations our curvature operator takes the reduced form:

$$\mathcal{K}(f) = -g\frac{\partial f}{\partial z} \tag{114}$$

$$g = \frac{2}{R_0 B_0} \tag{115}$$

Where g is often described as an effective gravity constant.

3 2D Filament simulations

In order to reduce the expense of filament simulations parallel (to B) dynamics are sometimes approximated such that a single 2D drift plane can be simulated. This allows relatively inexpensive 2D simulations to be used to carry out parameter scans that would require much more compute time if 3D simulations were used. To reduce our system from 3D to 2D the parallel dynamics must be approximated. Parallel gradients, parallel velocity, parallel heat flux and parallel current terms must be approximated for the case of 2D simulations. Two closures that are commonly applied to 3D fluid equations to obtain 2D fluid equations are the sheath dissipation closure and the vorticity advection closure, as applied for example in [76] or first in [28] These closures are briefly explained then applied below.

3.1 Parallel closures

The sheath dissipation closure is one of the most commonly applied closures. The assumption is made that the parallel gradients of scalar quantities are negligible. Sheath currents are assumed to take the usual form by way of linearised Debye currents. This closure is understood to be appropriate for situations where the parallel resistivity is low. Such a scenario is found in filaments that are connected to the sheath and are radially large such that current paths in the drift plane do not dominate. Equations eqs. (63), (65) and (66) are integrated along field lines under these assumptions to arrive at the 2D sheath dissipation equations for GEM.

$$\boldsymbol{\nabla}_\parallel J_\parallel \approx \frac{1}{L_\parallel}\left(\phi - T_{e\parallel} V_F\right) \tag{116}$$

$$\boldsymbol{\nabla}_\parallel u_{i\parallel} \approx \frac{1}{L_\parallel} p_{e\parallel} \qquad \boldsymbol{\nabla}_\parallel u_{e\parallel} \approx \boldsymbol{\nabla}_\parallel u_{i\parallel} - \boldsymbol{\nabla}_\parallel J_\parallel \tag{117}$$

$$\boldsymbol{\nabla}_\parallel q_{i\parallel} \approx \frac{3}{L_\parallel}\left(\tau_i T_{i\parallel}\right) \qquad \boldsymbol{\nabla}_\parallel q_{i\perp} \approx \frac{1}{L_\parallel}\left(\tau_i T_{i\perp}\right) \tag{118}$$

$$\boldsymbol{\nabla}_\parallel q_{e\parallel} \approx \frac{3}{L_\parallel}\left(T_{e\parallel}\right) \qquad \boldsymbol{\nabla}_\parallel q_{e\perp} \approx \frac{1}{L_\parallel}\left(T_{e\perp}\right) \tag{119}$$

Alternatively one can assume that polarisation currents close only through the drift plane and that no currents close through the sheath. This can occur

in the case of a highly resistive plasma, disconnected filaments or filaments with a small radial extent. One assumes that the parallel currents and heat fluxes take values typical of those found at the midplane. As with the sheath closure the variation of scalar quantities along field lines is assumed to be on the order of $L_\parallel$ and equations 63, 65, 66 are integrated along field lines.

$$\boldsymbol{\nabla}_\parallel u_{i\parallel} = \boldsymbol{\nabla}_\parallel u_{e\parallel} \approx \frac{c_s}{2L_\parallel} \tag{120}$$

$$\boldsymbol{\nabla}_\parallel q_{i\parallel} = \boldsymbol{\nabla}_\parallel q_{i\perp} \approx \frac{n_0 \tau_i T_0 c_s}{2L_\parallel} \tag{121}$$

$$\boldsymbol{\nabla}_\parallel q_{e\parallel} = \boldsymbol{\nabla}_\parallel q_{e\perp} \approx \frac{n_0 T_0 c_s}{2L_\parallel} \tag{122}$$

These two closures can be applied alternately to dissipate free energy in the system in ways similar to the parallel dynamics of 3D simulations. The sheath closure is most appropriate for radially large filaments which are connected to the sheath. In such filaments most of the polarisation current can be assumed to close through the sheath. Conversely, the vorticity advection closure is most appropriate for small filaments in which the polarisation current can close through the drift plane because of the short current path available in such cases.

3.2 Isolated filament simulations

Simulations have been carried out using both closure methods in order to study filament propagation under each of these simplifications. When the sheath closure is applied agreement with 3D simulations for large filaments is expected. Conversely, for the vorticity advection closure, agreement between 2D and 3D is expected for small filaments. When holding other parameters constant while varying the filament size in these simulations a scaling relation for peak filament velocity as a function of filament width should be obtained. These relations are well-known in the case of drift-fluid models, as discussed in section 1.3.2. It is expected that GEM, in the drift-fluid limit, should reproduce these scaling laws. If FLR effects significantly impact small filaments, as we suggest they might then when FLR effects are included we expect agreement only in the limit of large filament widths. If the propagation of small

filaments is strongly impacted by FLR effects then a deviation in the inertial limit (concerning small filaments) of the velocity scaling laws should be measured in our FLR simulations.

A 2D slab geometry with the following parameters (listed in table 1) was chosen for both the vorticity advection closure and for the sheath dissipation closure:

Table 1: Parameters chosen for 2D, isolated filament simulations

Input Parameters	Normalisation Parameters	Derived Parameters
$n_x = 256$	$\rho_s = 2.6 \times 10^{-3}\,\mathrm{m}$	$\nu_e = 6.9 \times 10^{-2}$
$n_z = 256$	$c_s = 4.3 \times 10^{4}\,\mathrm{m\,s^{-1}}$	$\nu_i = 1.2 \times 10^{-3}$
$\nu_{\perp N_i} = \nu_{\perp N_e} = 5 \times 10^{-3}$	$\Omega_i = 1.7 \times 10^{7}\,\mathrm{s^{-1}}$	
$T_e = 40\,\mathrm{eV}$		
$T_i = 40\,\mathrm{eV}$		
$N_i = 8 \times 10^{18}\,\mathrm{m^{-3}}$		
$N_e = 8 \times 10^{18}\,\mathrm{m^{-3}}$		
$R = 1.5\,\mathrm{m}$		
$B = 0.5\,\mathrm{T}$		
$l_\parallel = 10\,\mathrm{m}$		

In the case of both the sheath dissipation closure and the vorticity advection closures the domain size was varied. Filaments are initialised with a radius proportional to the domain width and so by varying the domain size so too is the filament width varied. The ratio of the filament width to domain size was kept constant. For each filament width two simulations were carried out. One in the drift-fluid limit with FLR effects excluded and one with FLR effects included through the gyroaverage operators. The propagation characteristics were strongly influenced by the size of the filament, the closure chosen, and in the case of small filaments, on whether FLR effects were included. Some examples of these propagation types are shown below but of most interest is the dependence of peak filament velocity on filament width. This dependence should be of interest to models that attempt to relate filament parameters to radial profiles such as in [79]. Simulations were also carried our with thermal moments included and excluded for each species to isolate thermal effects, if any. Particle and energy sources were chosen such that without any perturbation normalised temperatures and densities were near unity.

Filaments used for these simulations were chosen with Gaussian density

and temperature profiles with a normalised amplitude of $A = 1$. This is a widely used initial condition and will be justified in sections 6.1 and 6.2 using turbulent simulations. A general, simple model for understanding filament propagation in either 2D or 3D cases is as follows. The curvature drive is charge dependent. It therefore drives a charge separation in the filament. An electric field forms as a result of the charge separation which therefore leads to an $\mathbf{E} \times \mathbf{B}$ force. It is the $\mathbf{E} \times \mathbf{B}$ force that leads to the radial propagation of the filament. The curvature drive can also be considered as a current source. The current that arises from this drive term closes through some combination of currents in drift planes or through currents parallel to the $\mathbf{B}$ field. In the case of these 2D simulations the current is assumed to close solely through either sheath currents or through polarisation currents but in principle some combination of parallel and perpendicular currents will be present in 3D.

The filament centre of mass for these 2D simulations is defined as

$$\begin{bmatrix} x_{CoM} \\ z_{CoM} \end{bmatrix} = \frac{\int_0^{L_x} \int_0^{L_z} \begin{bmatrix} x \\ z \end{bmatrix} n\left(x, z, t\right) dx\, dz}{\int_0^{L_x} \int_0^{L_z} n\left(x, z, t\right) dx\, dz} \tag{123}$$

Where n is the density field associated with the filament. This is obtained by a simple thresholding procedure. Once the centre of mass position is obtained the radial and binormal velocity are trivially calculated using a finite difference approximation for the time derivative.

3.2.1 Vorticity-advection simulations

First simulations with the vorticity advection closure applied were considered. Simulations were carried out alternately with 1, or 3 moments included per species giving a total of 2 to 6 moments per simulation. In the case of the 2 moment simulations, which neglect thermal effects the propagation was found to be symmetric about a radial axis centred on the filament initial position. This symmetry only held in the drift limit. When gyroaveraging is enabled, as is the case in our FLR simulations here asymmetry is introduced. This is easily understood by considering that ions and electrons, on the inclusion of gyroaveraging, will experience different electric potentials and hence different $\mathbf{E} \times \mathbf{B}$ drifts as their gyro-orbits are significantly different. This difference in

experienced ($\mathbf{E} \times \mathbf{B}$) velocity drives a charge separation and hence a potential with a monopole component. The monopole component in potential generates spinning in the filament and causes propagation to occur off the symmetry axis. Spinning arising from a monopole potential is common to many filament simulations[80, 76, 81], but not by this mechanism. In 2D drift fluid models the inclusion of sheath currents through the use of a sheath closure gives rise to a monopole component in potential. It is known as Boltzmann spinning and is so named because it is the Boltzmann response that gives rise to the effect. In 3D simulations the parallel gradient term $\nabla_\parallel(\phi_G + \tau_z p_{z\parallel})$ in the electron momentum equation eq. (64) couples the electron density and the potential. This same mechanism is captured in 2D when the sheath closure (eq. (116)) is used through which the electron density can be coupled to the potential. Without an expression for the electron momentum, and without a sheath current or gyroaveraging it is expected that filaments should exhibit symmetric radial propagation without any rotation of the filament about its centre. This is the behaviour seen in fig. 10 the typical "mushrooming" [45] propagation that has long been attributed to filaments is produced. There is a very slight asymmetry in the spiral arms of the filament shown in fig. 10 however this is suspected to be an artefact of initialising the filament with its centre not landing exactly on a grid cell.

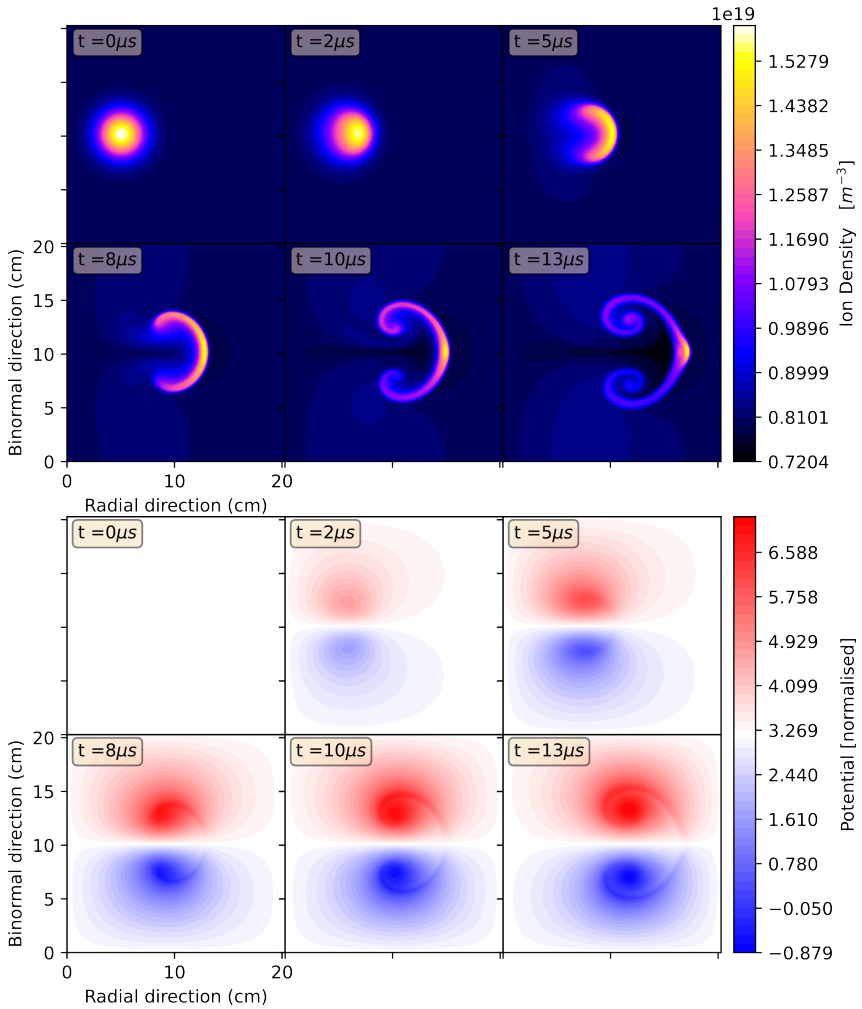

Figure 10: 2D filament simulation in the drift limit and under the vorticity advection closure. The filament exhibits the well-known mushrooming behaviour and the potential takes the form of a symmetric dipole

3.2.2 Sheath-dissipation simulations

Simulations with the sheath dissipation closure exhibit a symmetry break even in the 2-moment drift reduced simulations. This occurs because the sheath current contributes a monopole component to the electric potential. This monopole induces a rotation as described above leading to propagation off the symmetry axis.

Filaments under this closure exhibit two main propagation types. In the case of small filaments fig. 11 the filament remains coherent during propagation. In the case of large filaments such as in fig. 13, they tend to exhibit what has been previously called a "fingering" motion [76]. They are also broken apart by unstable Rayleigh-Taylor like instabilities. This motion is shown

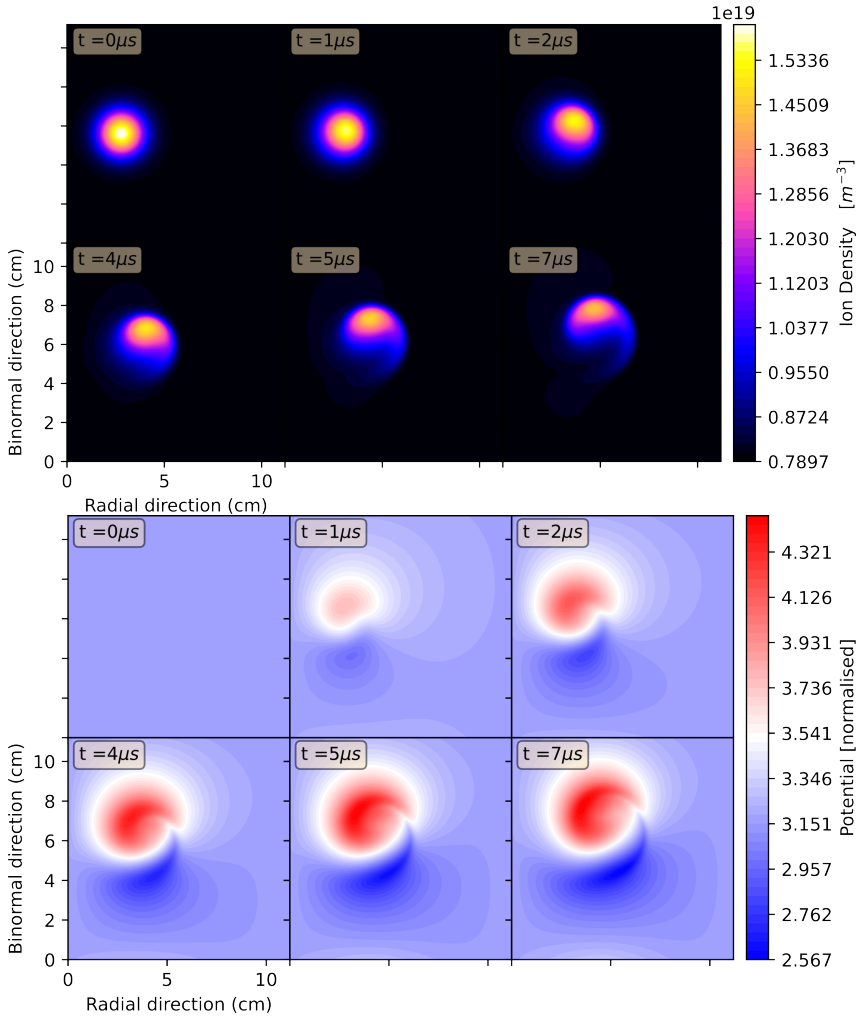

Figure 11: 2D small filament simulation with FLR effects included under the Sheath-dissipation closure exhibiting largely coherent propagation. The mushrooming exhibited in the drift limit is suppressed by a rotation driven by a monopole potential component that arises from gyroaveraging

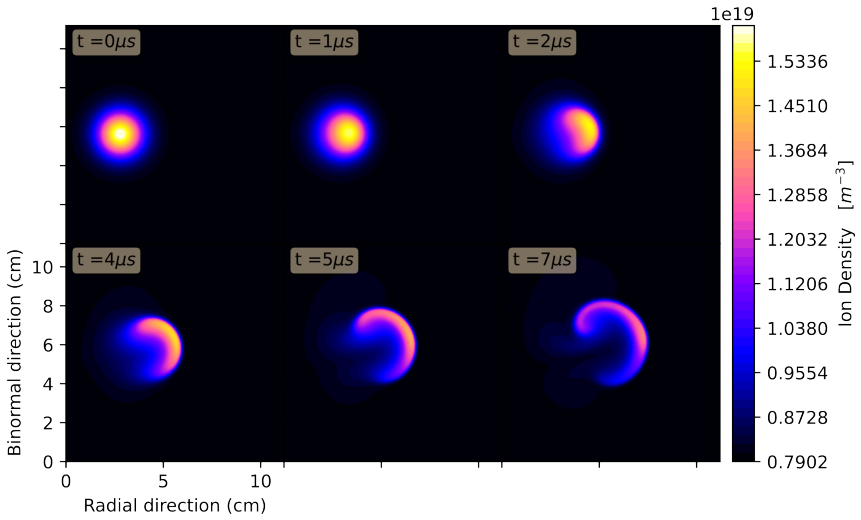

Figure 12: 2D small filament simulation in the drift-fluid limit under the Sheath-dissipation closure exhibiting mushrooming and a rotation arising from the sheath current

in fig. 13

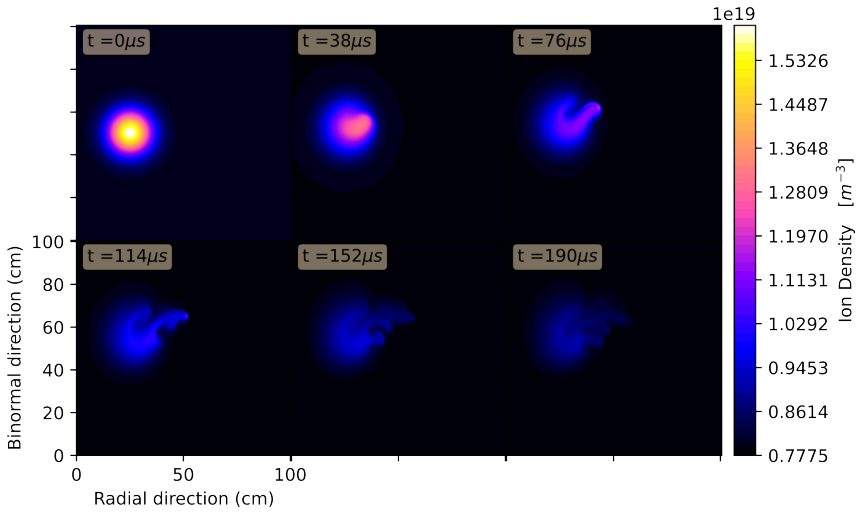

Figure 13: 2D large filament simulation with FLR included under the sheath-dissipation closure exhibiting what is usually called finger propagation. The filament breaks apart into smaller "finger" type structures as it is dissipated

3.2.3 Velocity scaling relations

It is illustrative to consider individual filament simulations as above. However, it is more useful to consider trends in filament propagation. Naturally we hope to recover the well-known filament velocity scaling relations discussed in section 1.3.2 for the drift reduced simulations. We expect that we should observe a deviation from the scaling relations in the case of simulations including FLR effects. The centre of mass for a series of isolated filament simulations is shown in fig. 14. A simple regression is carried out to fit power laws to the peak radial velocities as a function of blob width. The sheath dissipation recovers the scaling relation for sheath-limited filaments very well. Surprisingly it also reproduces the inertially limited velocity scaling relation even though the sheath dissipation closure is not valid for filaments of this size. In the case of inertially-limited filaments a strong reduction in the peak radial velocity is observed when FLR effects are included.

It is clear from fig. 15 and fig. 14 that the sheath-limited scaling relation has been reproduced in the drift limit and also with FLR effects included. This is to be expected as the FLR simulations tend toward the drift-reduced simulations in the limit of small gyro-radius. The sheath-limited relation is obtained after all, for large filaments where the FLR and drift-reduced simulations are almost identical. Importantly, a deviation from the inertial limit is captured. This is

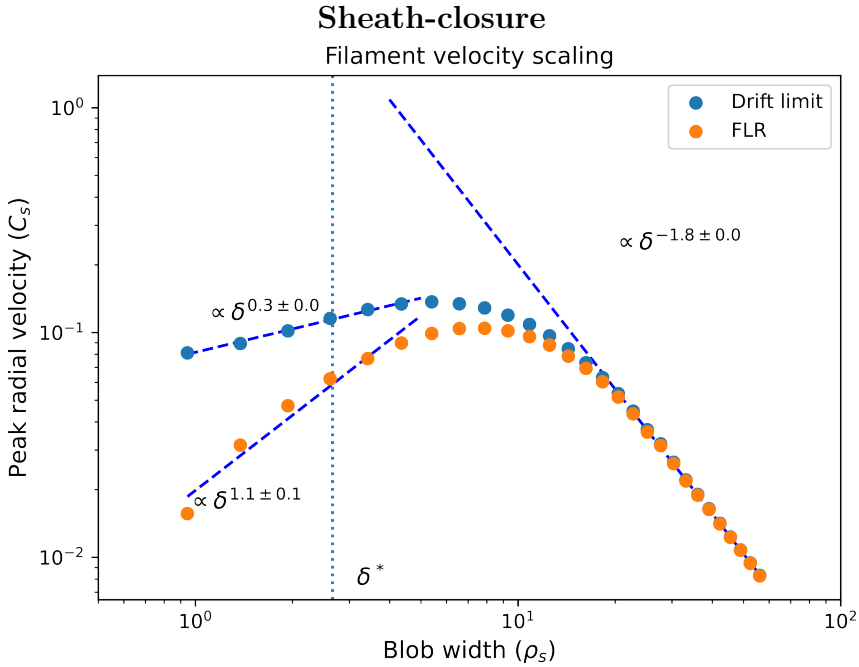

Figure 14: Velocity scaling relation for 2D sheath-closure simulations. The inverse square velocity scaling relation is recovered for both drift-reduced and FLR simulations

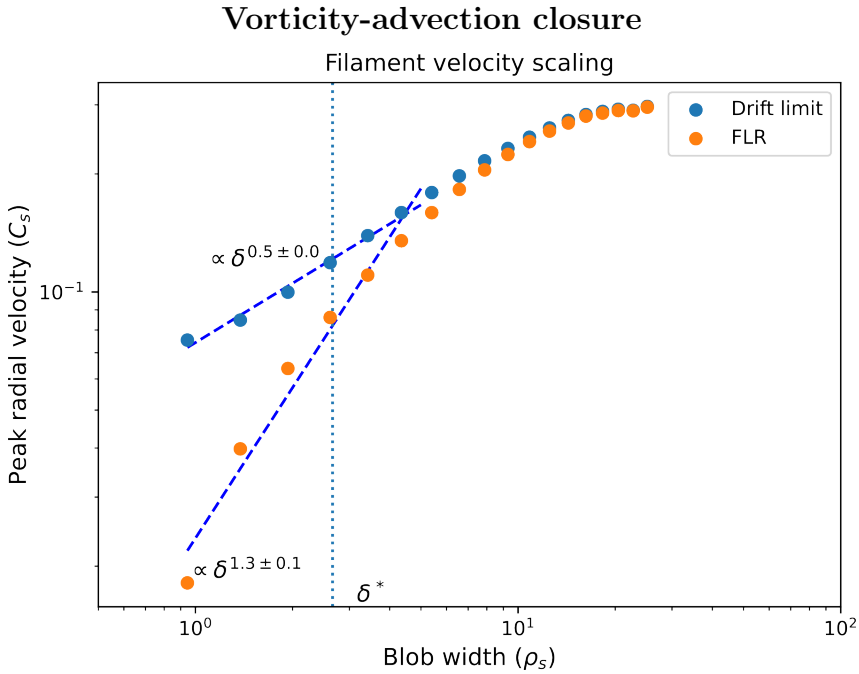

Figure 15: Velocity scaling relation for 2D vorticity-advection simulations. The square-root velocity scaling relation is recovered for the drift-reduced simulations, but there is a clear deviation for FLR simulations

exactly what was suggested should be measured if FLR effects are impactful for small filaments.

This result will be investigated further in 3D simulations in section 4 where parallel dynamics are not approximated as they have been here with closures, but the fact that the scaling relations have been captured here suggests that the closures are operating as intended.

3.3 Grid Resolution Study

In order to ensure our results do not depend on the grid resolution, our grid was systematically refined for both the drift-fluid limit and with FLR effects included. This was repeated for both the sheath closure and the vorticity-advection closure. The parameters used for this study were the same as shown in table 1 except of course for the grid resolution, which was varied. An example of the solutions obtained under successive refinement is shown in fig. 16

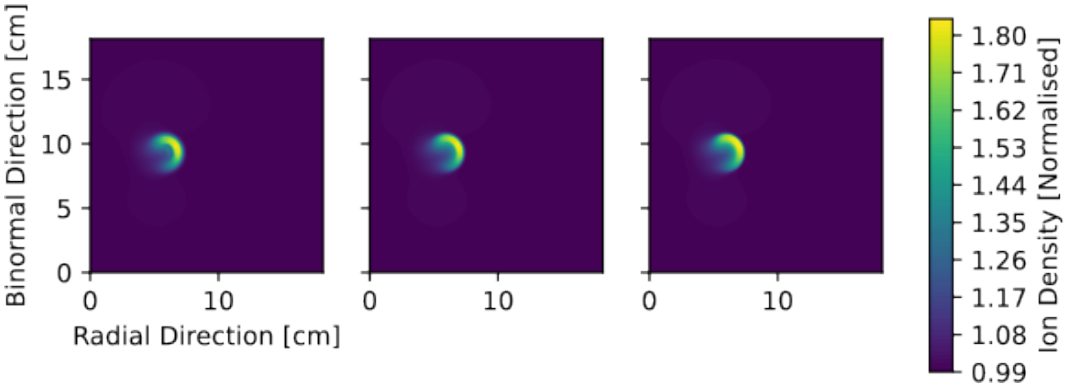

Figure 16: Three simulations of the same filament taken in the drift-limit with the sheath closure applied. From left to right we have a coarse mesh with 128×128, a medium mesh with 256×256 points and a fine mesh with 512×512 points

It is expected the quantities measured should converge as we refine our grid, this follows from our discussion of truncation error in section 2.1. As our grid is refined our truncation error should reduce according to the order of our differencing scheme and approach the 'true' value. In this case we consider the filament velocity as the main quantity to be measured and also the error in the ion density field. When the peak radial filament velocity is plotted against grid resolution N in fig. 17 the expected behaviour is observed. The

measured velocity is dominated by errors on very coarse meshes however, it quickly converges to a constant value on medium and fine meshes. The 2D simulations in section 3.2 used meshes that are within the converged region for the filament velocity.

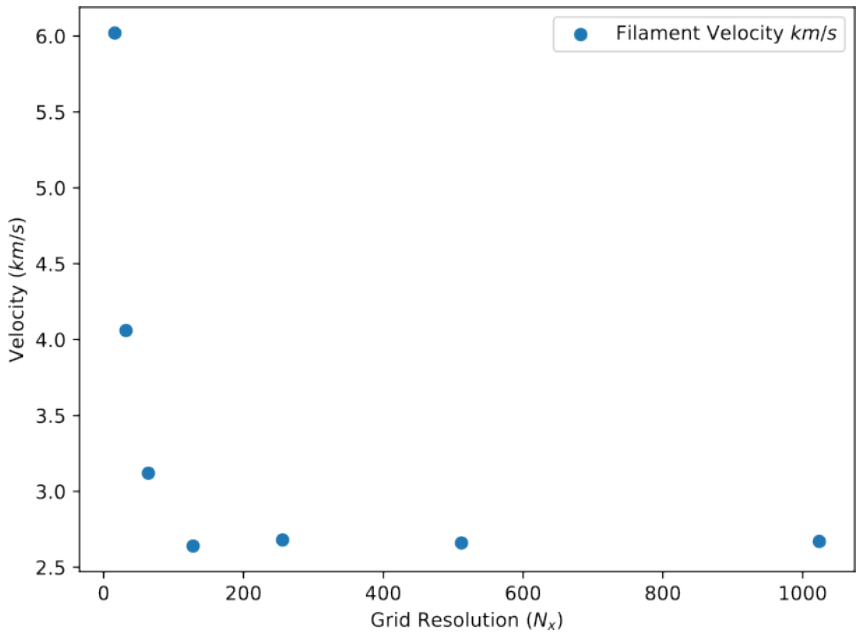

Figure 17: Filament peak radial velocity plotted against grid resolution. As our grid is successively refined our peak velocity converges to a constant value

We now consider solutions for n_i at the time of peak radial velocity. Since we do not have an analytical expression for the ion density after some time t the error ϵ is approximated for each grid as the difference between the solution f on that grid and the solution on the finest grid. The fine grid was downsampled for this calculation so that grid points that corresponded exactly to each other were used without the need for interpolation.

$$\epsilon = f = f_{fine} \tag{124}$$

We then use the discrete, volume-normalised L_2 norm to calculate the error norm for each grid, which for our uniform grid with N points and cell volume dV takes the form:

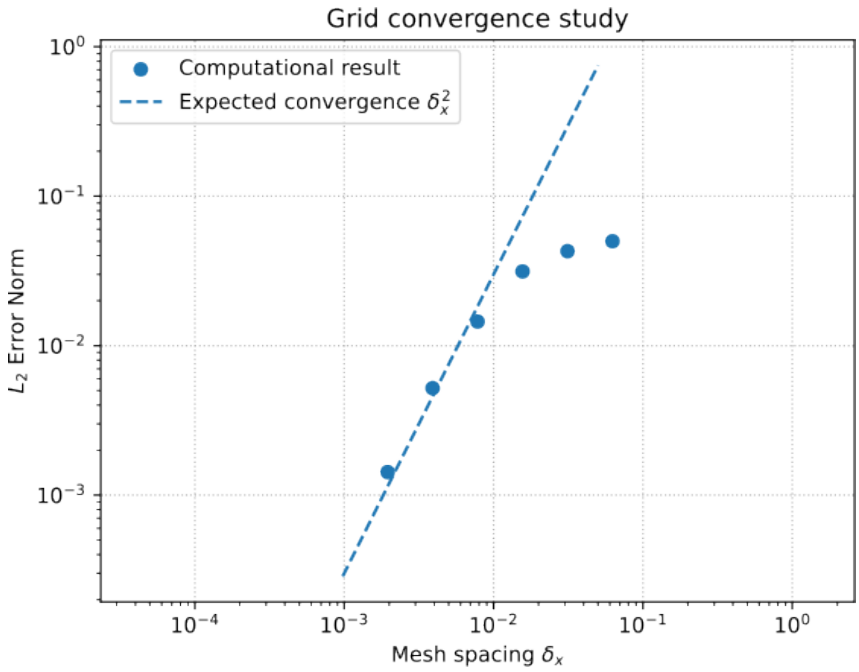

Figure 18: L_2 error norm plotted against mesh spacing δ_x alongside the expected convergence. Our coarsest meshes are not converging as quickly as we expect them to however, our medium and fine meshes converge at approximately the expected rate

$$\|f\|_2 = \sqrt{\sum_i^N \epsilon_i^2 dV} \tag{125}$$

When we plot our error norm against our grid spacing (fig. 17) we see that while our coarse meshes do not converge at the expected rate our fine and medium meshes converge at approximately the expected convergence for our differencing scheme (second order central differencing).

3.4 Filament interactions

Having reproduced filament scaling laws and shown a deviation on inclusion of FLR effects the next scenario in which filaments may be affected by FLR effects is when they are in proximity to each other. A previous study on filament interactions using a drift fluid model concluded that filaments do not interact with each other unless they are in proximity [43]. It was also concluded that they do not interact strongly even when in proximity to each other.

Similar simulations are here presented using the 2D, electrostatic version of GEM. The intention of these simulations is to determine if FLR effects modify filament interactions sufficiently to modify the conclusions drawn by Militello et al. [43], or in fact if the conclusions hold even with the inclusion of FLR effects.

The parameters used for these simulations are almost identical to those used in [43]. These parameters are listed in table 2. The sheath-dissipation closure was used both to match Militello et al. and also because the closure was a good approximation for the 3D dynamics.

Table 2: Parameters chosen for 2D filament interaction simulations

Input Parameters	Normalisation Parameters	Derived Parameters
$n_x = 384$	$\rho_s = 1.8 \times 10^{-3}\,\mathrm{m}$	$\nu_e = 1.1 \times 10^{-1}$
$n_z = 256$	$c_s = 3.1 \times 10^{4}\,\mathrm{m\,s^{-1}}$	$\nu_i = 2.0 \times 10^{-3}$
$L_x = 0.182\,\mathrm{m}$	$\Omega_i = 1.7 \times 10^{7}\,\mathrm{s^{-1}}$	
$L_z = 0.273\,\mathrm{m}$		
$L_\parallel = 10.05\,\mathrm{m}$		
$\nu_{\perp N_i} = 5 \times 10^{-3}$		
$\nu_{\perp N_e} = 5 \times 10^{-3}$		
$T_i = 20\,\mathrm{eV}$		
$T_e = 20\,\mathrm{eV}$		
$N_i = 5 \times 10^{18}\,\mathrm{m^{-3}}$		
$N_e = 5 \times 10^{18}\,\mathrm{m^{-3}}$		
$R = 1.5\,\mathrm{m}$		
$B = 0.5\,\mathrm{T}$		

The filaments were initialised almost identically to those in the drift-fluid simulations cited above. The only difference is the filament amplitude chosen. In the drift-fluid simulations [43] a large normalised filament amplitude of $A = 4$ was chosen. GEM is strictly unsuitable for simulating large amplitude perturbations like this so a more conservative normalised amplitude of $A = 1$

was chosen. This limitation arises from the delta-f nature of GEM. Apart from this all other parameters were matched to facilitate comparison.

Filaments were initialised with the usual Gaussian profile. Sources were chosen such that without perturbation stable solutions at the background densities and temperatures were obtained. Filaments were then added to these backgrounds. Different widths of filaments were considered, widths of 2.5, 5 and 7.5 Larmor orbits were considered. Two configurations were also considered, namely, radially separated and poloidally separated configurations. The separation distance between the filaments was also varied according to a parameter epsilon $\epsilon = [1.5, 2, 3, 5]$. This parameter was used to set the initial position as a multiplier of the sum of filament widths.

For two radially separated filaments A and B initialised at locations $(x_{0,A}, z_{0,A})$ and $(x_{0,B}, z_{0,B})$ with widths w_A and w_B and separation parameter ϵ.

$$\begin{aligned} z_{0,A} &= z_{0,B} = 0.5L_z \\ x_{0,A} &= 0.25L_x \\ x_{0,B} &= x_{0,A} + \epsilon\left(w_A + w_B\right) \end{aligned}$$

And similarly for two poloidally separated filaments A and B

$$\begin{aligned} x_{0,A} &= x_{0,B} = 0.25L_z \\ z_{0,A} &= 0.5L_x + \epsilon\left(w_A + w_B\right)/2 \\ z_{0,B} &= 0.5L_x - \epsilon\left(w_A + w_B\right)/2 \end{aligned}$$

Simulations were again carried out for three different combinations of moment equations included and, in each case, both in the drift limit and with FLR effects included. Snapshots of some hot-ion, hot-electron simulations are shown below.

It is always useful to consider the potential when studying filament interactions. This is all the more true when considering filament interactions. In the case of the radially separated filaments shown in figs. 19 and 20 it is clear that the dipole potential of each individual filament adds together constructively. This means the $\mathbf{E} \times \mathbf{B}$ drive is enhanced and hence both filaments propagate slightly faster than they would have done individually. The trailing filament

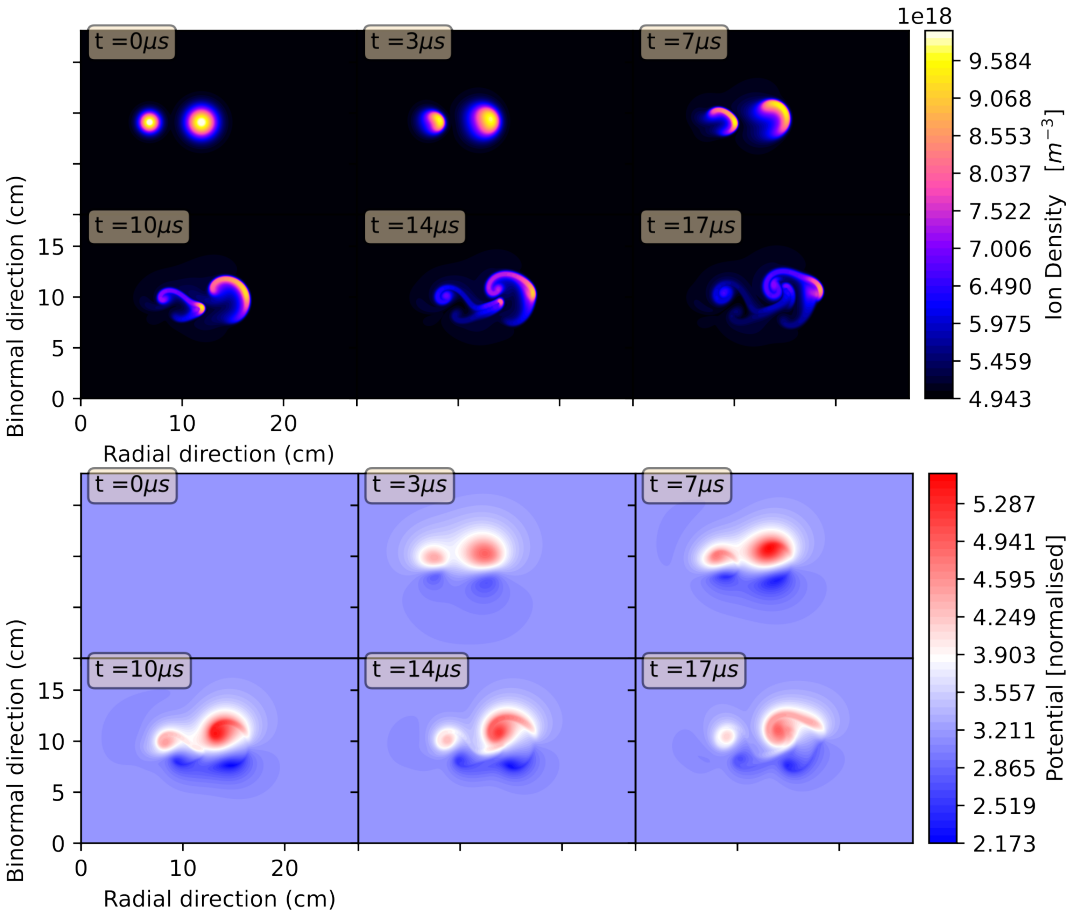

Figure 19: Radially displaced blobs interacting in the drift limit with parameters $(\epsilon, w_A, w_B) = (1.5, 5, 7.5)$. The individual filament dipole potentials add constructively leading to enhanced acceleration of the filaments compared to the isolated case. The rear filament in is accelerated into the wake of the leading filament

in the drift limit case in particular experiences an acceleration into the wake of the leading filament. In the FLR case (fig. 20) the enhanced acceleration still occurs, but the rear filament is no longer swept up into the wake of the leading filament due to the enhanced spinning that can be attributed to gyroaveraging.

The individual dipole potentials in the poloidally separated case clearly destructively combine leading to a diminished field. This means the $\mathbf{E} \times \mathbf{B}$ drive is reduced and hence both filaments propagate slightly slower than they would have done individually. One feature of note is the formation of a current path in the poloidally separated case. The drift-reduced GEM simulation shown in fig. 21 is remarkably similar to the drift-reduced STORM simulation in [43], a current path forms between the two poloidally displaced filaments. The propagation with FLR effects included is still broadly similar to the drift-

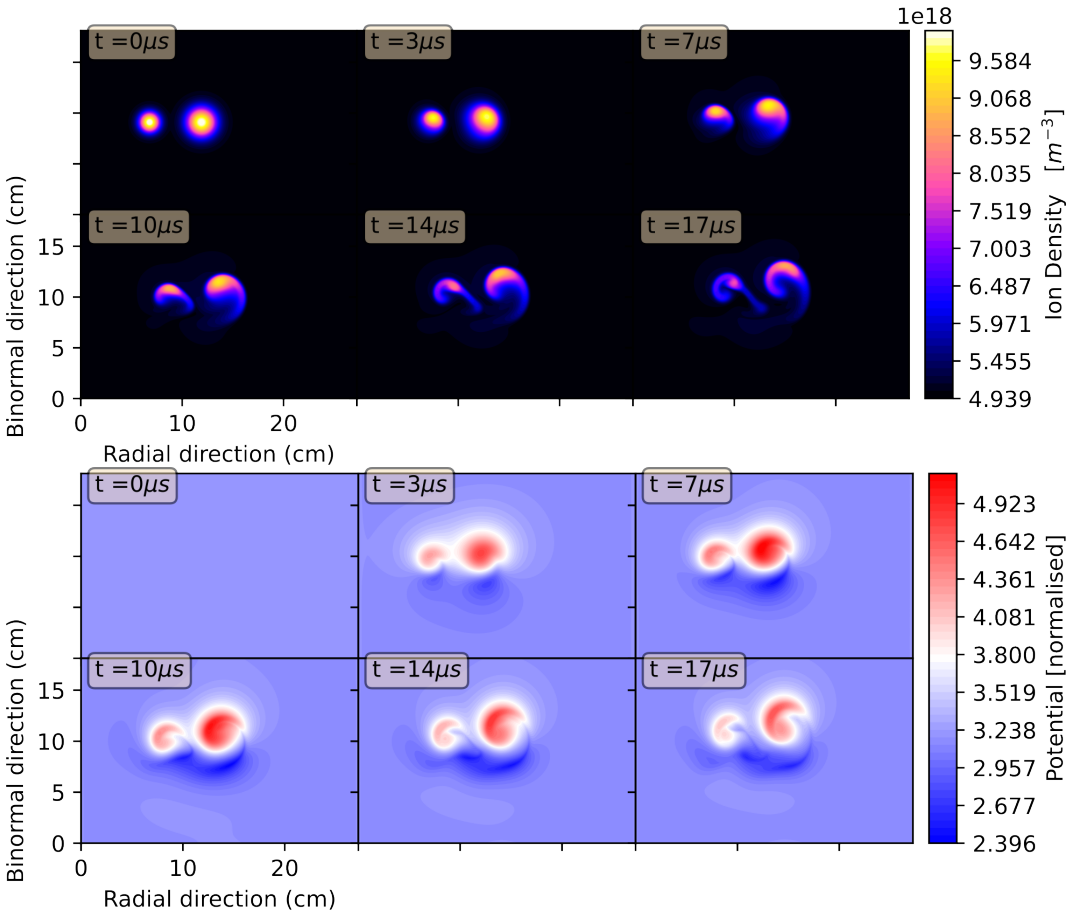

Figure 20: Radially displaced blobs interacting with FLR effects included $(\epsilon, w_A, w_B) = (1.5, 5, 7.5)$ The potential dipoles still add constructively to enhance the acceleration of the filaments however the effect is reduced due to gyroaveraging

reduced case however, the filaments remain more coherent due to an induced spinning.

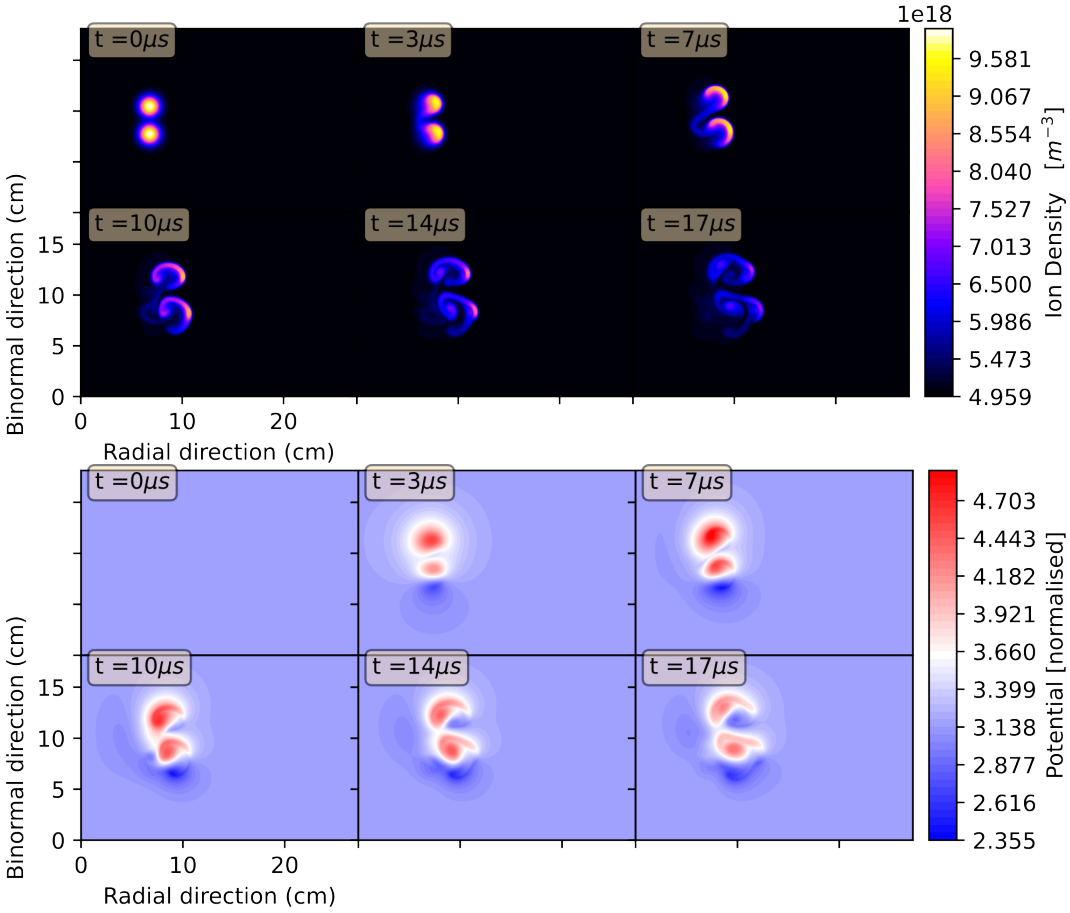

Figure 21: Poloidally displaced blobs interacting in the drift limit $(\epsilon, w_A, w_B) = (1.5, 5, 5)$ The individual dipole potentials from each filament combine destructively and reduce radial propagation. A current path is seen to form linking the two filaments, this is particularly visible at $t = 7us$.

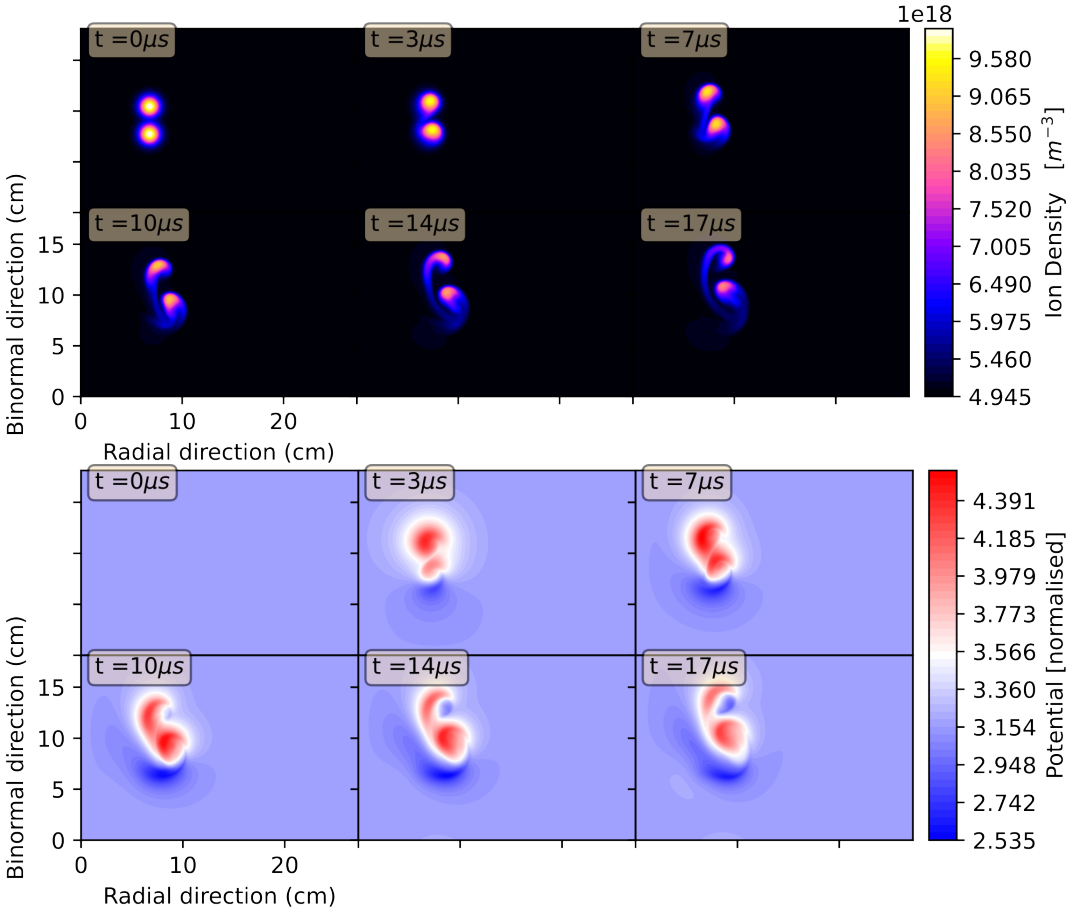

Figure 22: Poloidally displaced blobs interacting with FLR effects included $(\epsilon, w_A, w_B) = (1.5, 5, 5)$ Similar to the drift limit case the individual dipole potentials from each filament combine destructively and reduce radial propagation. A current path still forms between the two filaments.

While it is illustrative to look at individual cases it is more useful to define a metric to apply to each simulation. The same heuristic measure as in [43] is adopted here whereby the centre of mass velocity of the system is calculated for each case and compared to a constructed non-interacting centre of mass velocity. The maximum centre of mass velocity V_{CoM} defined as in eq. (123) is taken as a proxy for a measure of the transport associated with each filament pair.

Similarly, a reference centre-of-mass velocity is constructed by running simulations with each filament individually, summing their density fields and calculating the centre of mass for this non-interacting system.

$$\begin{bmatrix} x_{CoM,ref} \\ z_{CoM,ref} \end{bmatrix} = \frac{\int_0^{L_x} \int_0^{L_z} \begin{bmatrix} x \\ z \end{bmatrix} \left(n_A \left(x, z, t \right) + n_B \left(x, z, t \right) \right) dx \, dz}{\int_0^{L_x} \int_0^{L_z} n \left(x, z, t \right) dx \, dz} \tag{126}$$

These centres of mass are then differentiated using finite difference methods to obtain the centre-of-mass velocity and centre-of-mass reference velocity.

$$V = \begin{bmatrix} V_{xCoM} \\ V_{zCoM} \end{bmatrix} = \frac{\partial}{\partial t} \begin{bmatrix} x_{CoM} \\ z_{CoM} \end{bmatrix} \tag{127}$$

Armed with this metric we can now calculate the maximum centre of mass velocity for each filament pair. The hot-ion, hot-electron results are shown for both drift-reduced simulations and simulations including FLR effects. As was done in [43] simulations where the edge of the simulation domain was reached are excluded. Similarly, simulations where artificial interactions occur through the periodic boundary conditions are excluded. In each of these cases the results can be considered non-physical. Finally, the difference between the centre of mass velocities is defined as:

$$\Delta V = \begin{bmatrix} \Delta V_{xCoM} \\ \Delta V_{zCoM} \end{bmatrix} = 100 \begin{bmatrix} \left(V_{xCoM} - V_{xCoM,ref} \right) / V_{xCoM,ref} \\ \left(V_{zCoM} - V_{zCoM,ref} \right) / V_{zCoM,ref} \end{bmatrix} \tag{128}$$

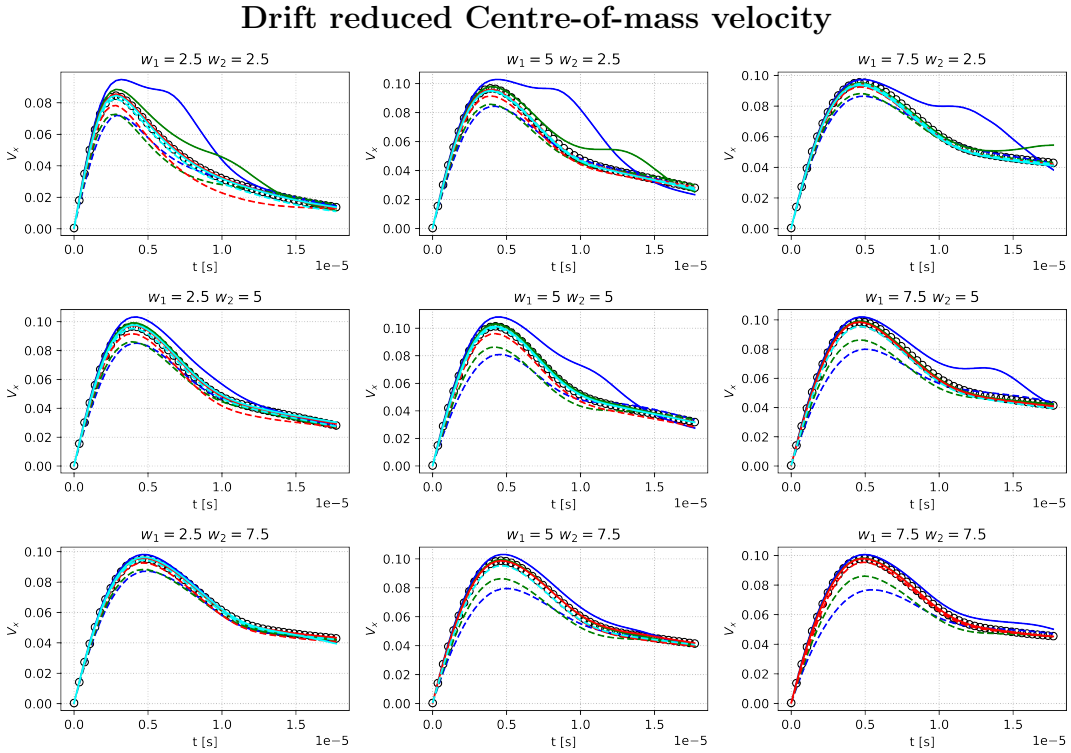

Figure 23: Each subfigure represents a combination of filaments of width w_1, w_2 given in the title. The solid lines represent radially separated filaments, dashed lines represent poloidally separated filaments. The black circles represent the non-interacting reference velocity. Blue, green, red and cyan curves represent ϵ equal to 1.5, 2, 3 and 5 respectively. Solid blue and green lines most often show enhanced velocities compared to the non-interacting reference (black circles). Dashed blue and green lines most often show reduced velocities compared to the non-interacting reference.

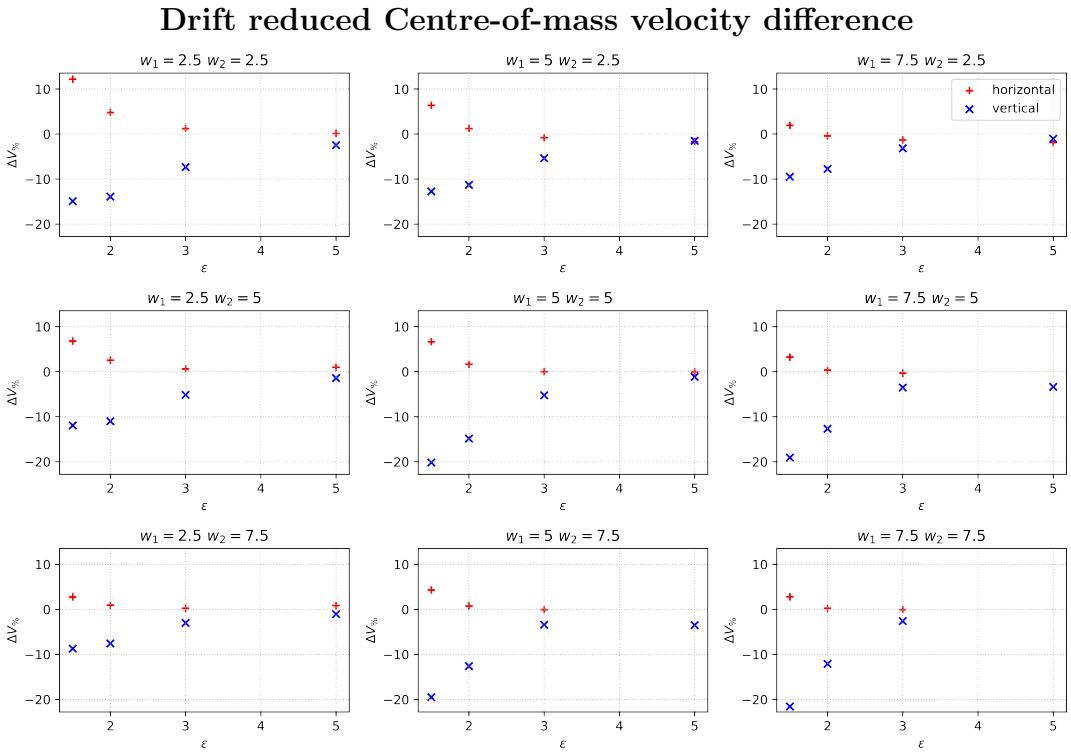

Figure 24: Fractional difference in peak centre-of-mass velocity as a function of separation distance. Blue crosses represent poloidally separated filaments while red pluses represent radially separated filaments. In the case of small separations a relative change in velocity of up to $\approx 20\%$ compared to the non-interacting reference is measured.

Figures 23 and 24 (plotted in this fashion to facilitate comparison with [43]) capture the behaviour we reasoned about for the drift limit simulations when considering the potential for each configuration. Radially separated filaments propagate faster than the reference case while poloidally separated filaments propagate with a diminished velocity. In either case the deviation from the non-interacting reference case tends towards zero with increasing separation distance, as is to be expected. This is the same result obtained in [43]. The similarity is striking, the main difference is that the simulations carried out here predict a reduced maximum deviation. This may be due to the reduced filament amplitude compared to the STORM simulations. In any case the trend is recovered.

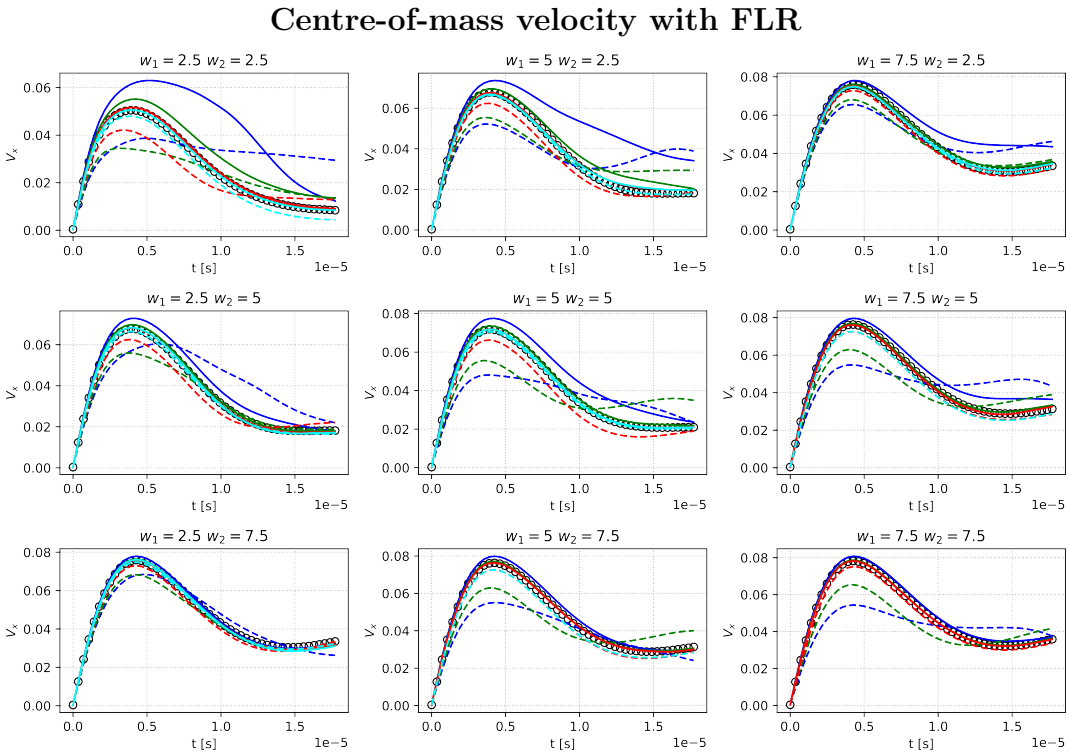

Figure 25: Each subfigure represents a combination of filaments of width w_1, w_2 given in the title. The solid lines represent radially separated filaments, dashed lines represent poloidally separated filaments. The black circles represent the non-interacting reference velocity. Blue, green, red and cyan curves represent ϵ equal to 1.5, 2, 3 and 5 respectively. The trends remain the same as in the drift limit. Solid blue and green lines most often show enhanced velocities compared to the non-interacting reference (black circles). Dashed blue and green lines most often show reduced velocities compared to the non-interacting reference.

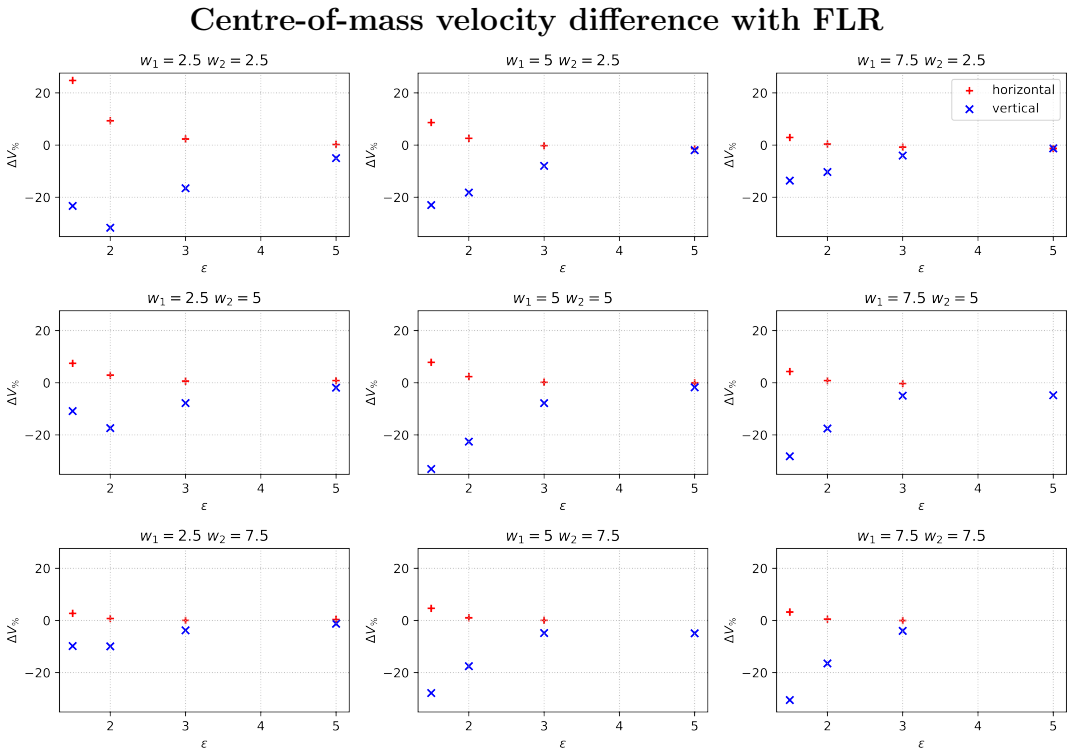

Figure 26: Fractional difference in peak centre-of-mass velocity as a function of separation distance. Blue crosses represent poloidally separated filaments while red pluses represent radially separated filaments. In the case of small separations a relative change in velocity of up to $\approx 30\%$ compared to the non-interacting reference is measured, an increase compared to the drift limit simulations.

With the STORM results replicated with drift-reduced GEM simulations one can then consider what impact, if any, the inclusion of FLR effects through gyroaveraging has. If FLR effects are important to filament interactions then the trends observed in the FLR case should deviate from the drift limit simulations. Figures 25 and 26, clearly show exactly the same trends as the drift-limit case. Radially separated filaments exhibit a slightly enhanced velocity at small separation distances while poloidally separated filaments propagate slightly slower when in proximity. The maximum relative change in velocity for the smallest filaments at the closest separations is increased when FLR effects are included, but the trend is still that the interaction between filaments still tends quickly to zero with increasing separation. This is largely as a result of gyro-reduction of the potential and increased filament coherence. The conclusion then is that FLR effects do enhance the interaction effect for small filaments in proximity to each other but that FLR effects do not strongly

impact the trend predicted in [43].

The reference study [43] found a striking agreement between the sheath dissipation closure 2D simulations for interacting filaments with 3D filament simulations. Since FLR effects are included through gyroaveraging in drift planes it is expected than any correction to the drift fluid results should in this case be found in 2D, as in the simulations presented above. As such, it is unlikely that 3D GEM simulations would further modify the result of weakly-interacting filaments and, we can conclude that the conclusions drawn in [43] could equally be drawn with GEM, with FLR effects included. Filaments do not strongly interact with each other, even though the interaction is enhanced under the inclusion of FLR effects and this interaction only occurs over short ranges on the order of 3 blob widths.

4 3D Filament simulations

Filaments are fundamentally 3D phenomenon as discussed in section 1.3. 3D simulations are therefore required to properly capture their dynamics. Appropriate parallel closures can capture the dynamics in certain regimes reasonably well [45] and allow filaments to be simulated in a 2D domain, thereby reducing the computational expense greatly. We have thoroughly explored such 2D simulations in section 3. However, an appropriate parallel closure must be chosen to match the dominant current path through which the curvature-driven diamagnetic current closes for a given simulation. Therefore, capturing filament dynamics over various spatial scales necessitates a 3D treatment of the system. As such, 3D, electrostatic GEM simulations of isolated filaments follow with various combinations of moments included for each species. These simulations are the 3D counterparts to those carried out in section 3.2.

4.1 1D Background simulations

In order to simulate filaments in a 3D slab as we intend to in this section the system must first reach a 1D equilibrium solution. The equilibrium solution for these 3D simulations is obtained using a 1D simulation where we assert that there are no gradients perpendicular to B. Only gradients parallel to the $\mathbf{B}$ field are non-zero in these simulations. A source term intended to represent recycling at the target and a corresponding momentum conservation term have been added to our moment equations as done in [45]. We now take our moment equations defined in section 1.7.1 and set the perpendicular gradients to zero and set the parallel gradient terms equal to the sources.

$$\frac{\mathrm{d}n_z}{\mathrm{d}t} = -B\nabla_\parallel \frac{u_{z\parallel}}{B} + S \tag{129}$$

$$\mu_z \frac{\mathrm{d}u_{z\parallel}}{\mathrm{d}t} = -\nabla_\parallel(\phi_G + \tau_z n_z) - \mu_z \frac{S u_{z\parallel}}{n_z} \tag{130}$$

When looking for equilibrium solutions to eqs. (129) and (130) the time derivative goes to zero.

$$B\nabla_{\parallel}\frac{u_{z\parallel}}{B} = S \tag{131}$$

$$\nabla_{\parallel}(\phi_G + \tau_z n_z) = -\mu_z \frac{S u_{z\parallel}}{n_z} \tag{132}$$

The isothermal, electrostatic solutions are then trivially found by assuming quasineutrality at equilibrium and that no equilibrium current exists. The parallel velocity profile at equilibrium for a given density source can be found using eq. (129).

$$S(y) - 1.0\frac{d}{dy}\,\mathrm{u}_{\parallel}(y) = 0 \tag{133}$$

$$u_{\parallel} = \int S(y)\,dy \tag{134}$$

Similarly, combinations of eq. (130) for electrons and ions can be used to find the parallel density and potential profiles. First we solve for the density by taking the difference of eq. (130) for electrons and ions.

$$\frac{\mu_e S(y)}{n(y)}\,\mathrm{u}_{\parallel}(y) - \frac{\mu_i S(y)}{n(y)}\,\mathrm{u}_{\parallel}(y) + 1.0\tau_e\frac{d}{dy}n(y) - 1.0\tau_i\frac{d}{dy}n(y) = 0 \tag{135}$$

$$\implies n(y) = \sqrt{\frac{2}{\tau_e - \tau_i}\left(C_1 - \mu_e\int S(y)\,\mathrm{u}_{\parallel}(y)\,dy + \mu_i\int S(y)\,\mathrm{u}_{\parallel}(y)\,dy\right)} \tag{136}$$

And then we solve for the potential by taking the sum of eq. (130) for electrons and ions.

$$\mu_e\left(-\frac{\mu_i S(y)}{n(y)}\,\mathrm{u}_{\parallel}(y) - 1.0\tau_i\frac{d}{dy}n(y) - 1.0\frac{d}{dy}\phi_G(y)\right) \tag{137}$$

$$-\mu_i\left(-\frac{\mu_e S(y)}{n(y)}\,\mathrm{u}_{\parallel}(y) - 1.0\tau_e\frac{d}{dy}n(y) - 1.0\frac{d}{dy}\phi_G(y)\right) = 0$$

$$\implies \phi_G(y) = \frac{1.0}{\mu_e - \mu_i}\left(C_2 - 1.0\mu_e\tau_i n(y) + 1.0\mu_i\tau_e n(y)\right) \tag{138}$$

The density source is chosen such that it is localised to the end of the

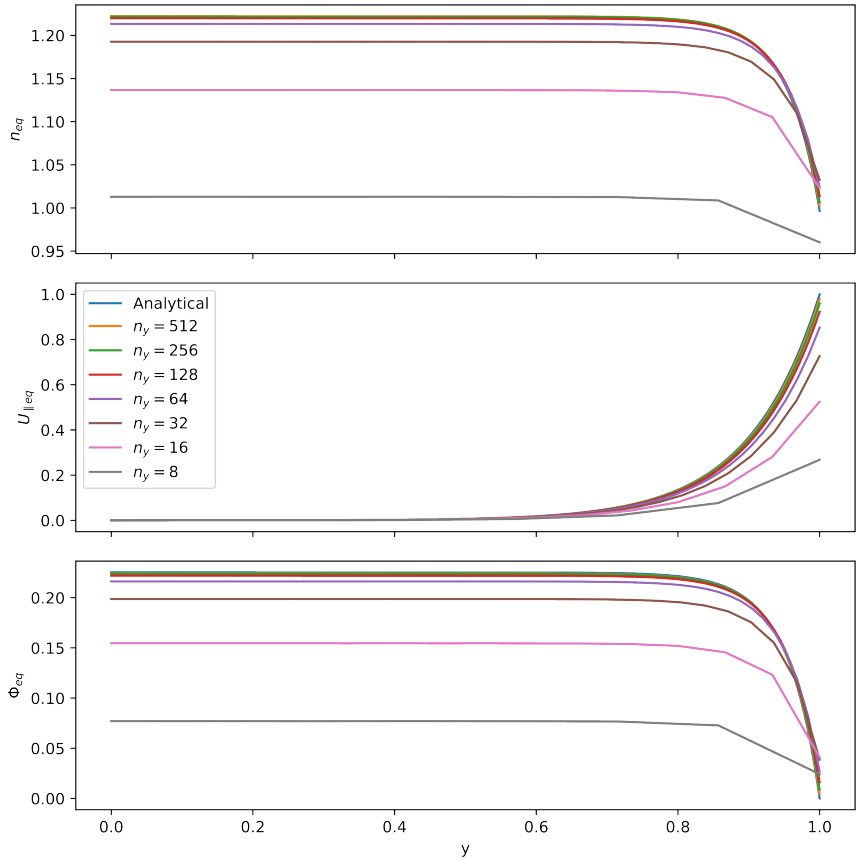

Figure 27: 1D Equilibrium profiles produced for various parallel resolutions alongside the analytical results. Note: Analytical result obscured by $n_y = 512$ case

domain nearest to the target similarly to [45]

$$S(y) = 10\frac{\exp\{10y\}}{L_{\parallel}\exp\{10\}} \tag{139}$$

The constant of integration is set for the parallel velocity such that the velocity at the midplane is zero, as is required to maintain symmetry. Similarly, the constant of integration for the potential is set such that the floating potential is reached at the target. When the constant of integration for the analytical density solution is chosen to match a high resolution numerical result then the numerical potential and velocity profiles match the analytical results.

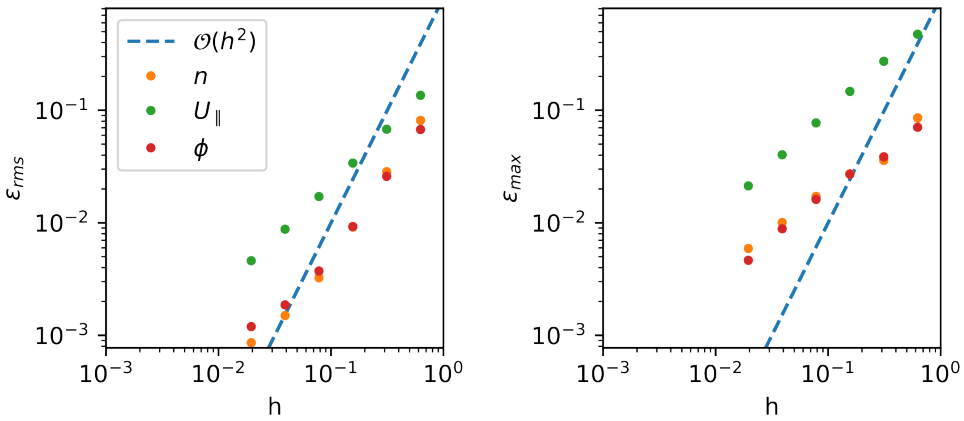

Figure 28: RMS error (left) and maximum error (right) plotted for various parallel grid spacings. The dashed line denotes $\mathcal{O}(h^2)$ convergence

The error from the analytical solution was calculated and plotted for each of the parallel grid spacings used in fig. 28. The convergence order is then calculated using both the root-mean-square (RMS) value and the absolute maximum value of the error. It is clear from fig. 28 that the numerical results are converging to the analytical solutions. However, since a second order central differencing scheme was used to approximate gradient operators one would expect the solution to converge to second order since the error in a second order central differencing scheme should be $\mathcal{O}(h^2)$ where h is the grid spacing. This is not the convergence order observed, however, the same behaviour was identified in [76] and attributed to a discontinuous derivative at the sheath boundary. The same convergence issue is occurring here.

4.2 Background control for 3D simulations

While the background can be solved for analytically as above for a given set of particle and energy sources as shown above this approach is impractical for 3D simulations where 6 moments are solved for both ions and electrons. Instead, a different control method is applied to find exact stable background solutions within specified error bounds. Namely, 1d simulations are carried out with PID (proportional–integral–derivative) controllers applied to the source terms. The particle source $S_{Ni,e}$, perpendicular $S_{T\perp i,e}$, and parallel $S_{T\parallel i,e}$ energy source terms take the form where $A_{Ni,e}$, $A_{T\perp i,e}$, and $A_{T\parallel i,e}$ are the scalar amplitudes of each source:

$$S_{Ni,e} = A_{Ni,e} * \exp\left(10(y-1)\right) \tag{140}$$

$$S_{T\perp i,e} = A_{T\perp i,e} * \exp\left(-5y\right) \tag{141}$$

$$S_{T\parallel i,e} = A_{T\parallel i,e} * \exp\left(-5y\right) \tag{142}$$

The density sources take the form of recycling sources and are localised to the vicinity of the sheath and the energy source is localised upstream. The upstream value of the scalar field associated with a given source is fed into the PID controller as the controller input with a target of a normalised value of 1 for each variable.

The controller then iteratively varies the source coefficients according to the following scheme until a solution within given error-bounds is found

$$A = \begin{bmatrix} A_{Ni} \\ A_{T\perp i} \\ A_{T\parallel i} \\ A_{Ne} \\ A_{T\perp e} \\ A_{T\parallel e} \end{bmatrix} \qquad I = \begin{bmatrix} N_i \\ T_{\perp i} \\ T_{\parallel i} \\ N_e \\ T_{\perp e} \\ T_{\parallel e} \end{bmatrix}_{y=0} \qquad SP = \begin{bmatrix} N_{i0} \\ T_{\perp i0} \\ T_{\parallel i0} \\ N_{e0} \\ T_{\perp e0} \\ T_{\parallel e0} \end{bmatrix} \tag{143}$$

$$E = I - SP \tag{144}$$

$$u\left(t\right) = K_p E\left(t\right) + K_i \int_0^t E\left(t\right) dt + K_d \frac{\mathrm{d}}{\mathrm{d}t} E\left(t\right) \tag{145}$$

$$A \mathrel{+}= u\left(t\right) \tag{146}$$

Where A is an array of source coefficients, I is an array of process variables and SP is an array of setpoints. The output $u(t)$ is propagated back into the control variables iteratively until a solution is found. K_p, K_i and K_d are tuning parameters that control the proportional, integral and derivative gain respectively.

This optimisation process is complicated by the fact that the source coefficients don't just affect their primary process variable. Since the moment equations are of course, all coupled. Changing one source term affects all the process variables. As such, a conservative tune was applied to the PID control-

lers, relying mainly on the small integral term to handle adjustment so that no large changes would be made which could prevent a stable solution being found.

This controller was used to find source terms that produced the correct backgrounds for 3D filament simulations, except for core-SOL simulations. The controller was not applied during the 3D simulations as it is only necessary to control the sources to find stable, stationary backgrounds on top of which to initialise filaments.

An example of the PID controller searching for a stable background is shown in fig. 29 and the resulting background is found in fig. 30 While the controller appears to initially overshoot the density setpoint this is a result of other process variables being initialised relatively far from a stable solution.

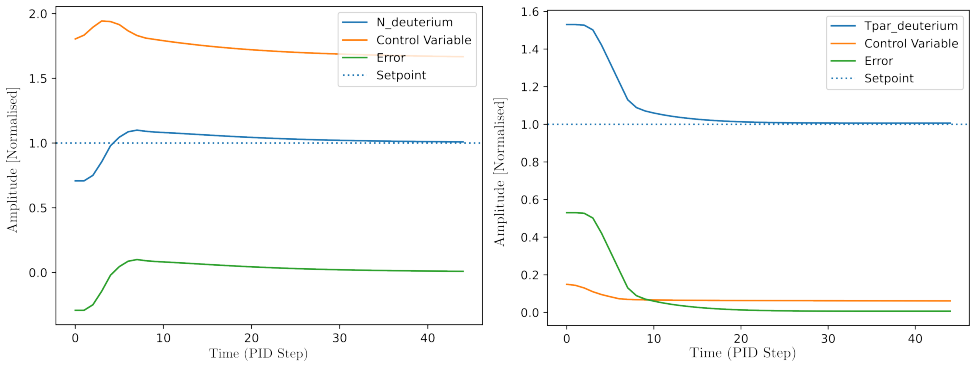

Figure 29: PID controller monitor for N_i and $T_{\|i}$, $y = 0$ is the midplane, the sheath is at $y = 10$

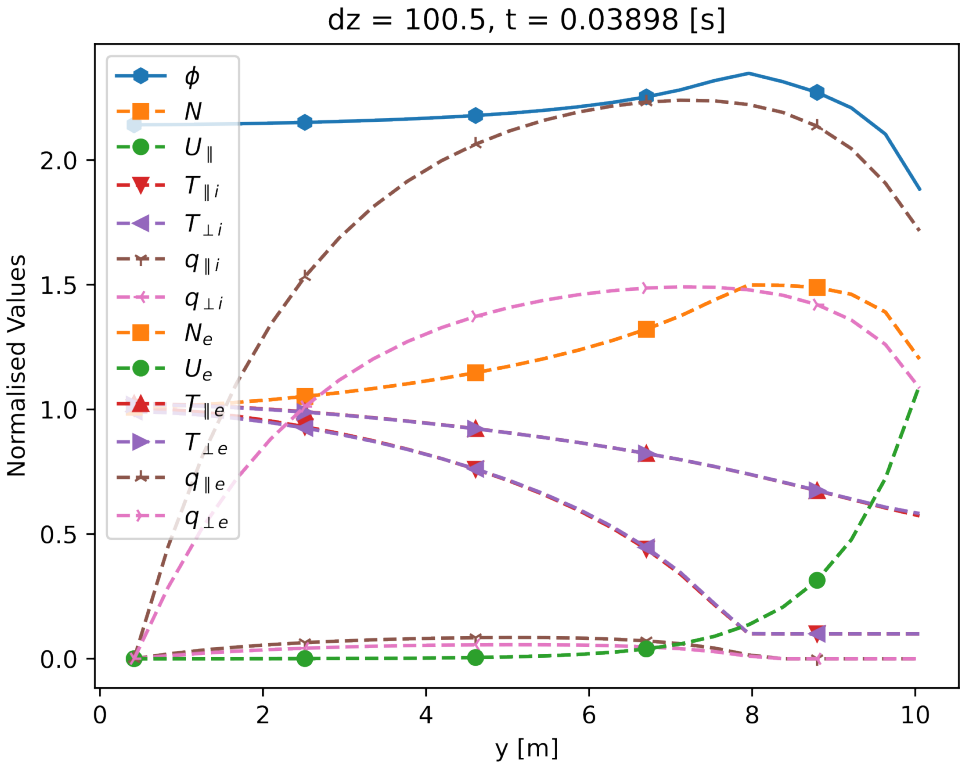

Figure 30: Representative stable background suitable for the addition of filaments for 3D simulations. Densities and temperatures all take normalised values of 1 upstream of the target

Once a background is obtained for a given set of input parameters, as described above, one or more filaments are added as perturbations to the background. These filaments take the usual form adopted by fluid models when studying filament dynamics as in [45],[76],[44],[43], and [82], among others. Specifically the filaments are initialised with a Gaussian profile in the drift plane and a hyperbolic tangent (Tanh) profile in the parallel direction. The parallel length, tip steepness, perpendicular width and amplitude are the variables that parameterise the filaments used here.

One potential pitfall here is that when FLR effects are included, the particle positions and gyrocentre positions are, by definition, not co-located. This means that in order to ensure filaments are initialised with zero initial vorticity the electron gyrocentre density should be set according to eq. (147) or some other equivalent expression that ensures zero initial vorticity.

$$N_e = \Gamma_1 N_i - \Gamma_2 T_{\perp e} + \Gamma_2 T_{\perp i} \tag{147}$$

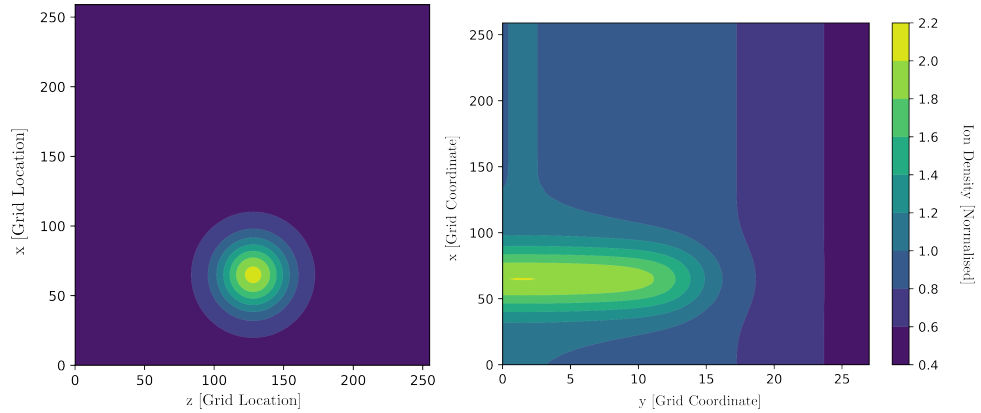

Figure 31: Filament profiles in the drift plane (left) and parallel direction (right). A half connected filament is shown here.

$$\begin{bmatrix} x_{CoM} \\ y_{CoM} \\ z_{CoM} \end{bmatrix} = \frac{\int_0^{L_x}\int_0^{L_y}\int_0^{L_z} \begin{bmatrix} x \\ y \\ z \end{bmatrix} n_f\left(x,y,z,t\right) dx\, dy\, dz}{\int_0^{L_x}\int_0^{L_y}\int_0^{L_z} n_f\left(x,y,z,t\right) dx\, dy\, dz} \tag{148}$$

When analysing filament motion the centre of mass is calculated using eq. (148). n_f is the density field associated with the filament. This density field is obtained by thresholding to isolate the filament from the background in a manner similar to [45]. The centre of mass is then differentiated using finite difference methods to obtain the filament centre-of-mass velocity.

$$V = \begin{bmatrix} V_{xCoM} \\ V_{yCoM} \\ V_{zCoM} \end{bmatrix} = \frac{\partial}{\partial t} \begin{bmatrix} x_{CoM} \\ y_{CoM} \\ z_{CoM} \end{bmatrix} \tag{149}$$

This centre of mass velocity can be calculated for a range of filament sizes to obtain scaling relations describing filament centre-of-mass velocity as a function of filament width.

The parameters chosen for the 3D isolated filament simulations are given in table 3

Table 3: Parameters chosen for 3D isolated filament simulations

Input Parameters	Normalisation Parameters	Derived Parameters
$n_x = 256$	$\rho_s = 2.6 \times 10^{-3}\,\mathrm{m}$	$\nu_e = 6.9 \times 10^{-2}$
$n_y = 24$	$c_s = 4.3 \times 10^{4}\,\mathrm{m\,s^{-1}}$	$\nu_i = 1.2 \times 10^{-3}$
$n_z = 256$	$\Omega_i = 1.7 \times 10^{7}\,\mathrm{s^{-1}}$	
$\nu_{\perp N_i} = 5 \times 10^{-3}$		
$\nu_{\perp N_e} = 5 \times 10^{-3}$		
$T_e = 40\,\mathrm{eV}$		
$T_i = 40\,\mathrm{eV}$		
$N_i = 8 \times 10^{18}\,\mathrm{m^{-3}}$		
$N_e = 8 \times 10^{18}\,\mathrm{m^{-3}}$		
$R = 1.5\,\mathrm{m}$		
$B = 0.5\,\mathrm{T}$		
$l_\parallel = 10\,\mathrm{m}$		

The boundary conditions for isolated filament simulations are those introduced in section 2.3.5. Standard linearised Debye sheath boundary conditions [69, 68] are applied at the target which is located at $y = L_\parallel$. Symmetry boundary conditions are applied at the midplane which is located at $y = 0$. The z boundaries are periodic as they were previously. Neumann boundary conditions are applied at x boundaries.

$$u_{\parallel i} = p_{\parallel e} \qquad u_{\parallel e} = u_{\parallel i} - (\phi - \Lambda T_{\parallel e}) \tag{150}$$

$$q_{\parallel e} = 3T_{e\parallel} \qquad q_{\perp e} = T_{\perp e} \tag{151}$$

$$q_{\parallel i} = 3\tau_i T_{i\parallel} \qquad q_{\perp i} = \tau_i T_{\perp i} \tag{152}$$

Where Λ is the floating potential.

4.3 Cold-electron, Cold-ion simulations

The simplest 3D filament simulation that we can consider with this implementation of GEM is the cold-ion, cold-electron case where only the density (eq. (63)) and velocity (eq. (64)) moments for each species are evolved. Before isolated filaments can be considered a stable background is first required. The PID controllers described in section 4.1 allow us to find such a background for the chosen input parameters. That background is shown in fig. 32.

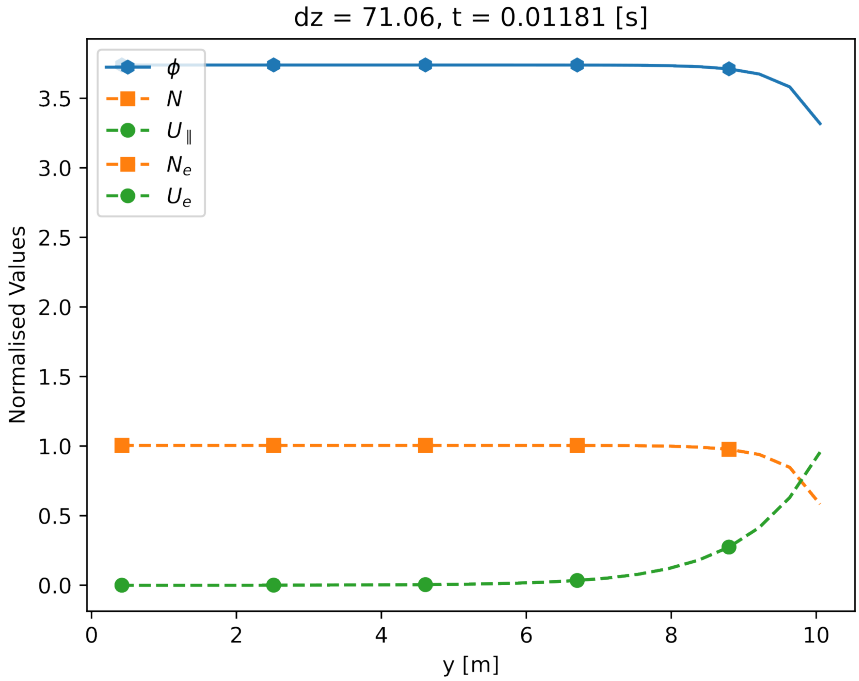

Figure 32: Stable background onto which filaments are superimposed for cold ion, cold electron simulations

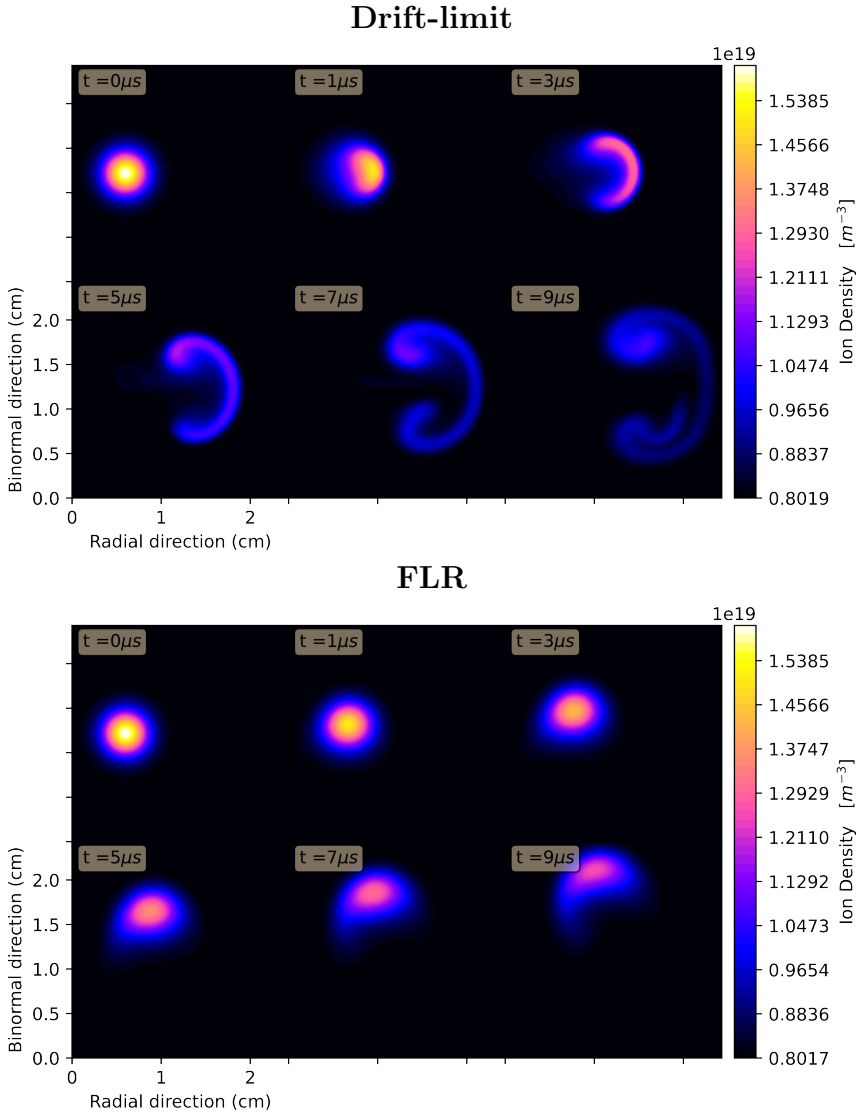

Figure 33: In the case of small filaments an enhanced initial spinning motion is observed, compared to 2D simulations both with FLR included and without. This spinning is attributed to the sheath currents as described in section 3.2.2

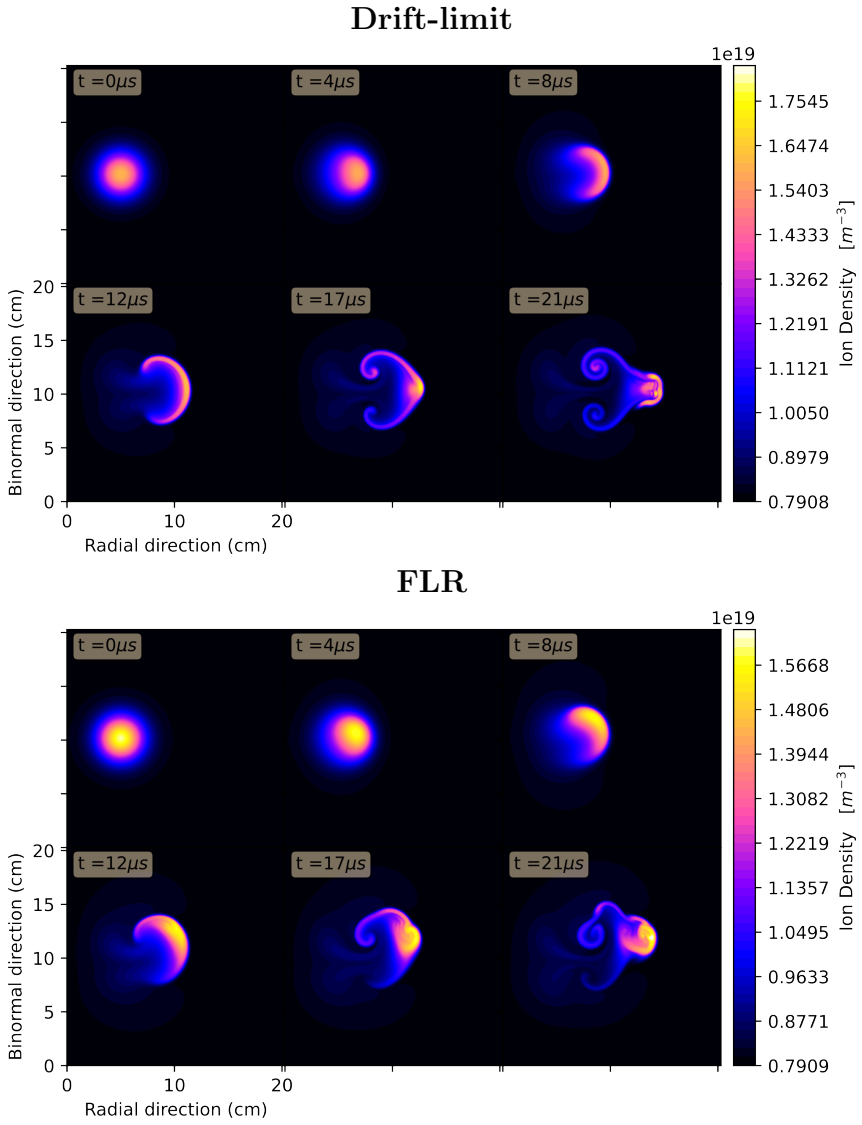

Figure 34: For intermediate sized filaments the typical largely coherent "mushrooming" motion is observed. The FLR filament shows an enhanced spinning due to gyroaveraging as was seen in section 3.2

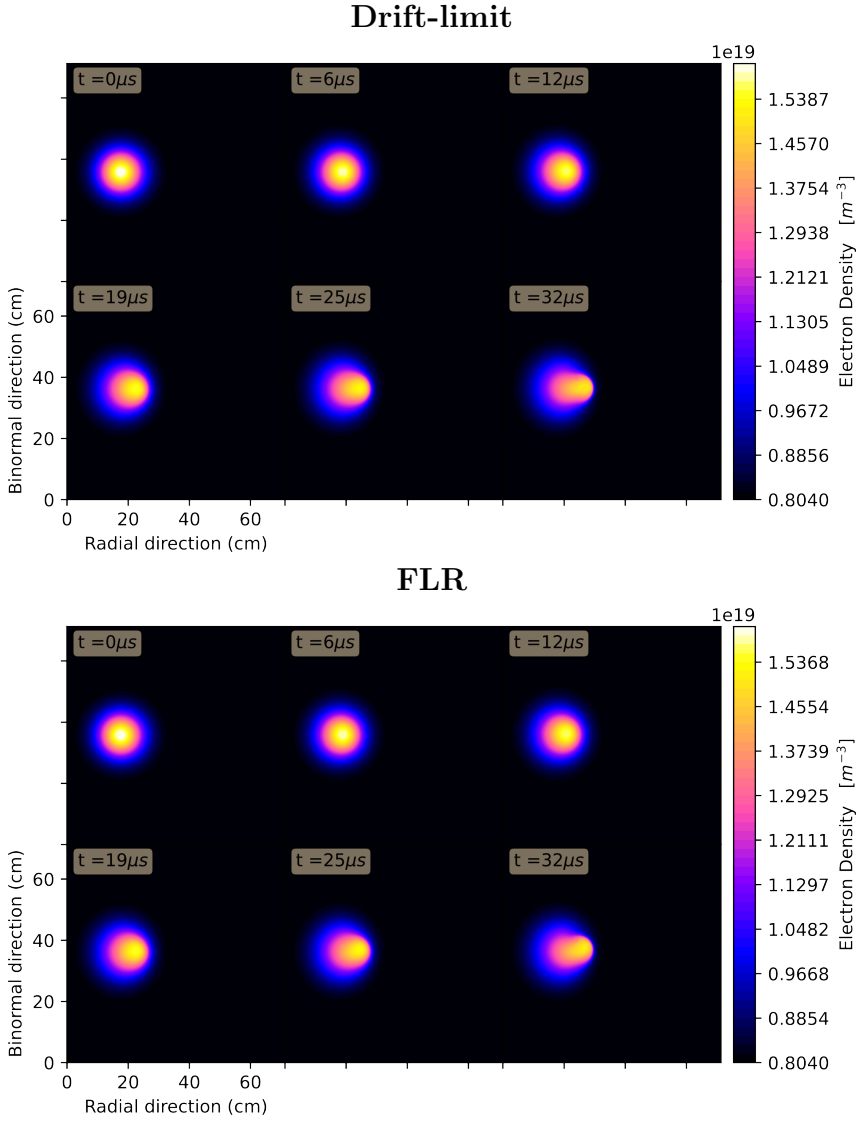

Figure 35: For larger filaments the filament remains mostly coherent and does not move much before reaching its peak velocity both when FLR effects are included and when they are not. There is a slightly enhanced monopole component of the potential in the case where FLR effects are included which leads to slight rotation and binormal propagation as compared to the drift-limit.

As in the 2D simulations, more interesting than the modification to individual filament dynamics is the influence FLR effects have on filaments over a wide range of sizes. As explained in section 1.3.2 and section 5, and demonstrated in section 3.2 it is expected that the filament centre of mass velocity should scale with $\delta^{0.5}$ in the case of blobs in the inertial regime. In the case of blobs in the sheath limited regime the velocity should scale with δ^{-2}. Small blobs are expected to be in the inertial regime because the current arising from the curvature drive can close easily through polarisation currents since the current path is short. Conversely, large filaments are expected to be sheath-limited as the current closes through the sheath. These velocity scaling relations have been observed in drift fluid models previously [45, 83].

Filament velocity calculated from the simulations described above is plotted vs filament width in fig. 37. Power laws are fit to the data to extract the scaling law. The exponents calculated when FLR effects are not included, match the analytical scaling laws, as expected for what is effectively a drift fluid model. When FLR effects are included however, the trend for small filaments in the inertial regime is broken. The reason for this deviation from the trend is investigated in more detail in section 5. But broadly, this deviation arises due to a gyro-reduction of the potential experienced by ions and hence a corresponding gyro-reduction of the $\mathbf{E} \times \mathbf{B}$ velocity for small filaments. The width for which the peak filament velocity is reached is also shifted slightly towards larger filaments.

The simulations were repeated for a series of half-connected filaments that do not reach all the way to the downstream target and corresponding sheath. The trends in maximum filament velocity do not change substantially when going from connected filaments to half connected as shown in fig. 37. There is a reduction in peak velocity in the half connected case

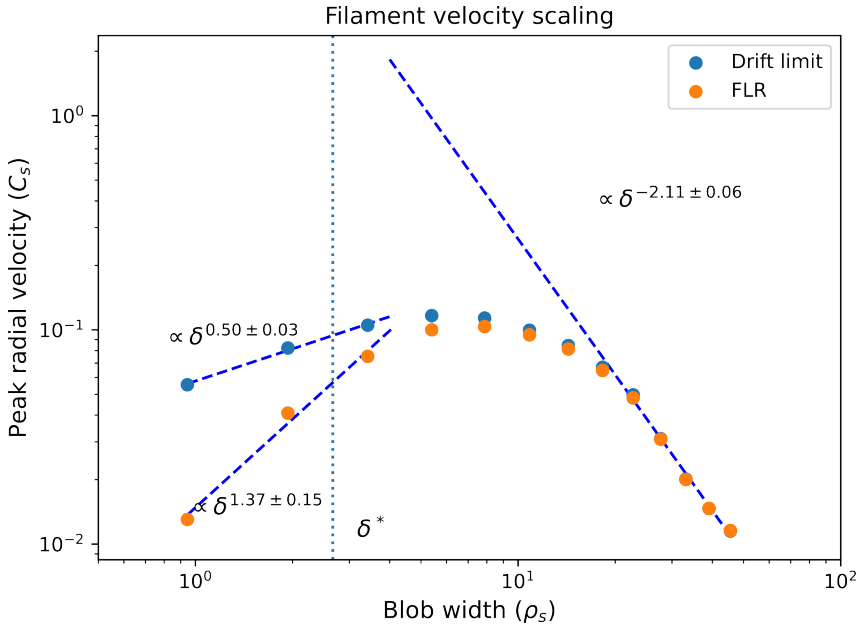

Figure 36: Maximum filament radial velocity as a function of blob width for fully connected filaments. In the sheath limited case, for large filaments, both the drift-limit simulations and FLR simulations reproduce the analytical velocity scaling law. In the inertial limit the drift-limit simulations reproduce the analytical scaling law but the FLR simulations show a clear deviation

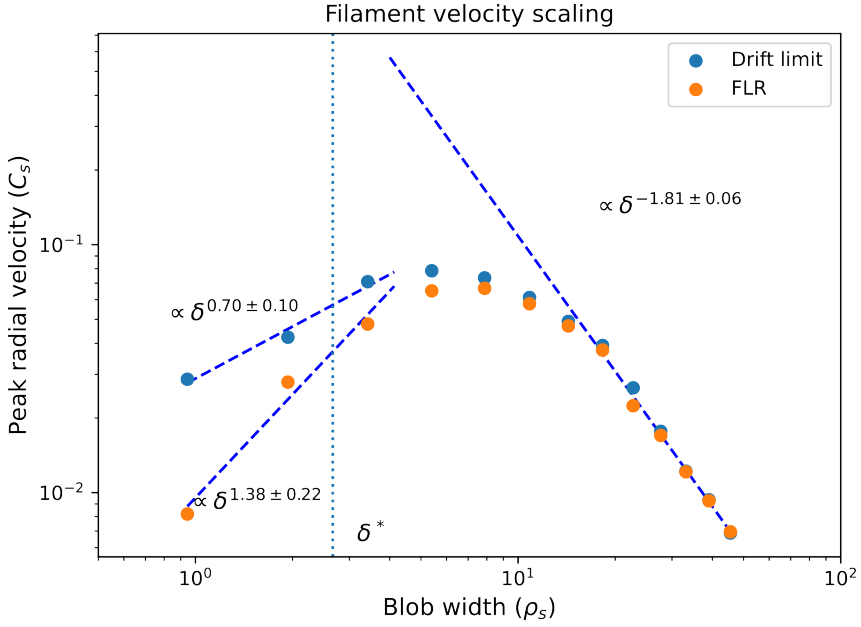

Figure 37: Maximum filament radial velocity as a function of blob width for half connected filaments. A similar deviation between FLR and drift-limit simulations is observed in the case of small filaments.

4.4 Hot-electron, Cold-ion simulations

On inclusion of electron thermal moments the main modification for filaments is inclusion of $T_{\parallel e}$ in the sheath boundary condition. This coupling allows Boltzmann spinning to occur as was the case in 2D simulations section 3.2 when the sheath closure was applied. As was the case for cold-electron, cold-ion simulations, a stable background was located with the use of the PID controllers described in section 4.1. This background is shown in fig. 38

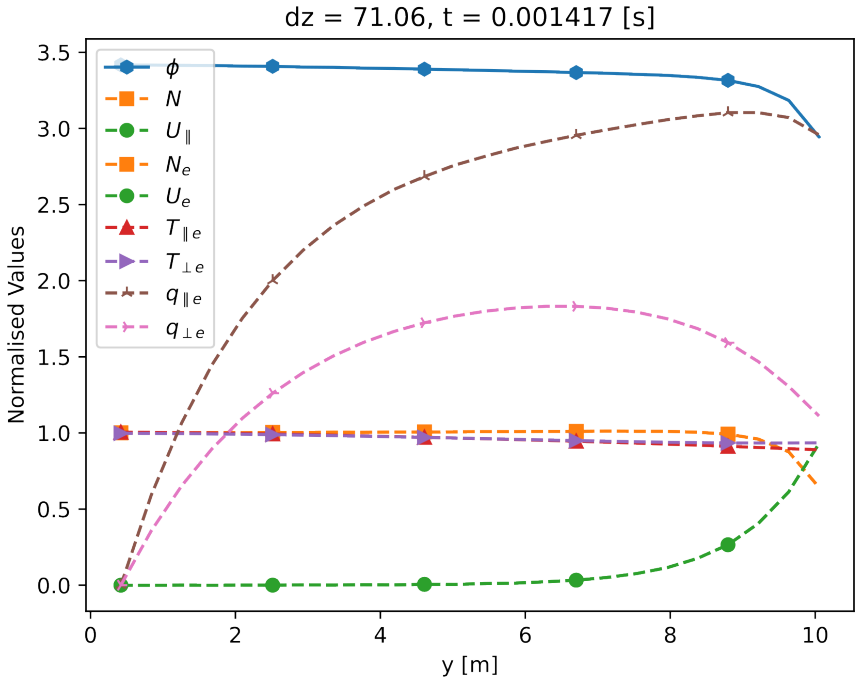

Figure 38: Stable background onto which filaments are superimposed for cold-ion, hot-electron simulations.

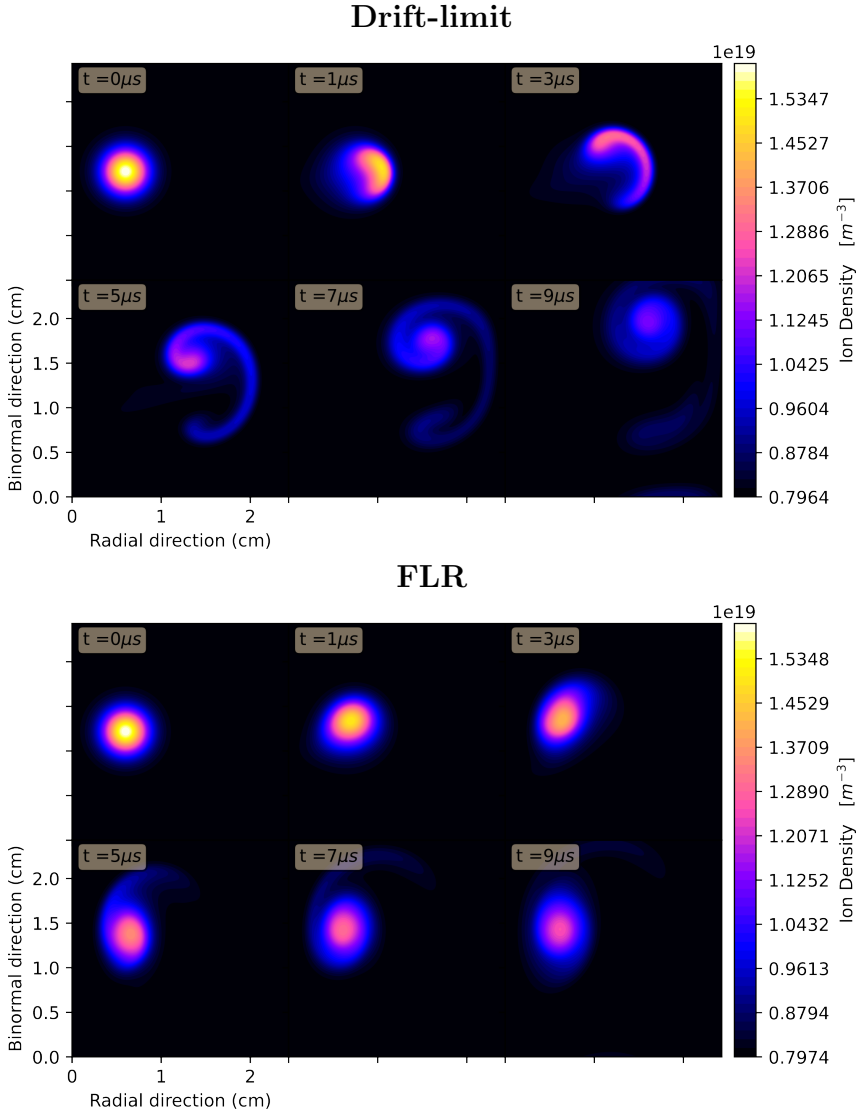

Figure 39: In the case of small filaments, spinning is enhanced when the electron temperature is evolved. This can be attributed to Boltzmann spinning. The monopole electron temperature in the filament affects the floating potential at the sheath which couples to the electron density though the parallel gradient term $\nabla_{\parallel}(\phi_G + \tau_z p_{z\parallel})$ in eq. (64)

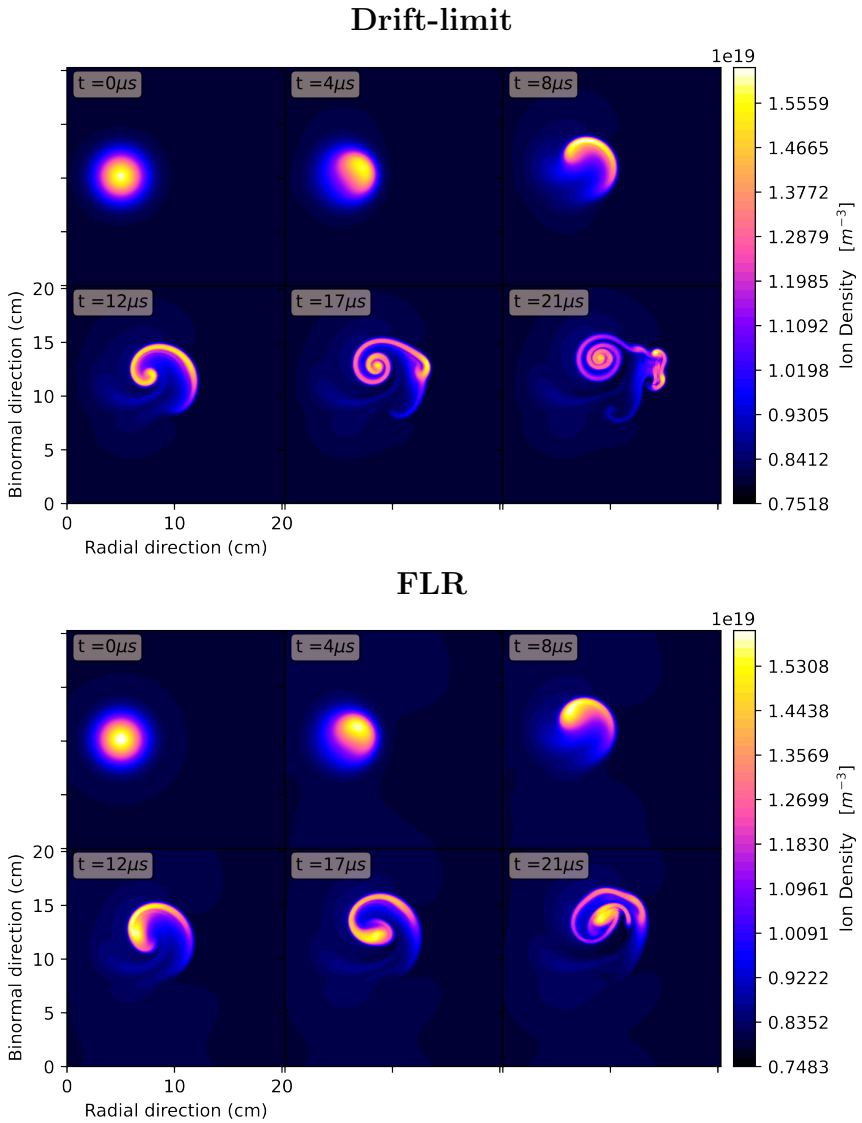

Figure 40: For intermediate sized filaments the typical largely coherent "mushrooming" motion is still observed in the initial stages of filament propagation. However, the finite electron temperature contributes a monopole component to the potential through modification of the floating potential at the sheath which leads to the filament spinning up (Boltzmann spinning) and thereby remaining more coherent than in the cold electron case

As with the cold electron, cold ion simulations above we are concerned mostly with filament velocity trends and less with the form of individual filaments. As such, we once again consider the peak radial velocity as a function of blob width and look for scaling relations. It is clear from fig. 41 that the deviation in the inertial limit is still present for hot electron simulations that include FLR compared to drift-reduced hot electron simulations.

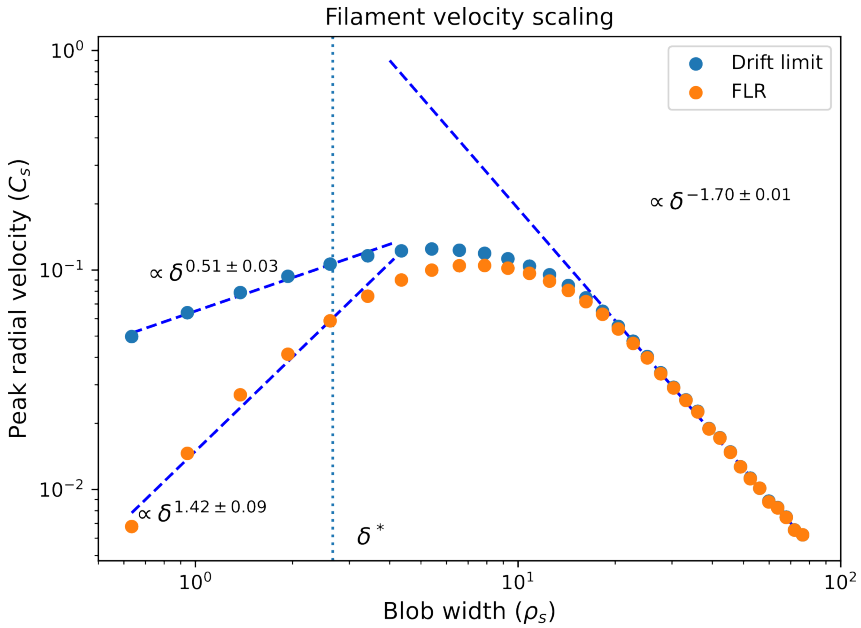

Figure 41: Maximum filament radial velocity as a function of blob width for fully connected filaments. Agreement for large filaments and a deviation for small filaments is still observed between the drift-limit and FLR case

The simulations were repeated for a series of half-connected filaments that do not reach all the way to the downstream target and corresponding sheath. The deviation in the case of FLR effects does not change substantially when going from connected filaments to half connected as shown in fig. 42.

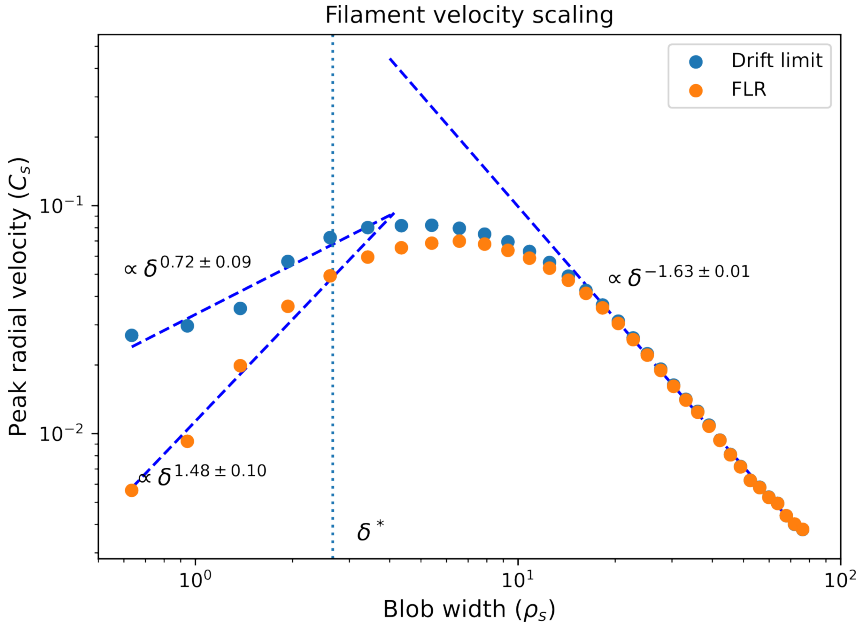

Figure 42: Maximum filament radial velocity as a function of blob width for half connected filaments. As with the connected case, agreement for large filaments and a deviation for small filaments is still observed between the drift-limit and FLR case

4.5 Hot-electron, Hot-ion simulations

Hot-ion, hot-electron simulations represent the maximal set of equations implemented here. Ion thermal moments allow the higher order FLR correction terms to be calculated. For example, in the density moment equations (eq. (63)) $[\Omega_G, T_{z\perp}]$ represents an FLR correction to the advection of n by the $\mathbf{E} \times \mathbf{B}$ velocity. As in previous simulations a stable background was located with the use of the PID controllers described in section 4.1. This background is shown in fig. 43

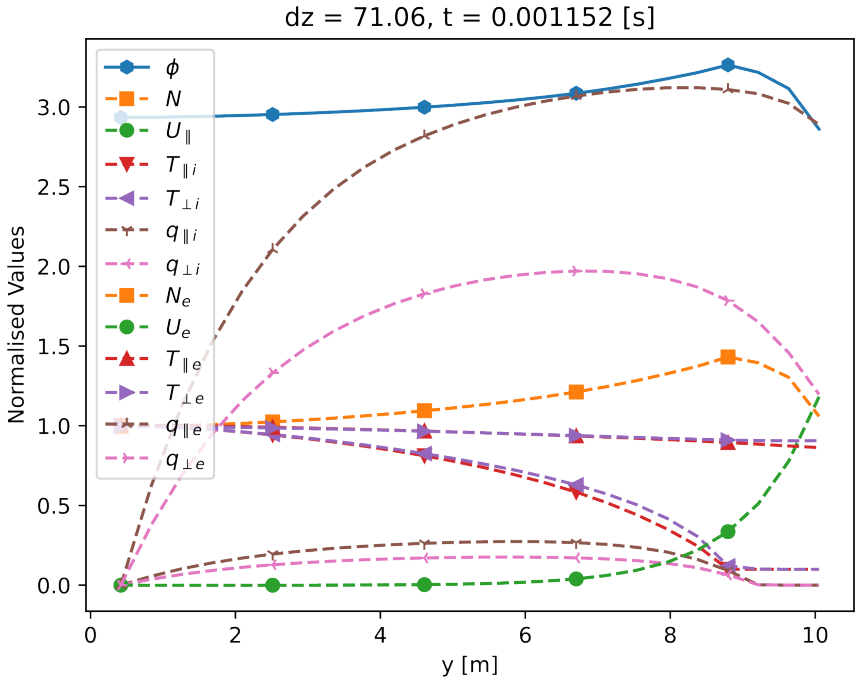

Figure 43: Stable background onto which filaments are superimposed for hot ion, hot electron simulations.

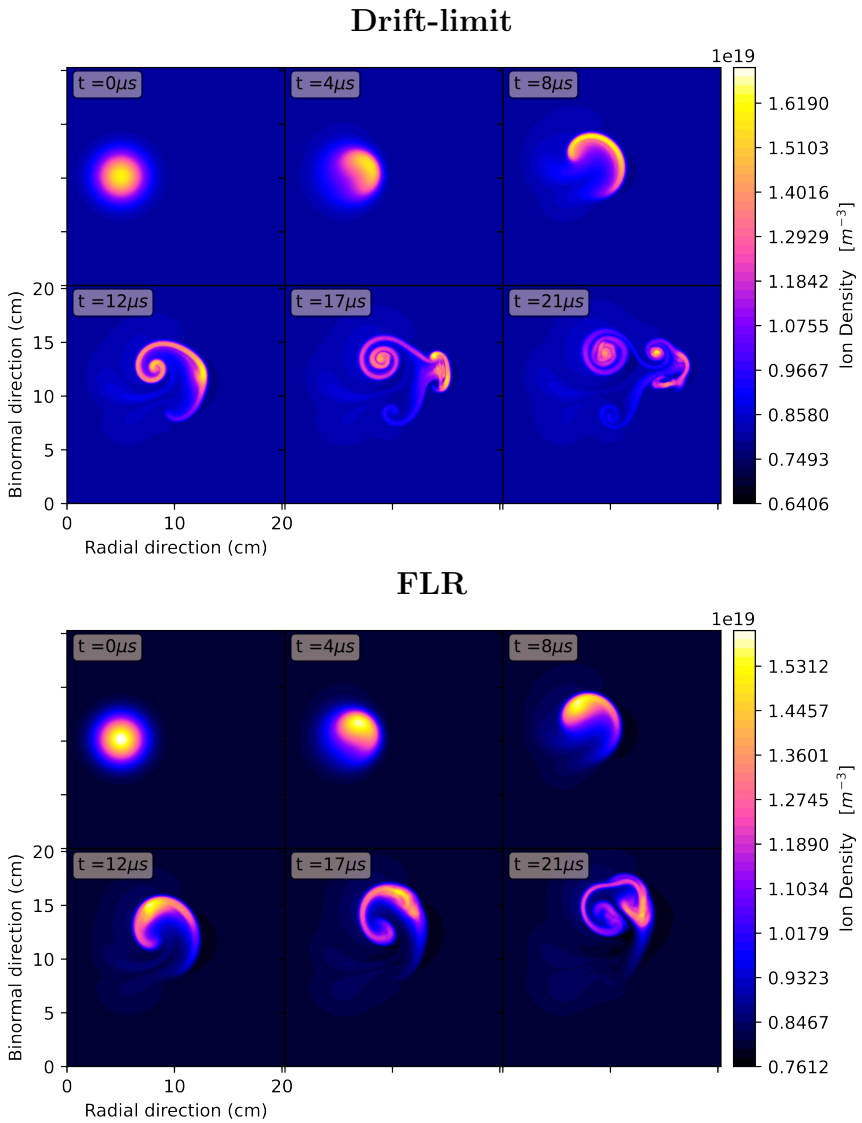

Figure 44: For intermediate sized filaments the typical largely coherent "mushrooming" motion is still observed in the initial stages of filament propagation. For all filaments a slight further spin-up and rotation is also observed on inclusion of finite ion temperature

As with the cold electron, cold ion simulations, and hot electron simulations above we are concerned mostly with filament velocity trends and less with the form of individual filaments. As such, we once again consider the peak radial velocity as a function of blob width and look for scaling relations. It is clear from fig. 45 that the deviation in the inertial limit is still present for hot electron simulations that include FLR compared to drift-reduced hot electron simulations.

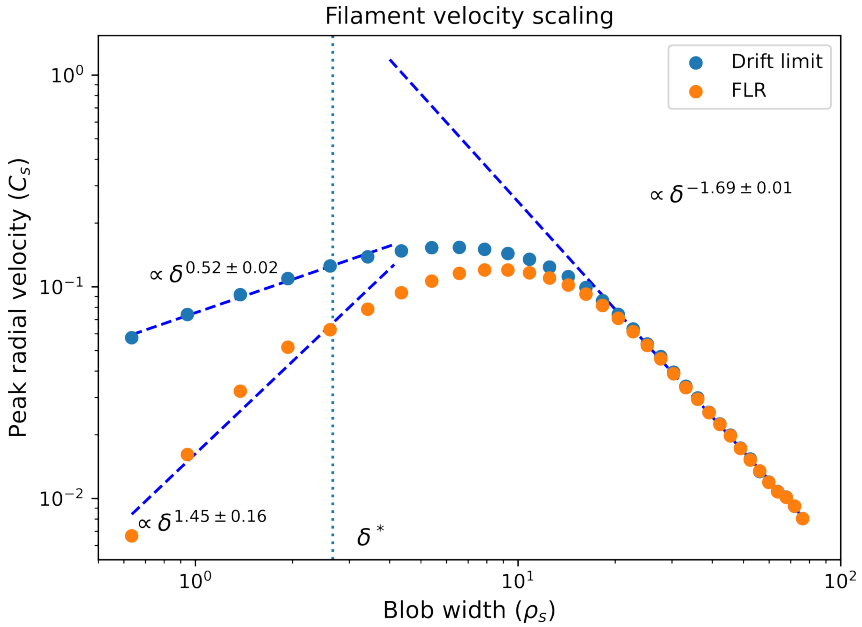

Figure 45: Maximum filament radial velocity as a function of blob width for fully connected filaments. As has been the case for all the filament simulations presented there is agreement between the drift-limit and FLR simulations for large filaments but a deviation for small filaments.

The simulations were repeated for a series of half-connected filaments that do not reach all the way to the downstream target and corresponding sheath. The deviation between the FLR and drift-limit simulations does not change substantially when going from connected filaments to half connected as shown in fig. 46.

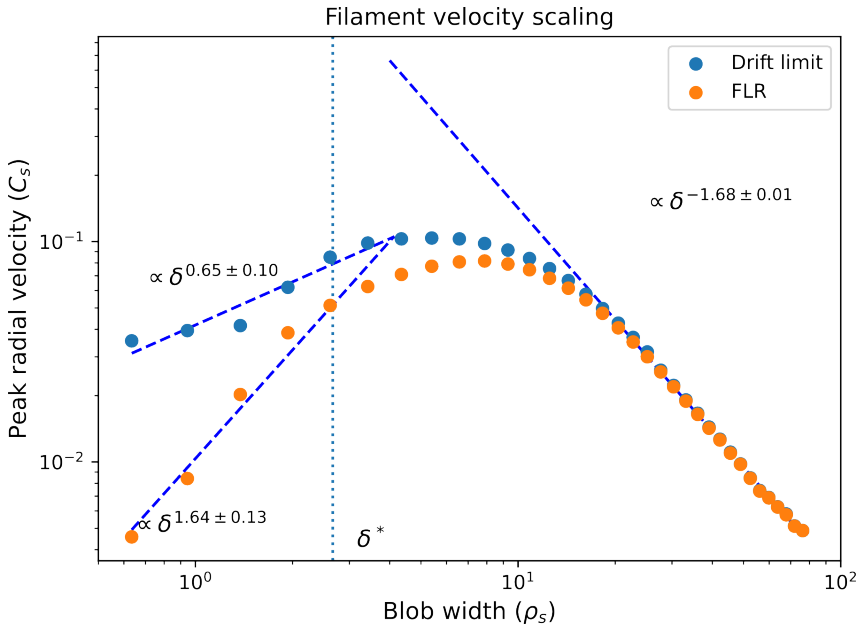

Figure 46: Maximum filament radial velocity as a function of blob width for half connected filaments. As with the fully-connected case there is agreement between the drift-limit and FLR simulations for large filaments but a deviation for small filaments

4.6 Results

Apart from altering the form of individual filaments which is only of cursory interest it is clear that a consistent deviation is predicted in the inertial limit (the limit of small filaments) upon inclusion of FLR effects. This deviation is readily apparent with any combination of moment equations included for ions and electrons. The deviation also robust to varying filament parallel length. The exponents from each of the power law fits above are shown in section 4.6. This result supports the earlier 2D simulation results and will be itself supported further though linear analysis in section 5. This deviation may be of interest to models that take inputs or set parameters based on filament statistics. These results support our hypothesis that FLR effects impact the dynamics of small filaments.

	Filament Length	Simulation Type	Inertial Limit	Sheath Limit
Cold-Electron, Cold-ion	$L_\parallel$	Drift Limit	0.50 ± 0.03	-2.11 ± 0.06
		FLR	1.37 ± 0.15	
	$L_\parallel/2$	Drift Limit	0.70 ± 0.10	-1.81 ± 0.06
		FLR	1.38 ± 0.22	
Hot-Electron, Cold-ion	$L_\parallel$	Drift Limit	0.51 ± 0.03	-1.70 ± 0.01
		FLR	1.42 ± 0.09	
	$L_\parallel/2$	Drift Limit	0.72 ± 0.09	-1.63 ± 0.01
		FLR	1.48 ± 0.10	
Hot-Electron, Hot-ion	$L_\parallel$	Drift Limit	0.52 ± 0.02	-1.69 ± 0.01
		FLR	1.45 ± 0.16	
	$L_\parallel/2$	Drift Limit	0.65 ± 0.10	-1.68 ± 0.01
		FLR	1.64 ± 0.13	

Table 4: Power law exponents for velocity scaling laws fit to simulation data for connected ($L_\parallel$) and half connected ($L_\parallel/2$) filaments for each set of 3D simulations

4.7 Effect of diffusivity

As shown in section 1.7.1, dissipative terms enter in the form, of collisional dissipation to the momentum, temperature and heat flux moments. The density and temperature equations also have diffusive terms of the form $-(\nu_\perp \nabla_\perp^2 + \nu_\parallel \nabla_\parallel^2)$.

In most fluid models these diffusive parameters (as well as viscous terms that act to diffuse vorticity) are derived based on physical arguments [84]. In gyrofluid models dissipative terms (if they are included) are typically based on physical arguments such as collisional processes or Landau damping. The diffusive terms however, are viewed as artificial and are included simply to remove energy that cascades to the highest spatial frequency supported by the grid and to damp oscillations that arise from the numerical discretisation [85, 72]. As such, they must be set as small as possible [72]. Also, where the viscosity in drift-fluid models is set from some physical argument it enters in gyrofluid models through the polarisation equation. Diffusive terms in the density and temperature equations enter the polarisation equation (through the gyroaverage operators) and appear as viscous terms as follows.

Starting with the polarisation equation eq. (69):

$$\sum_z a_z \left[\Gamma_1 n_z + \Gamma_2 T_{z\perp} + \frac{\Gamma_0 - 1}{\tau_z} \phi \right] = 0 \tag{153}$$

Now considering the two moment version of GEM and assuming a single ion species while also approximating the electron gyroaverage operators because the electron Larmor radius is much smaller that of ions.

$$\Gamma_0 \to 1 (\text{for electrons}) \tag{154}$$

$$\Gamma_1 \to 0 (\text{for electrons}) \tag{155}$$

$$\Gamma_1 n_i - n_e + (\Gamma_0 - 1)\, \phi = 0 \tag{156}$$

Now substituting in the Padé approximated gyro-average and gyro-screening operators for ions from eqs. (77) and (78):

$$\frac{n_i}{1-\frac{1}{2}\rho_i^2\nabla_\perp^2} - n_e + \left(\frac{1}{1-\rho_i^2\nabla_\perp^2} - 1\right)\phi = 0 \tag{157}$$

$$n_i - \left(1 - \frac{\rho_i^2}{2}\nabla_\perp^2\right) n_e + \rho_i^2\nabla_\perp^2\phi = 0 \tag{158}$$

Next the time derivative is taken and the $\nabla_\perp^2\phi$ is identified as the vorticity ω (which we show in section 5)

$$\frac{\partial n_i}{\partial t} - \left(1 - \frac{\rho_i^2}{2}\nabla_\perp^2\right)\frac{\partial n_e}{\partial t} + \rho_i^2\frac{\partial \omega}{\partial t} = 0 \tag{159}$$

It is clear from this expression that diffusive terms entering the ion and electron density equations also feed through to the polarisation equation where they act as viscous terms. Similarly, particle sources or sinks also act as vorticity sources and sinks respectively. As such, their addition must be treated with care.

It is clear from figs. 47 and 48 that the effect of a higher diffusivity is to smooth out features with a high spatial frequency. The diffusivity chosen here is an order of magnitude greater than that of the simulations discussed in previous sections. It is of interest because it more closely matches typical drift fluid simulations. The results with this diffusive term are qualitatively similar to those obtained in typical drift fluid simulations. They lack the enhanced vorticity striations that are typically seen with gyrofluid models. They also do not exhibit the spiral arms seen in the less diffusive simulations as well as in [86] and in the hybrid (kinetic-ion, fluid-electron) simulation in [87]. This is somewhat unsurprising because a large diffusivity simply acts to damp out high spatial frequency components. In the case of fig. 48 the blob itself is very quickly dissipated. It is important to remember however, that the numerical diffusion from the gyrofluid treatment is distinct from the physically motivated fluid treatment. In the gyrofluid case the numerical diffusion coefficients should in principle be minimised and for each set of simulations were selected such that no grid scale numerical instabilities formed.

However, completely damping out high spatial frequency dynamics does not seem like a desirable effect. In fact in some cases hyperdiffusive terms of

the form $\nu_\perp \nabla_\perp^4$ are used in place of the diffusive terms due to the narrower spectral range over which they operate [71], these hyperdiffusive terms would naturally act also as hyperviscous terms through the polarisation equation as explained above.

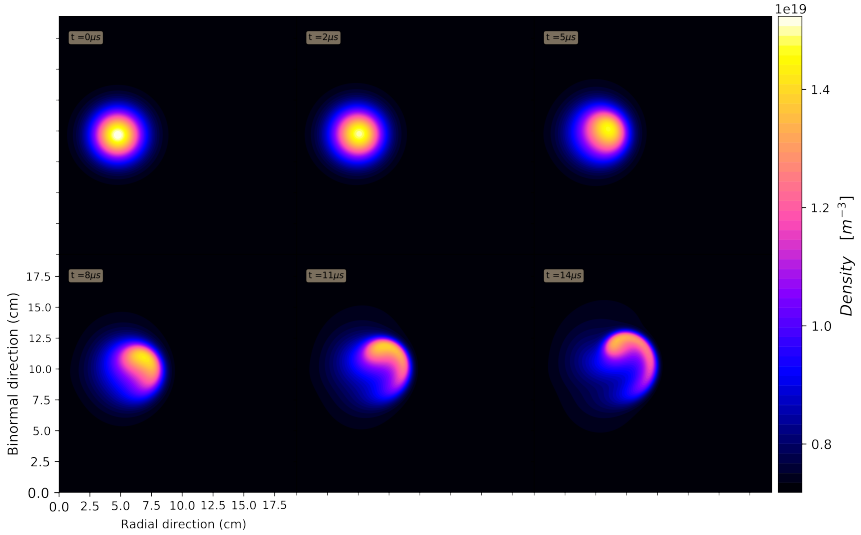

Figure 47: Medium-sized blob evolution with FLR effects included with a diffusion coefficient of 10^{-2}. The blob dissipates more quickly with increasing artificial diffusion.

It is not entirely unexpected then that the scaling should deviate for small blobs, as it does in fig. 49. In the case of large artificial diffusion coefficients are quickly, and artificially, diffused away before reaching their peak velocity as can be seen in fig. 48. For intermediate values of the diffusion coefficient the velocity scaling remains unchanged compared to the results presented above.

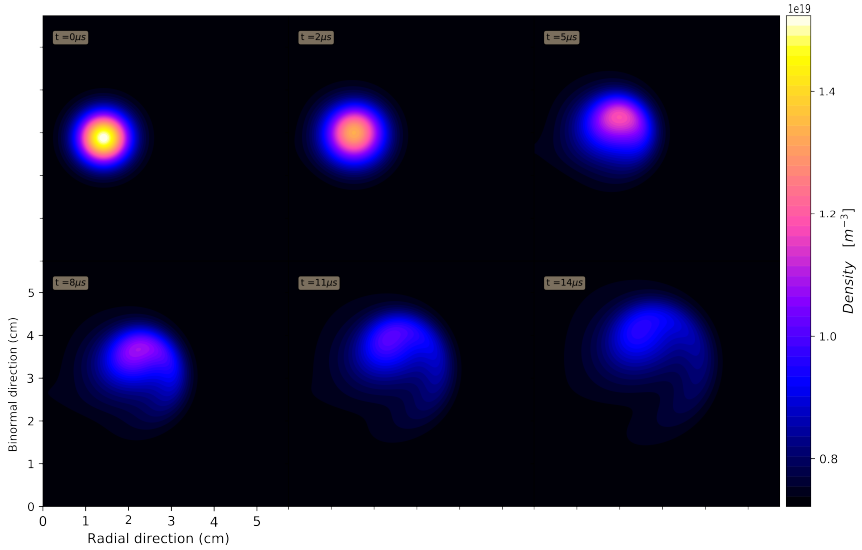

Figure 48: Cold Ion, Cold Electron Small blob evolution with FLR effects included with a diffusion coefficient of 10^{-2} The artificial diffusion quickly dissipates the blob

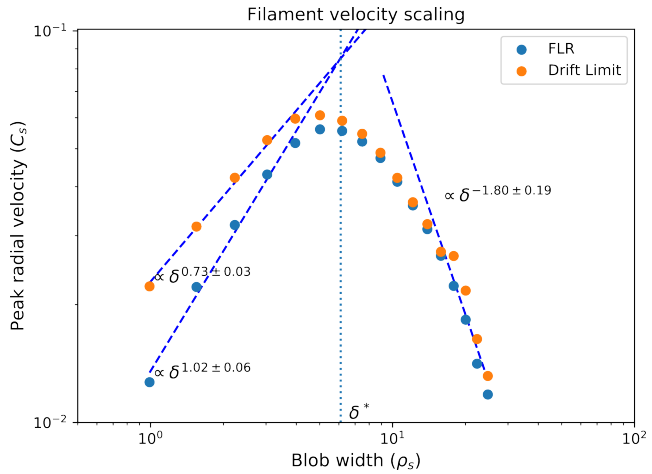

Figure 49: Cold Ion, Cold Electron filament centre of mass peak velocity with a diffusion coefficient of 10^{-2} The high artificial diffusion quickly dissipates small blobs and the resultant artificial viscosity affects the measured velocities

4.8 Grid Resolution Study

Similar to section 3.3 we systematically refined our 3D grids for both the drift-fluid limit and with FLR effects included to ensure our results did not depend on the grid resolution. The parameters used for this study were the same as shown in table 3 except for the grid resolution which was varied.

An example of the solutions obtained under successive refinement by a factor of 2 for the grid spacing is shown in fig. 50

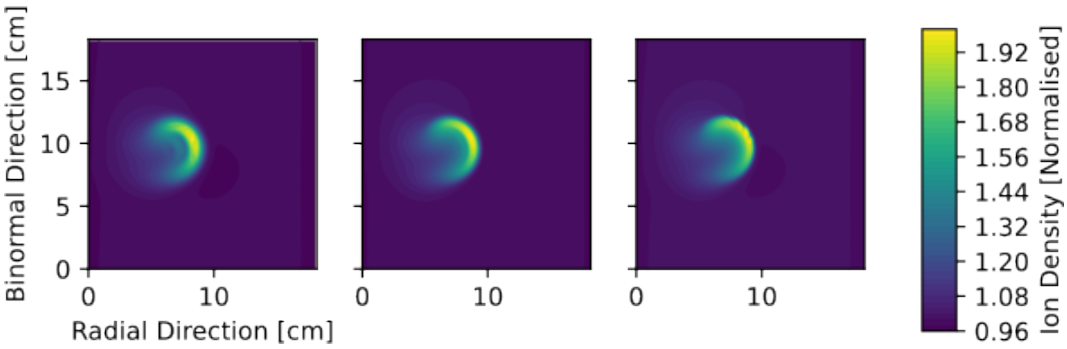

Figure 50: Three 3D simulations of the same filament taken in the drift-limit plotted at the midplane. From left to right we have a coarse mesh $N_x \times N_z \times N_y$ with $64 \times 64 \times 10$, a medium mesh with $128 \times 128 \times 20$ points and a fine mesh with $256 \times 256 \times 40$ points

The same argument regarding convergence from section 3.3 applies also here for our 3D simulations. For the same reasons as outlined there we successively refine our meshes and measure the filament velocity and the error norm for our density field.

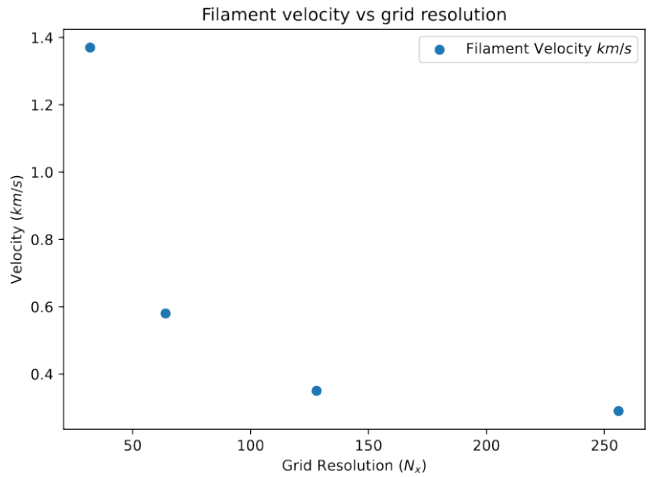

Figure 51: Filament peak radial velocity plotted against grid resolution. As our 3D mesh is successively refined our peak velocity converges, similarly to how it did in our 2D grid resolution study

We now consider our solutions for n_i at the time of peak radial velocity. The same method as in the 2D case is used to estimate our error and calculate the error norm. When we plot the error norm against grid spacing (fig. 51) we see convergence below the rate expected. However, this is arising from the discontinuous downstream boundary condition outlined in section 4.1. To test this we calculate the error norm also for just the midplane (fig. 52) and see that our error converges at the expected rate.

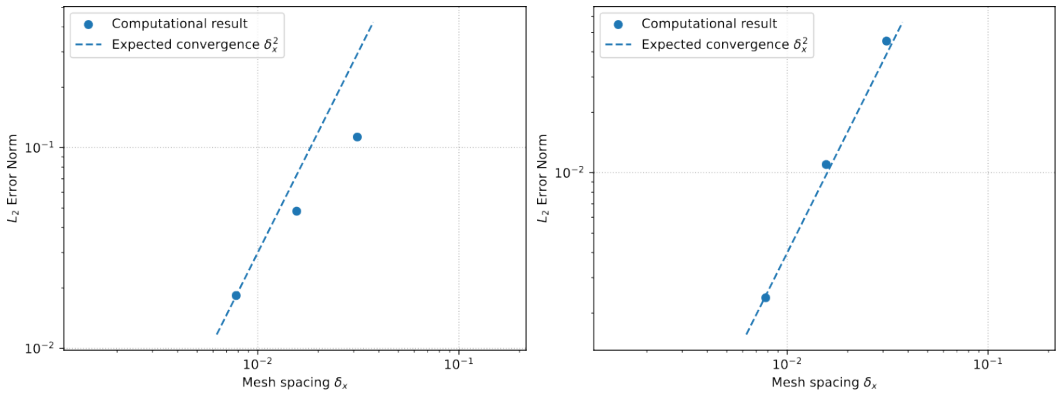

Figure 52: L_2 error norm for the whole domain (left) and for the midplane (right) plotted against mesh spacing δ_x alongside the expected convergence. It is clear that the solution for the whole domain is converging more slowly than expected. This is attributed to the boundary condition discontinuity discussed in section 4.1 and the error is likely dominated by this discontinuity. At the midplane, far from our downstream boundary, our solutions are converging at the expected rate for our differencing scheme

5 Analytic scaling relations

5.1 Motivation

We have shown in previous sections that when simulating filaments with a gyrofluid model there is a deviation for small filaments from the well-known velocity scaling laws and from drift fluid simulations. We have reasoned about this by considering the potential that generates the filament $\mathbf{E} \times \mathbf{B}$ velocity and how it is affected by gyroaveraging. In this section we intend to justify this computational result more rigorously.

Filament dynamics are generally nonlinear, however analytic scaling relations appear to adequately describe the maximum radial centre of mass velocity as a function of (among other parameters) the filament cross-sectional width [50, 28, 27]. The two regimes of interest here are the inertial (resistive-ballooning) regime and the sheath-limited (sheath interchange) regime. The inertial limit is found when $\delta_\perp \approx \rho_i$, the blob velocity in this limit is found to scale with the square root of the blob width $v_r \propto \delta_\perp^{0.5}$. In the limit of large blob width $\delta \gg \rho_i$ the sheath-limited regime is relevant and the blob velocity scales inversely with the square of the blob $v_r \propto \delta_\perp^{-2}$. These relations have been obtained from dispersion relations by Myra [50] where a correspondence principle is applied and also through linearisation of fluid equations by Walkden [48]. These analytic relations have also been reproduced using nonlinear fluid simulations [76].

Here we will apply Walkden's method to the GEM gyrofluid equations that we have used for the simulations in the previous sections. We will find that the analytic relations can be recovered from the GEM equations only when FLR effects are neglected.

5.2 Method & Result

The analytic relations can be recovered from the GEM moment equations as follows, starting from our polarisation equation eq. (69) and neglecting thermal

dynamics:

$$\sum_z a_z \left[\Gamma_1 n_z + \frac{\Gamma_0 - 1}{\tau_z} \phi \right] = 0$$
$$\Gamma_0 = \left(1 - \rho_z^2 \nabla_\perp^2\right)^{-1}$$
$$\left(\Gamma_0 - 1\right) \phi = \frac{1 - (1 - \rho_z^2 \nabla_\perp^2)}{(1 - \rho_z^2 \nabla_\perp^2)} \phi = \Gamma_0 \rho_z^2 \nabla_\perp^2 \phi$$

Then in the drift limit $\Gamma_0 \to 1$, $\Gamma_1 \to 1$

$$\sum_z a_z \left[n_z + \rho_z^2 \nabla_\perp^2 \tau_z \phi \right] = 0$$

Now assuming one singly-charged ion species:

$$a_i n_i + a_e n_e + \rho_i^2 \nabla_\perp^2 \tau_i \phi + \rho_e^2 \nabla_\perp^2 \tau_e \phi = 0 \tag{160}$$

And letting $\rho_e \to 0$ since $\rho_e \ll \rho_i$

$$\rho_i^2 \nabla_\perp^2 \phi = n_e - n_i \tag{161}$$

Then we identify $\nabla_\perp^2 \phi$ as the generalised vorticity Ω (not to be confused with the gyroscreened electrostatic potential Ω_G) as follows, starting from the definition of vorticity where $\mathbf{v}$ is the flow velocity.

$$\Omega \equiv \nabla \times \mathbf{v} \tag{162}$$

In our case the flow velocity, which we assume to be incompressible, is the $\mathbf{E} \times \mathbf{B}$ velocity

$$\Omega = \nabla \times \frac{\mathbf{E} \times \mathbf{B}}{B^2} \tag{163}$$

In our slab geometry $\mathbf{B} = B\hat{\mathbf{y}}$ and in the electrostatic case $\mathbf{E} = -\nabla\phi$.

$$\Omega = -\nabla \times \frac{\nabla\phi \times B\hat{\mathbf{y}}}{B^2} \tag{164}$$

After working through the vector calculus one arrives at the following expression:

$$\Omega = \frac{1}{B}\nabla_{\perp}^{2}\phi \tag{165}$$

Now that we have justified our identification of $\nabla_{\perp}^{2}\phi$ as the generalised vorticity we continue from eq. (161) Taking the time-derivative in a system of units where $\rho_i \to 1$ we arrive at an expression for the vorticity as a function of moment equations.

$$\begin{aligned}\frac{\partial \Omega}{\partial t} &= \frac{\partial n_e}{\partial t} - \frac{\partial n_i}{\partial t} \\ &= -\left[\phi, \Omega\right] + \nabla_{\parallel} J_{\parallel} - \mathcal{K}\left(\frac{p_{e\parallel} + p_{e\perp} + p_{i\parallel} + p_{i\perp}}{2}\right)\end{aligned} \tag{166}$$

An integral has been taken along the parallel direction while assuming filaments are fully connected to the sheath and therefore that density and temperature are homogenous in the parallel direction. Linearised Bohm sheath boundary conditions [69] are used and pressures are linearised under gradient operators.

$$\begin{aligned}\frac{\partial \Omega}{\partial t} =& -\left[\phi, \Omega\right] + \frac{1}{L_{\parallel}}\left(\phi - T_e V_F\right) \\ &+ g\frac{\partial}{\partial z}\left(n_e + n_i + \frac{T_{e\parallel} + T_{e\perp} + T_{i\parallel} + T_{i\perp}}{2}\right)\end{aligned} \tag{167}$$

Equation 167 is then split into even and odd spatial components using the method introduced by Walkden et al. [88] which has previously been applied to drift-ordered fluid equations [48]. For simplicity an isothermal filament where $T_i = T_e$ and $T_{\parallel} = T_{\perp}$ will be considered.

$$\frac{\partial \Omega^e}{\partial t} + v_E^o \cdot \nabla_{\perp}\Omega^o + v_E^e \cdot \nabla_{\perp}\Omega^e = \frac{1}{L_{\parallel}}\left(\phi^e - T_e V_F\right) \tag{168}$$

$$\frac{\partial \Omega^o}{\partial t} + v_E^e \cdot \nabla_{\perp}\Omega^o + v_E^o \cdot \nabla_{\perp}\Omega^e = 2g\frac{\partial}{\partial z}p_e + \frac{1}{L_{\parallel}}\left(\phi^o\right) \tag{169}$$

The spatial gradient operators can be approximated as:

$$\nabla_{\perp} \approx \frac{\partial}{\partial z} \approx \frac{\partial}{\partial x} \approx \frac{1}{\delta_{\perp}} \tag{170}$$

Where $\nabla_\perp$ is the characteristic filament width. The time derivatives of the even and odd components of vorticity in eq. (169) and eq. (168) are set to zero independently to consider the period of filament propagation where vorticity is maximised, and therefore a filament is at its peak radial velocity. The drive and dissipative terms in eq. (169) and eq. (168) are equated, noting that the filament electron pressure is assumed even since the filament under consideration is assumed Gaussian. We also introduce the radial $\mathbf{E}\times\mathbf{B}$ velocity $v_r = |\mathbf{b} \times \nabla_\perp \phi^o| \approx \phi^o/\delta_\perp$. This highlights that only the odd component of the potential contributes to the radial velocity in this approximation. The even component of the velocity just leads to a rotation or spinning of the filament.

$$v_r \cdot \nabla_\perp \Omega^o = 2g \frac{\partial^2}{\partial z \partial p_e} \tag{171}$$

$$\frac{v_r^2}{\delta_\perp^2} = \frac{2g \cdot p_e}{\delta_\perp} \tag{172}$$

$$v_r \approx \sqrt{2\delta_\perp g \cdot p_e} \tag{173}$$

This, of course, is the inertial limit where the curvature drive is balanced by $\mathbf{E} \times \mathbf{B}$ advection. The characteristic velocity $v_r \propto \delta_\perp^{0.5}$ is recovered. Similarly, the sheath term can be balanced with the curvature drive term to capture the sheath limited regime.

$$\frac{1}{L_\parallel}(\phi^o) = -2g \frac{\partial}{\partial z} p_e \tag{174}$$

$$\frac{\delta_\perp v_r}{L_\parallel} = \frac{-2g \cdot p_e}{\delta_\perp} \tag{175}$$

$$v_r \approx \left| \frac{2g \cdot p_e \cdot L_\parallel}{\delta_\perp^2} \right| \tag{176}$$

Where again the characteristic velocity $v_r \propto \delta_\perp^{-2}$ associated with the sheath-limited regime has been recovered.

So here we have now recovered the velocity scaling laws with the GEM equations purely through an approximation of the parallel dynamics and decomposition of the potential into even and odd spatial components. Importantly this derivation only holds in the drift limit. If our gyroaverage operators aren't taken in the drift-fluid limit then we can't present a simple linearised equation from which the scaling relations are obtained. One should therefore

expect that when the filament size is on the order of the ion Larmor radius that the radial velocity may diverge from the relations derived here. By this same token one would expect these scaling relations to hold when the filament width is much larger than the ion Larmor radius as is the case in the sheath limited regime. It is also interesting that when Myra's correspondence method is used, the resistive ballooning mode that corresponds to the inertial limit is stabilised through the inclusion of FLR effects [89]. This is another justification for our small filament results. This analytical result serves to strengthen our argument arising from the numerical results for isolated filaments from sections 3 and 4.

6 Core-SOL Simulations

Now that isolated and interacting pairs filaments have been considered, the final obvious simulation that remains is a self-consistent core-SOL simulation. Such a simulation could, in theory, produce self-consistent turbulence driven radial profiles and heat flux predictions. Here we will start with 2D core-SOL simulations due to the reduced numerical complexity. We will, however, encounter an issue arising from the handling of parallel dynamics in the core in the 2D case. This issue motivates the transition to 3D simulations, which are shown in section 6.2

6.1 2D Core-Sol Simulations

Table 5: Parameters chosen for 2D core-SOL simulations

Input Parameters	Normalisation Parameters	Derived Parameters
$n_x = 128$	$\rho_s = 2.6 \times 10^{-3}\,\mathrm{m}$	$\nu_e = 6.9 \times 10^{-2}$
$n_z = 256$	$c_s = 4.3 \times 10^{4}\,\mathrm{m\,s^{-1}}$	$\nu_i = 1.2 \times 10^{-3}$
$\nu_\perp = 2 \times 10^{-2}$	$\Omega_i = 1.7 \times 10^{7}\,\mathrm{s^{-1}}$	
$\nu_\perp = 2 \times 10^{-2}$		
$T_e = 40\,\mathrm{eV}$		
$T_i = 40\,\mathrm{eV}$		
$N_i = 8 \times 10^{18}\,\mathrm{m^{-3}}$		
$N_e = 8 \times 10^{18}\,\mathrm{m^{-3}}$		
$R = 1.5\,\mathrm{m}$		
$B = 0.5\,\mathrm{T}$		
$l_\parallel = 10\,\mathrm{m}$		

Starting first with the 2D case we, as usual, need to apply a closure for the parallel dynamics as described in section 3. These simulations were carried out by applying the sheath-dissipation closure only outside the last closed flux surface. To handle the core we assume that parallel transport is so large that plasma parameters are homogenous in the parallel direction and so no net parallel loss occurs in the core. Such an approach was also taken in [90].

Density and energy sources were added in the core region which fall off exponentially from the inner x boundary. These sources lead to a pressure gradient in the core. In the case of these 2D simulations this pressure gradient is stable. In the 3D core-SOL simulations presented below, a Kelvin-

Helmholtz instability destabilises the pressure gradient and self-consistent turbulence forms. The suspected reason for this unphysical stability in 2D, is that our potential in the core isn't as tightly coupled as it needs to be to the electron density. With the assumptions about parallel losses above, which seem reasonable at first, we have effectively assumed that no current is present in the core. A parallel current is needed to dissipate the potential formed in the core and through such dissipation perturb the electron density. Our vorticity-advection closure is also not suitable for the core as it assumes currents close through the drift plane. Therefore, in order to properly capture parallel currents in the core, and dissipate the potential there in a physically motivated manner, another closure would have to be developed that is suitable for the core. In future work such a closure could be implemented.

In the absence of an initially growing mode to destabilise the core and generate turbulence a very small initial perturbation is added the density and temperature fields. This perturbation then quickly grows and allows turbulence to develop. The parameters chosen for these core-SOL simulations are given in table 5. A larger numerical diffusion than usual had to be selected for these simulations, as without it the solver failed to converge. Once the artificial potential in the core is understood it is clear that a higher artificial diffusion was required because it acts to damp this spurious potential in the absence of any other mechanism in the core to dissipate it.

Snapshots of the simulation are shown in figs. 53 to 55. A pressure gradient is seen to briefly form before the initial perturbation we seeded grows and destabilises the gradient. The initial blowout is shown in fig. 53 and its character is strongly dependent on the initial perturbation and therefore unphysical. It is suggested however, as in other similar turbulence studies [91], that the later behaviour of the system is not strongly determined by this initial condition. The saturated turbulent state is assumed to be largely independent of the artificial initial condition. Here we define a saturated state by a stationary power spectrum where energy is entering at some scale and cascades to dissipation scales.

As ever with these simulations it is useful to attempt to reduce the data. One such method is to take averages over some axes. The poloidal average can be considered when radial transport is of interest. Additionally, time averages

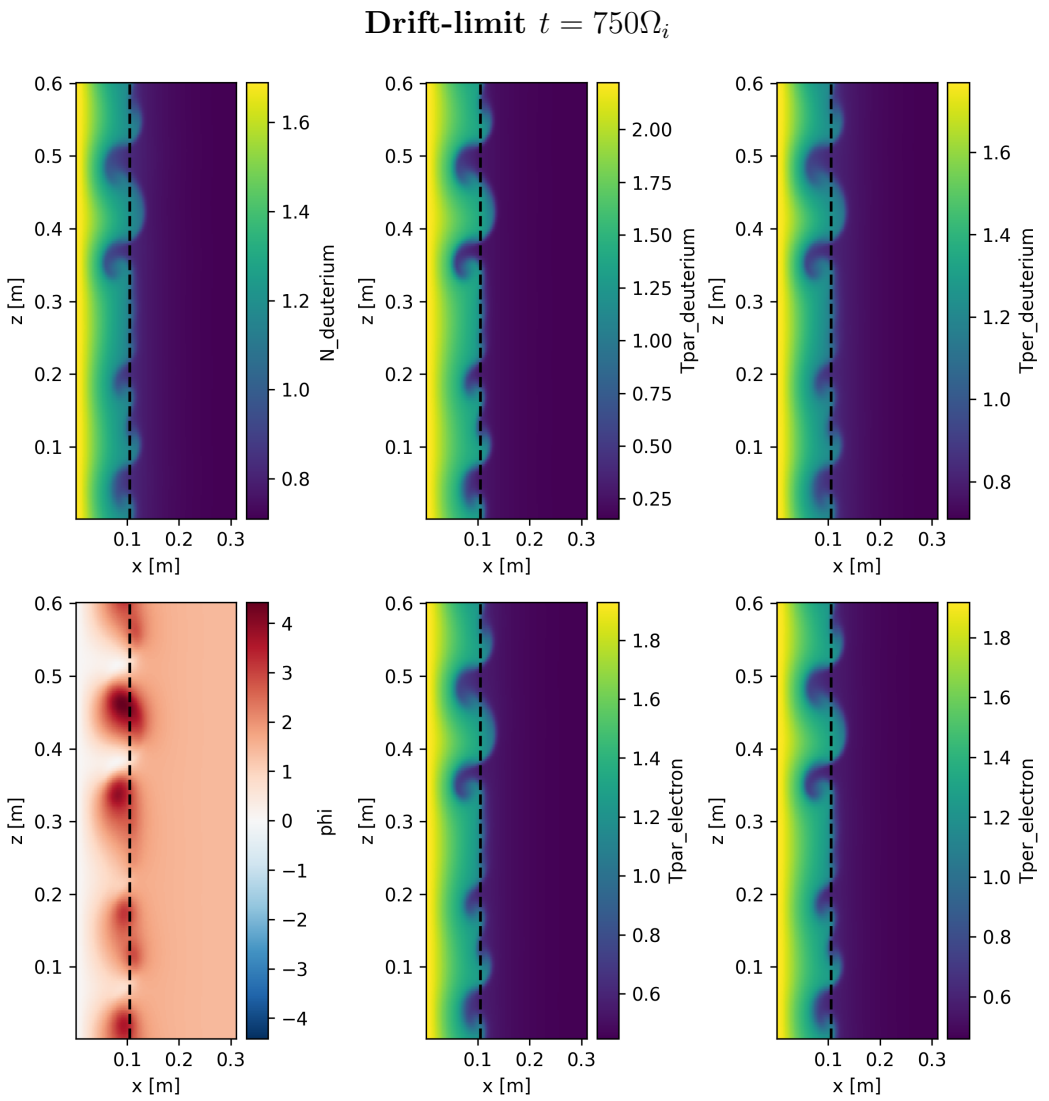

Figure 53: 2D core-SOL simulations with the sheath-dissipation closure applied outside the last closed flux surface (Drift-limit, $t = 750\Omega_i$) Here we see the seeded perturbation growing, this mode quickly cascades to smaller spatial scales and the disturbance it causes in the core persists and continues to eject plasma into the SOL. Roughly the same situation is seen in the FLR simulation up to this point.

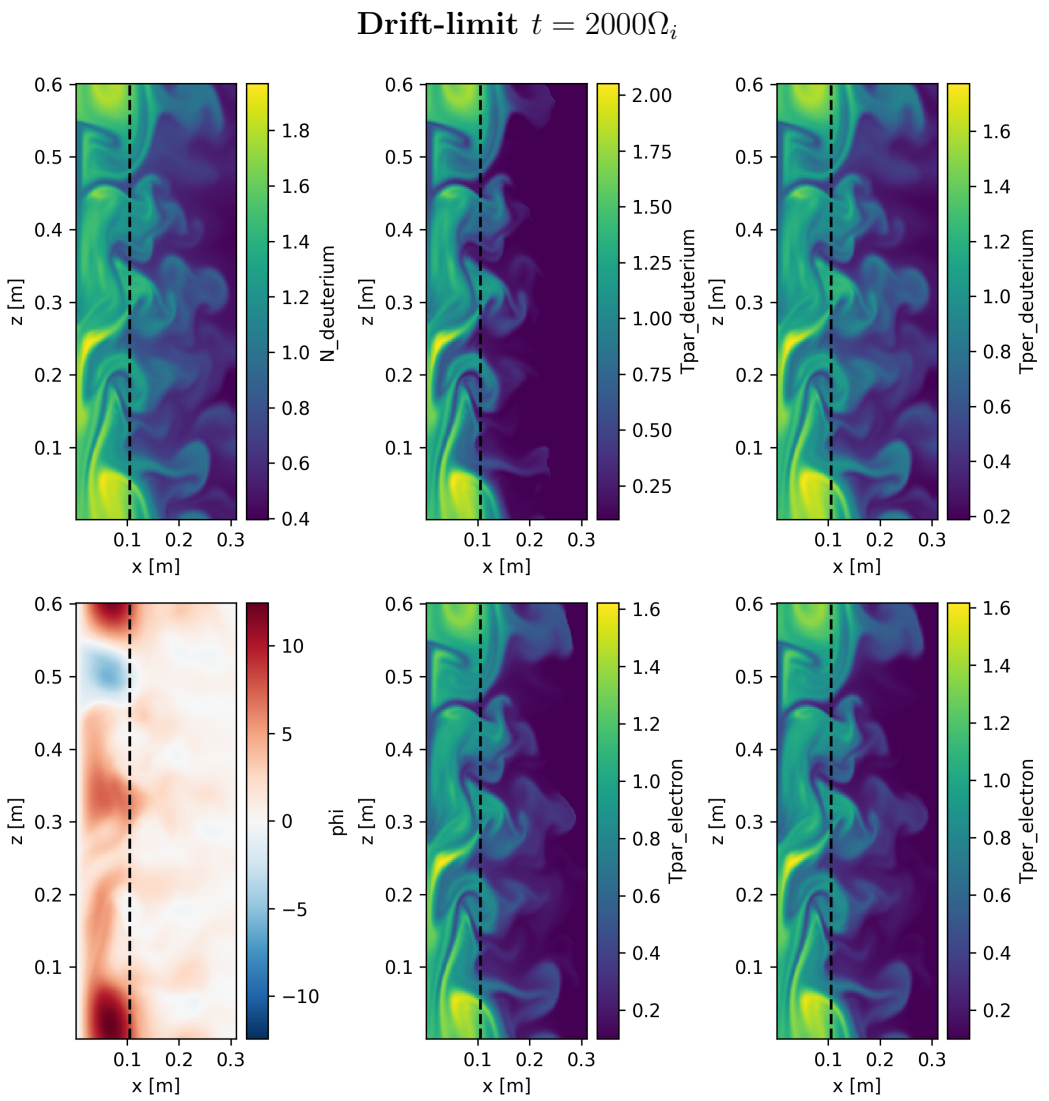

Figure 54: 2D core-SOL simulations with the sheath-dissipation closure applied outside the last closed flux surface (Drift-limit, $t = 2000\Omega_i$) At this point turbulence in the core is continuously generating filaments and ejecting them from the core

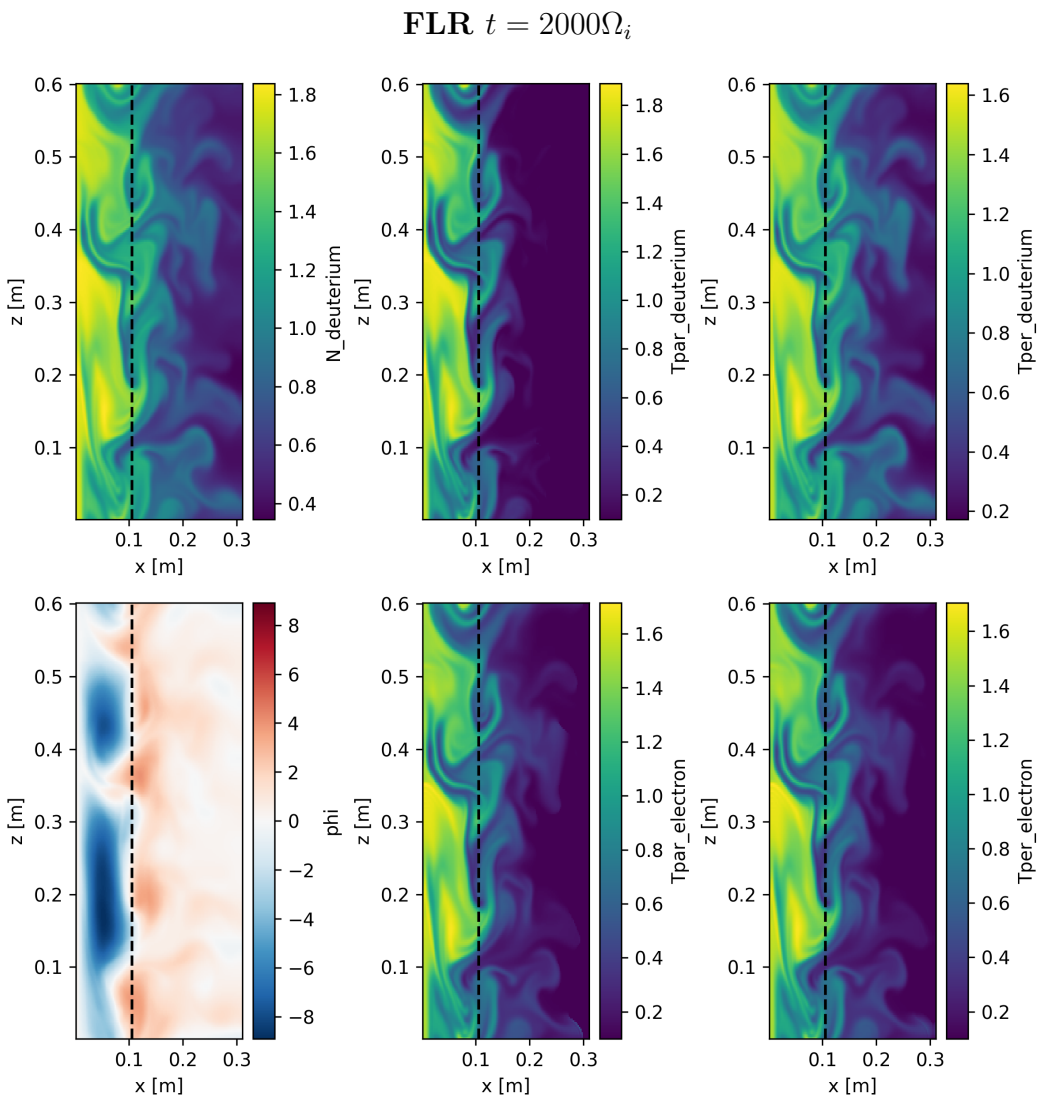

Figure 55: 2D core-SOL simulations with the sheath-dissipation closure applied outside the last closed flux surface (FLR, $t = 2000\Omega_i$) Similar to the drift-limited simulation turbulence in the core is continuously generating filaments which are ejected into the SOL

can be taken to consider the average behaviour of the system rather than considering transients. Both of these approaches are shown in figs. 56 and 57

One feature of interest in these simulations is the form the filaments take when ejected from the core. The similarity of these self-consistent filaments to the artificially initialised filaments we have considered thus far provides, to some extent, an a-posteriori justification for the previous single-filament and interacting filament simulations. Also noteworthy is the apparent reduction of transport at later times due to the formation of a shear flow. The formation of this shear flow can be seen clearly from the poloidally averaged plots of the average potential in fig. 56. The potential seen in the core gives rise to a radial electric field which then generates a poloidal (z direction in our slab geometry) $\mathbf{E} \times \mathbf{B}$ velocity. This shear flow may affect transport from the core, however, as we discussed above, it is unphysical. A similar flow generation, for the same reason as we outlined above, was identified in [92] where two possible solutions were presented. Firstly, one could increase artificial diffusion in the core above a threshold value which renders the core stable. This is the approach that was unknowingly taken here by increasing our artificial diffusion in response to simulations which failed to converge with less dissipation. An alternative solution proposed, and implemented, in [92] was to introduce a Hasegawa-Wakatani like term to reintroduce parallel currents to the core. A similar term could in principle be derived for the 2D version GEM. Nevertheless, we present here the results from our 2D core-SOL simulations, the results in the scrape off layer may still be interesting even with the unphysical potential that forms in the core.

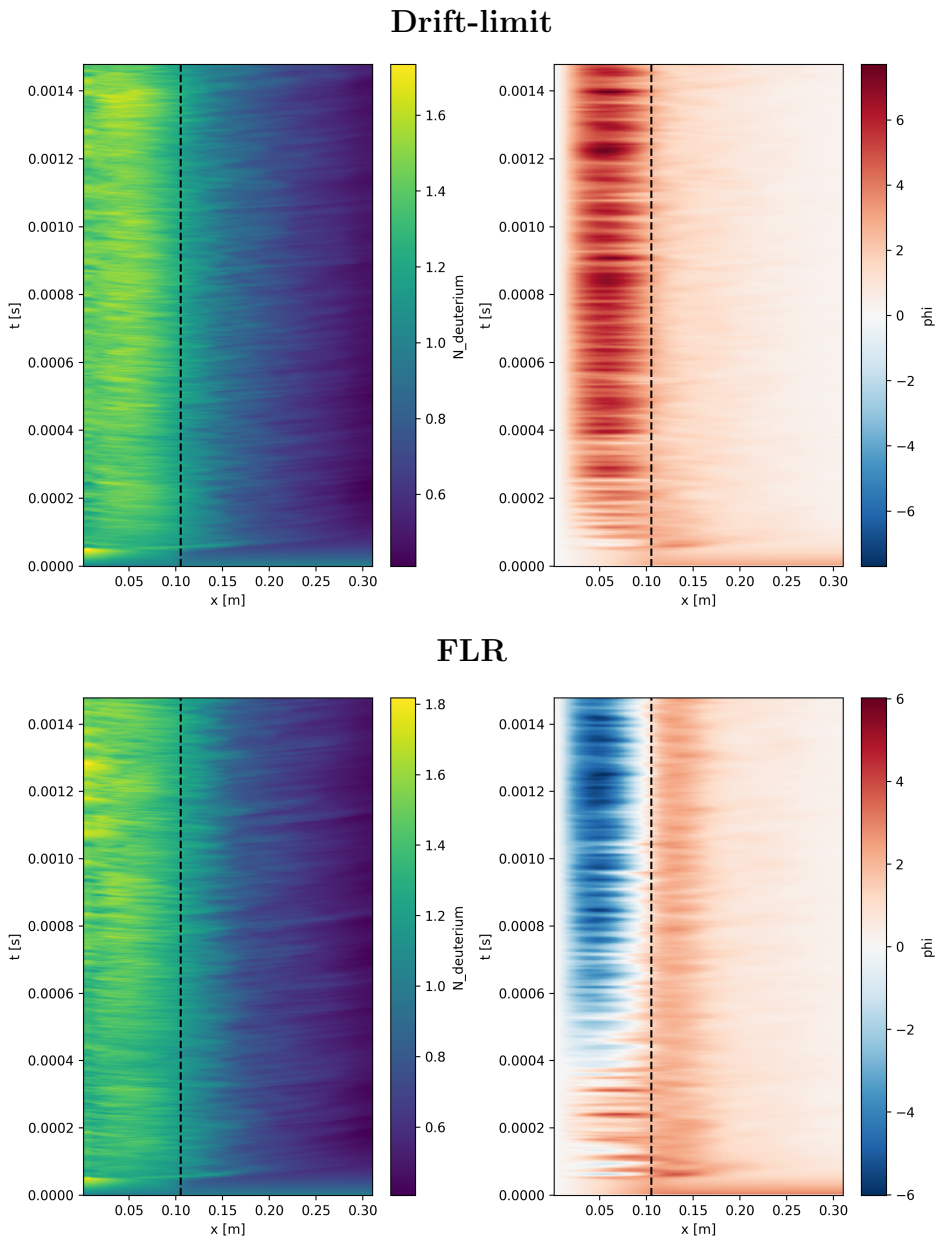

Figure 56: Poloidally averaged time series of density and potential for 2D core-SOL simulations. In both the simulation with FLR and in the drift-limit the density plot shows the initial blowout and transition into turbulence. The potential plot in both cases shows the formation of an unphysical potential. This potential forms due to a zero current assumption in the core

Starting from the poloidally averaged plots in fig. 56 two main differences between the case of FLR simulations and drift-limit can be identified. First the potential formed just inside the last closed flux-surface is affected. The sign of the potential in the core changes, but with our discussion of the unphysical nature of this potential it is difficult to say exactly what is causing our potential

in the core to flip like this. Secondly it can be seen that transport in the FLR case is more coherent than in the drift-fluid limit. This can also be identified visually from snapshots of the density (although it is clearer when viewed as an animation) more coherent blobs are visible in the FLR case.

When fig. 56 is time averaged to obtain radial profiles (fig. 57) it is also clear that there is a difference between the FLR case and the drift-limit.

Drift-limit **FLR**

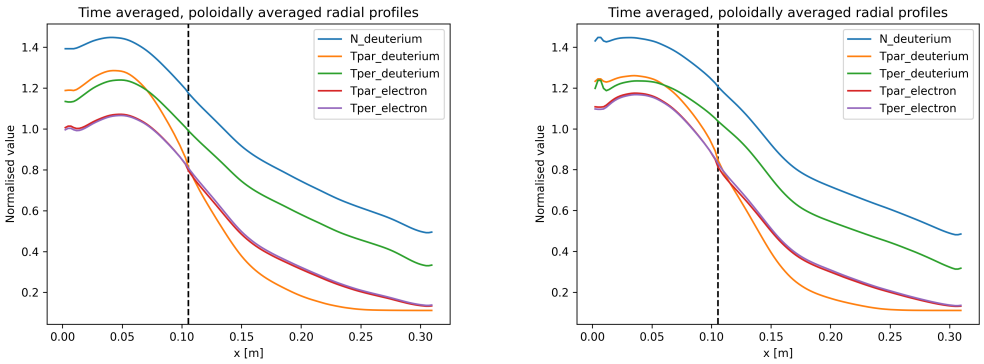

Figure 57: Poloidally averaged, time averaged radial profile of density and temperature for 2D core-SOL simulations. Profiles in the near-SOL are steeper than in the far SOL, however this is due to an unphysical potential forming in the core which gives rise to unphysical shear flow

The density and temperature gradients just outside the last closed flux surface are steeper than in the far SOL. One may be tempted to attribute this to increased parallel collision frequency as ion and electron temperature decreases radially. This is not the case however since in the 2D sheath closure a simple linearised Debye current is assumed and collisions enter only to our temperature equations where they act to reduce temperature anisotropy. Parallel resistivity enters the 3D model through the collisional term in the velocity equation, which in the 2D case is not evolved. Additionally, the presence of the unphysical shear flow arising through the lack of appropriate closure for parallel currents in the core casts doubt on the validity of this steepening. To investigate the edge profiles with more confidence we will look to 3D core-SOL simulations in section 6.2 where our parallel current in the core is more appropriately handled, and the uncertainty introduced by the unphysical potential is removed. However, if the broadening in the far SOL is physical then it is also relevant that in fig. 57 this effect is slightly more pronounced

in the case with FLR effects included. Given that we have shown in previous sections that filaments can be considered as largely non-interacting and have also shown that FLR effects strongly impact the dynamics of small filaments, it is not unreasonable to view the modification of the edge profiles shown in fig. 57 as being driven by FLR effects. Our FLR filaments therefore appear to be generating a profile with a more pronounced broadening in the far sol. By modifying individual filament trajectories the resultant radial profiles are also modified.

6.2 3D Core-SOL Simulations

Table 6: Parameters chosen for 3D core-SOL simulations

Input Parameters	Normalisation Parameters	Derived Parameters
$n_x = 256$	$\rho_s = 1.8 \times 10^{-3}\,\mathrm{m}$	$\nu_e = 1.2 \times 10^{-1}$
$n_y = 24$	$c_s = 3.1 \times 10^{4}\,\mathrm{m\,s^{-1}}$	$\nu_i = 1.9 \times 10^{-3}$
$n_z = 256$	$\Omega_i = 1.7 \times 10^{7}\,\mathrm{s^{-1}}$	
$\nu_{\perp N_i} = 1 \times 10^{-2}$		
$\nu_{\perp N_e} = 1 \times 10^{-2}$		
$T_e = 20\,\mathrm{eV}$		
$T_i = 20\,\mathrm{eV}$		
$N_i = 5 \times 10^{18}\,\mathrm{m^{-3}}$		
$N_e = 5 \times 10^{18}\,\mathrm{m^{-3}}$		
$R = 1.5\,\mathrm{m}$		
$B = 0.5\,\mathrm{T}$		
$l_\parallel = 10\,\mathrm{m}$		

Following on from our 2D coupled core-SOL simulations from section 6.1 we now consider the 3D case. In the 2D case we encountered the problem of an unphysical potential forming in the core which drove a shear flow in the near-SOL. This unphysical shear called into question the radial profiles calculated in the 2D case. Now, in the 3D case, we solve for currents in the core self-consistently, as parallel dynamics including parallel currents and resistivity are included in our moment equations. The boundary conditions listed in fig. 9 are applied. Namely, periodic z-boundary conditions and Neumann boundary conditions at x boundaries. Linearised Debye sheath boundary conditions were applied at the target which is now located only outside the last closed flux surface. Inside the separatrix the field lines are closed and hence periodic boundary conditions are applied inside the separatrix at the y-boundaries wrap the end of the core region back to the start. The parameters chosen for these core-SOL simulations are given in table 6.

With these simulations we hope to generate self-consistent turbulence in our coupled core-SOL simulations from which we can query the power spectrum of the turbulence and compare to experimental results. We will also consider averaged radial profiles, although our simplified geometry is likely to complicate comparison to experiment. Nevertheless, we will calculate the well-known λ_q parameter from our simulations and compare to the λ_q predicted by

the Goldston heuristic and experimental measurements from MAST.

6.2.1 Initial Instability

One major difference between these 3D core-SOL simulations and the 2D core-SOL simulations is that now no perturbation is required to destabilise the pressure gradient that forms in the core. Snapshots of the simulation are shown in figs. 58 to 61. This is due, in part, to the more appropriate handling of parallel currents in the core, in contrast to the 2D case. Once the radial pressure gradient, which is in the same direction as $\nabla\mathbf{B}$ reaches a critical value the gradient is destabilised which leads to a radial blowout that then allows self-consistent pressure-driven turbulence to form. At early times in our simulation the behaviour appears very similar to that described in [91]. In that study the growth rate of the predominant mode early in the simulation was calculated as a function of $k_\perp$ through linearisation of their vorticity equation. This allowed the mode to be identified as the transverse Kelvin-Helmholtz instability. The instability is known to form in the presence of sheared flows. In the case of both simulations, ours and Walkden's the electron temperature gradient drives a radial gradient in the sheath potential which in turn drives an $\mathbf{E}\times\mathbf{B}$ flow in the binormal direction. As such, it is likely that the mode formed at the start of our 3D simulations before we descend into nonlinear turbulence is that same mode, the transverse Kelvin-Helmholtz instability. A similar linearisation as was used to identify the mode should be possible in GEM, at least in the drift-fluid limit. As pointed out in [91] this linear mode is only of interesting as an initial condition for the true saturated turbulent phase. The dispersion relation presented in [91] shows that the fastest growing mode for the Kelvin-Helmholtz instability is inversely proportion to $L_\perp$, the characteristic length scale of the flow profile. One parameter that was varied before identifying the instability was L_z and the initial mode that formed was found to be constant. The dispersion relation makes clear why this was the case, varying L_z did not affect $L_\perp$ and hence didn't affect $k_\perp$. The frequency of this mode is reduced when FLR effects are included which suggests that FLR effects are shifting the fastest growing mode to lower spatial frequencies, possibly through broadening the $\mathbf{E}\times\mathbf{B}$ shear flow profile through gyroaveraging of the potential. It is worth noting that our initial conditions profiles are

particularly susceptible to the Kelvin-Helmholtz instability, a steep gradient in the core with no z variation forms due to our sources. This core gradient is allowed to grow until a perturbation appears in z and leads to an explosive ejection from the core. Our initial condition may be the cause of the physically large wavelength the initial instability takes. It is too large to be attributed to the drift-wave instability and is much longer than the fastest growing Kelvin-Helmholtz mode. A more appropriate initial condition should be adopted in future, possibly matching the 2D "hydrodynamic" approach taken in [91].

While there is a degree of uncertainty around the very first mode that forms, this initial phase of the simulation really only serves to lead into the saturated turbulent state that is more representative of SOL turbulence which we will now consider.

6.3 Average Profiles

Now that we have considered the initial mechanisms that lead into to our saturated turbulent state, we will shift to looking at the average radial profiles during the saturated state.

As with the 2D core-SOL simulations a feature of interest in these simulations is the form the filaments take when ejected from the core and also if any differences in transport are observed on inclusion of FLR effects.

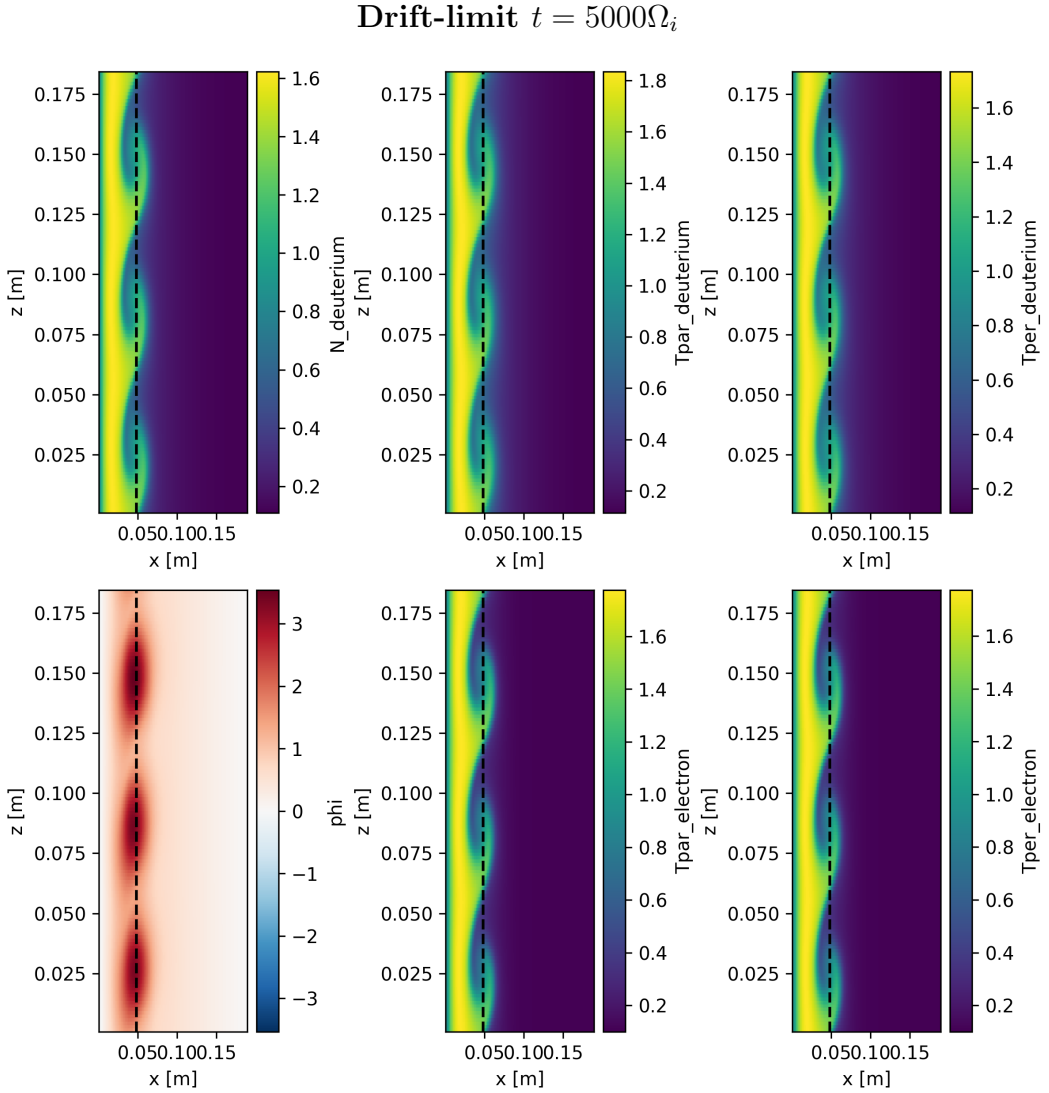

Figure 58: 3D core-SOL simulations with the sheath boundary conditions applied downstream, outside the last closed flux surface (Drift-limit, $t = 5000\Omega_i$). An instability in the core can be observed taking a form similar to a Kelvin-Helmholtz instability. At later times (see figs. 60 to 61) we get an energy cascade to higher frequencies. This instability quickly breaks apart

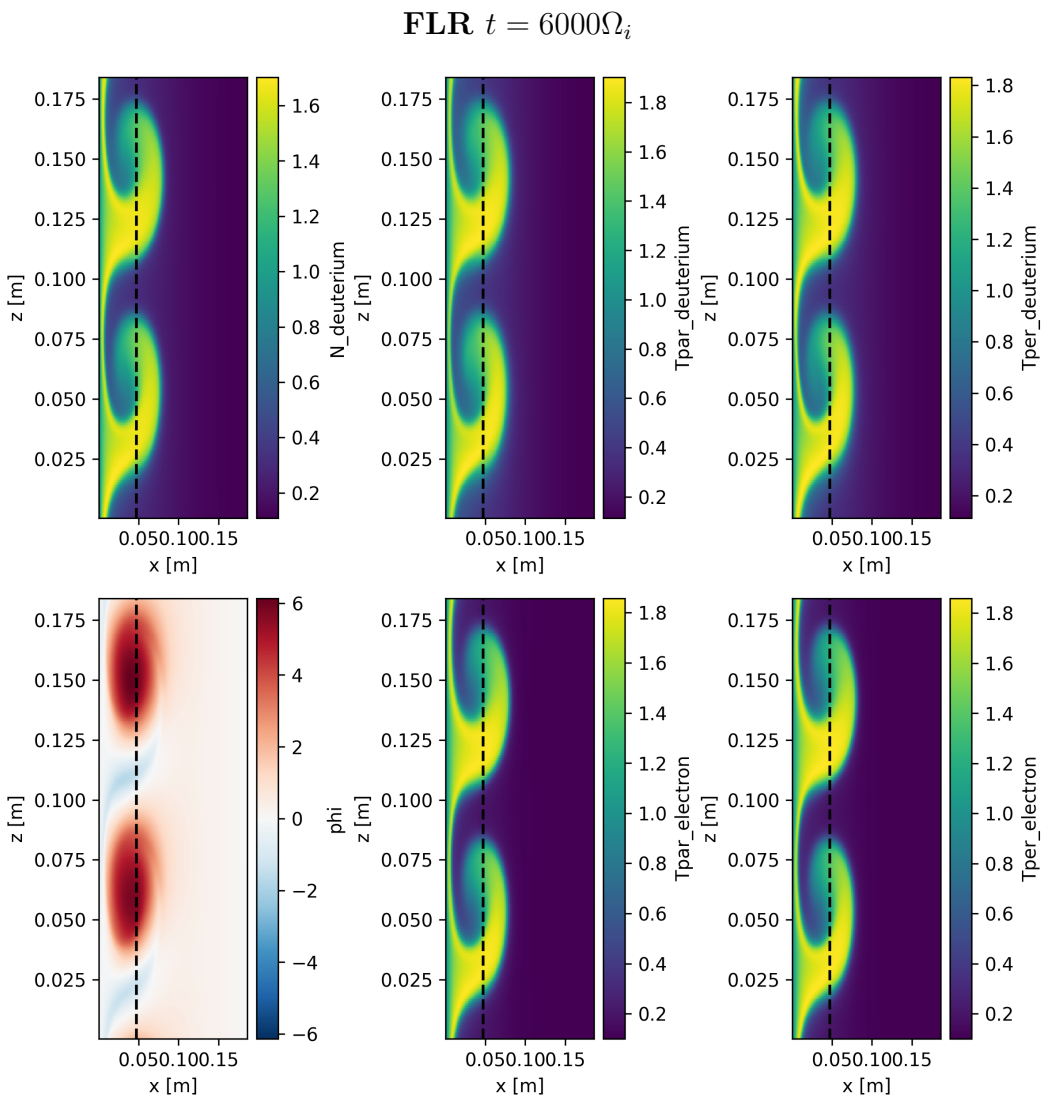

Figure 59: 3D core-SOL simulations with the sheath boundary conditions applied downstream, outside the last closed flux surface. (FLR, $t = 6000\Omega_i$) Interestingly, the instability that forms in the core takes a longer wavelength when FLR effects are included, see fig. 58

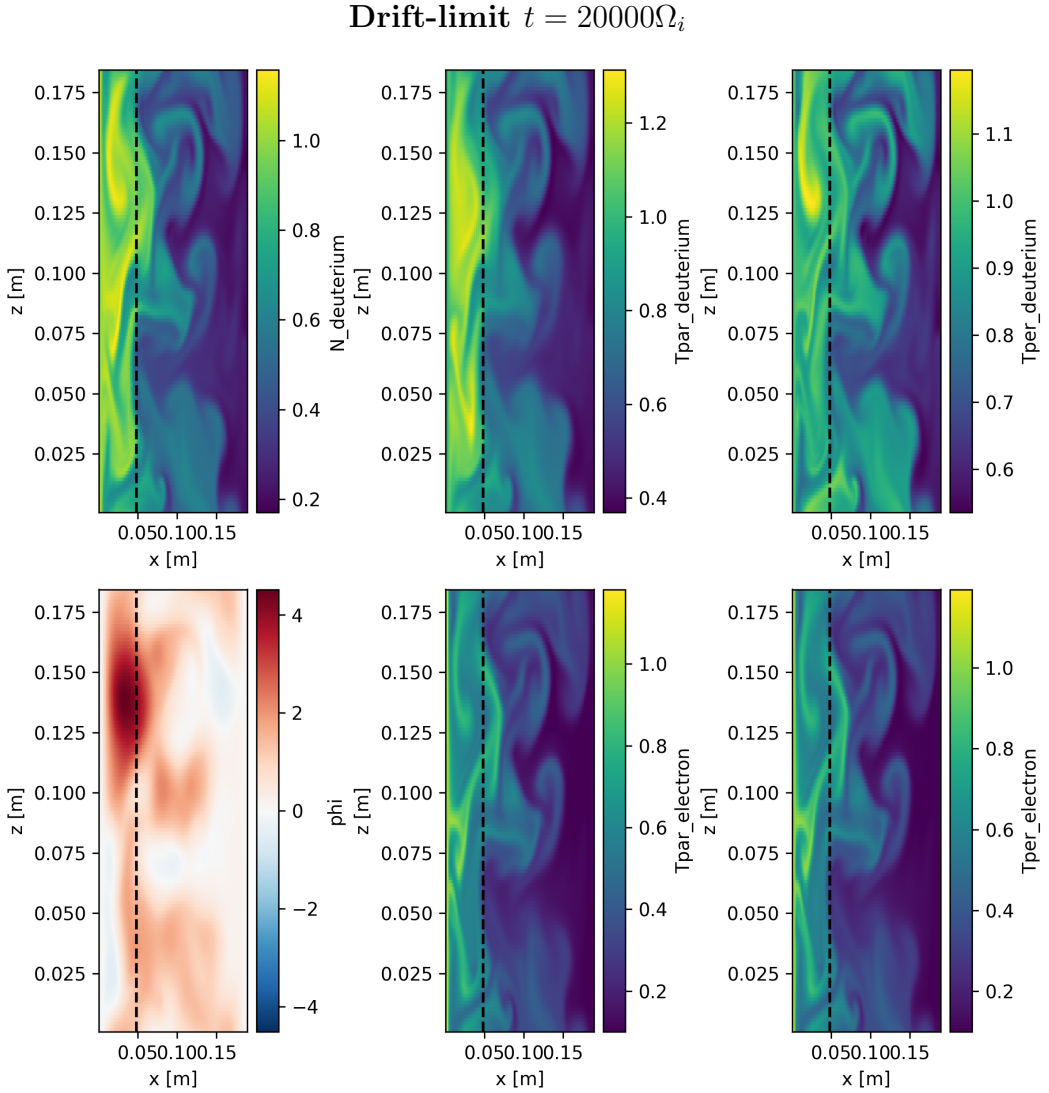

Figure 60: 3D core-SOL simulations with the sheath boundary conditions applied downstream, outside the last closed flux surface (Drift-limit, $t = 20000\Omega_i$) At this point turbulence is continuing in a self-consistent manner, structures which look and propagate similar to our isolated filament simulations are continuously ejected from the core

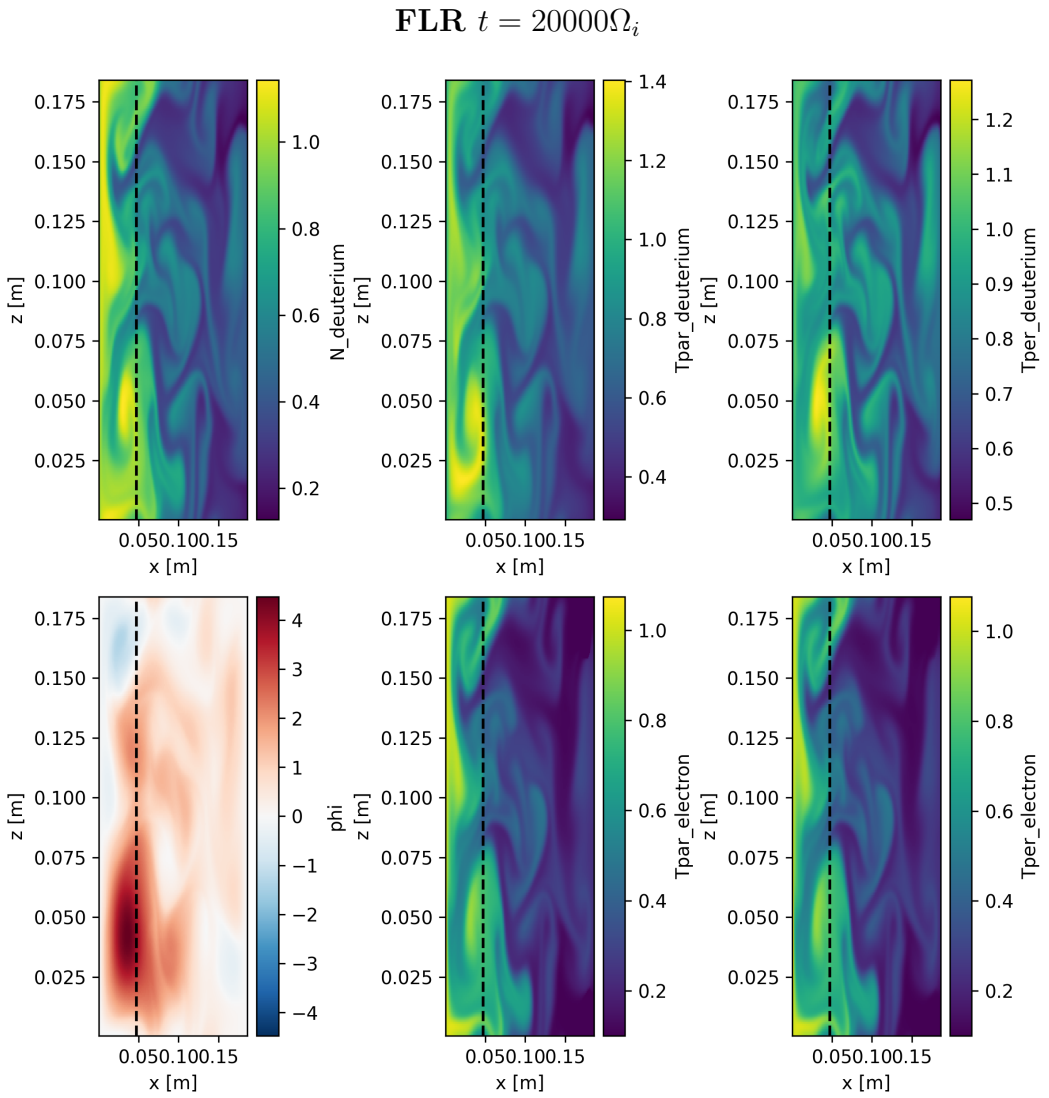

Figure 61: 3D core-SOL simulations with the sheath boundary conditions applied downstream, outside the last closed flux surface (FLR, $t = 20000\Omega_i$) Similar to the drift fluid case in figure 60 structures resembling filaments are generated self consistently and propagate radially outwards

As we did with the 2D core-SOL simulations, we now reduce the data by taking poloidal and temporal averages. These averages are shown in figs. 62 to 63

One important difference between the 3D core-SOL simulations and the 2D is that the spurious potential inside the last closed flux surface that we discussed in section 6.1 is, as expected, not present in 3D. This suggests that our theory regarding the lack of a parallel closure for the core was correct. In the 3D simulations parallel currents allow the potential perturbations in the core to be dissipated. Again, like the 2D simulations, it is clear that transport in the FLR case is much more coherent than in the drift-fluid limit. This can be identified from the higher amplitude, lower frequency components which are visible in the FLR case.

When fig. 62 is time averaged to obtain fig. 63 it is clear that the profile broadening in the far SOL that was observed in the 2D case is not present in the 3D drift-limit simulation. This indicates that the unphysical shear flow associated with the spurious potential in the 2D case was strongly influencing the 2D profiles. The simulation including FLR effects sees a steeper gradient form in the near SOL and a pronounced broadening in the far SOL. Were this due to increased parallel resistivity in the far SOL it should be present in both the drift limit and FLR cases. This is not the case, which suggests that gyroaveraging in drift planes is leading to this effect. The modification to filament velocities and enhanced coherence of FLR filaments may lead to this steepening, as filaments remain more coherent moving past the near SOL and dwell in the far SOL for longer before being dissipated.

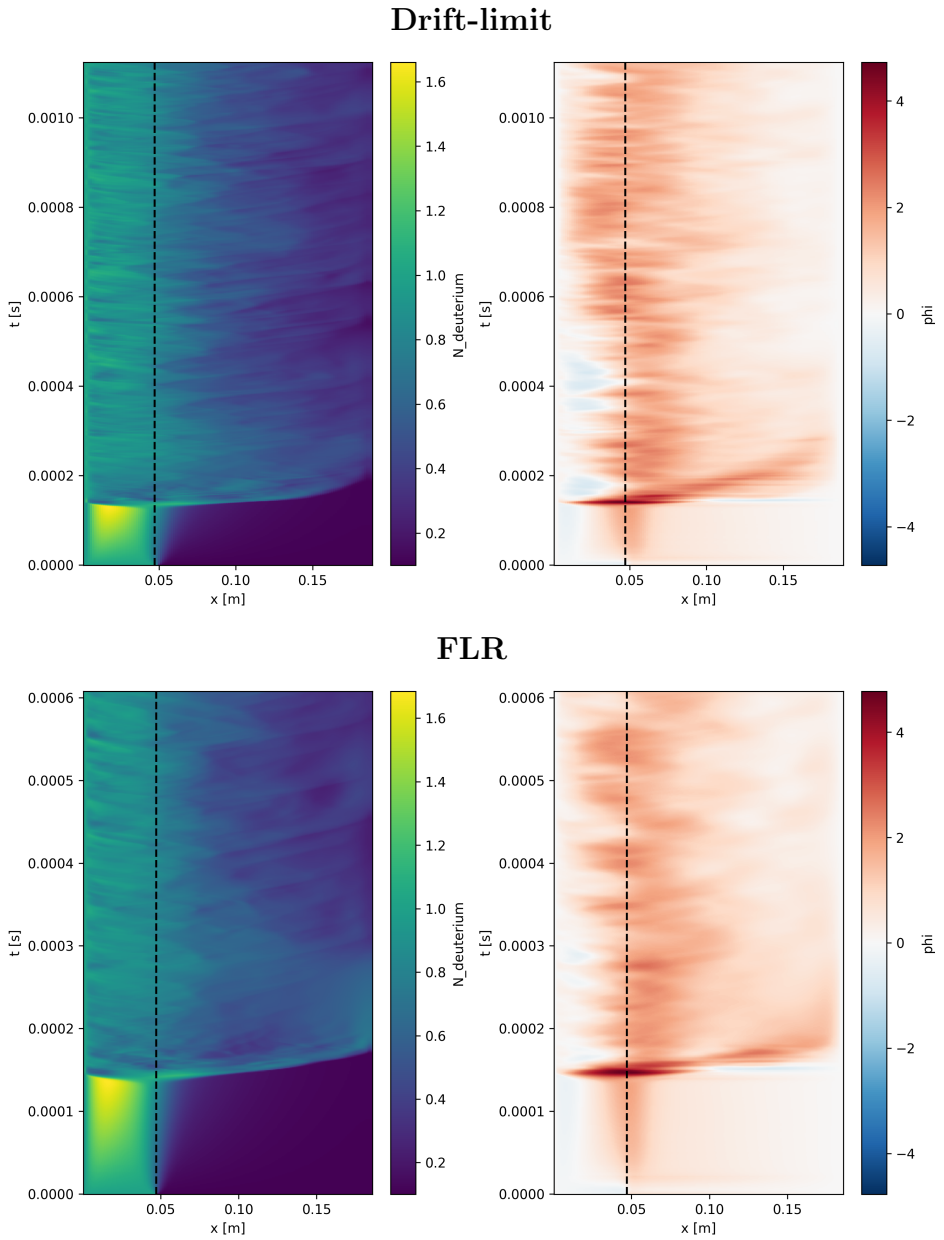

Figure 62: Poloidally averaged time series of density and potential for 3D core-SOL simulations. Note that we no longer see the spurious potential that was formed in the 2D case thanks to the consistent handling of parallel currents in the 3D case

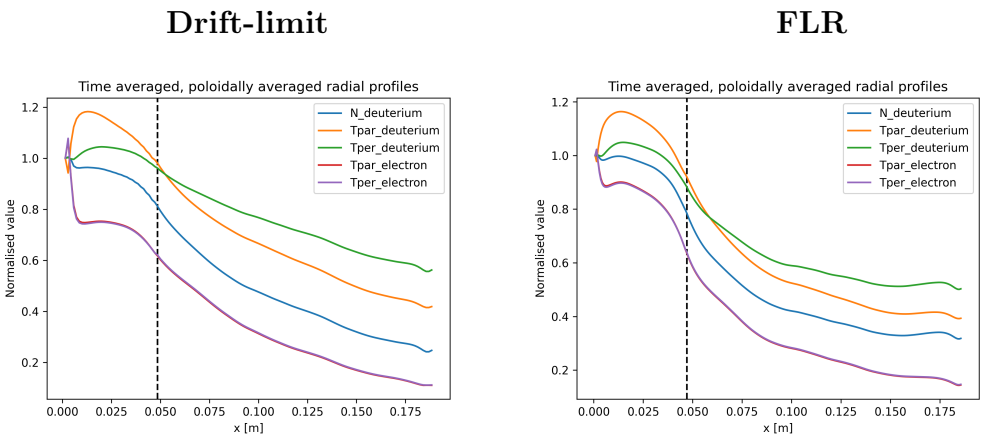

Figure 63: Poloidally averaged time averaged radial profile of density and temperature for 3D core-SOL simulations. When FLR effects are included a steepening of the near SOL profiles and broadening in the far SOL is observed

6.4 Power Spectra

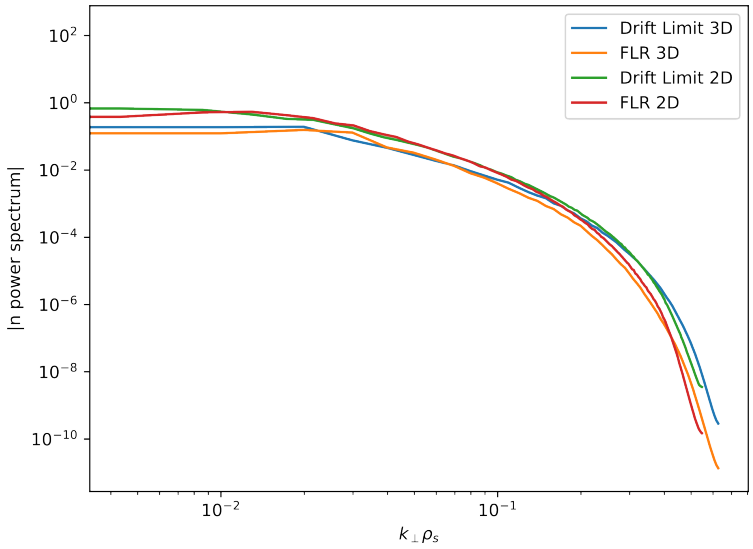

Figure 64: Power spectrum for 2D and 3D SOL turbulence simulations with FLR included and excluded. In both the 2D case and 3D case the inclusion of FLR effects moves the dissipation scale to larger wavelengths

In order to consider energy transfer between scales in our simulations and to investigate dissipation scales we calculate the power spectrum for both the 2D and the 3D simulations in the SOL during the saturated turbulent phase. fig. 64 takes the typical form of an inverse cascade. It can be seen that power enters the system at large wavelengths and is transferred to smaller scales through turbulent processes. When FLR effects are included it can be seen that dissipation at smaller scales is enhanced in both the 2D and 3D cases. This result is further evidence that FLR effects impact small scale dynamics in the edge.

The power spectrum we calculated here matches well with the 2D and 3D power spectra calculated for a drift fluid model in [92]

The correspondence between the 2D and 3D power spectra, and agreement with other 2D and 3D models, suggests that in the 2D case the turbulent cascade to dissipation scales was not strongly affected by the unphysical shear flow that formed just outside the separatrix.

6.5 Parallel Heat Flux

Of obvious interest when simulating the core and SOL is the total parallel heat flux at the target. This quantity is a limiting factor in reactor design due to the tremendous heat loads the divertor must handle. A key quantity when considering parallel heat fluxes in the SOL is λ_q the characteristic radial length scale over which the heat flux density decreases when moving radially outward from the last closed flux surface (LCFS). λ_q can be measured (often using IR thermography) [93, 94]. It can, in theory, be predicted from turbulent simulations as we will attempt to do here, and it can be predicted by heuristic models, one of the most well-known being the Goldston heuristic [95] which has found broad agreement with experimental results. The Goldston heuristic predicts $\lambda_q \approx \frac{2a\rho}{R}$ where a is the minor radius, R is the major radius and ρ is the ion Larmor radius. The model is based on the assumption that grad B and curvature drifts into the SOL are balanced by near-sonic parallel flows to the divertors. The width the Goldston heuristic predicts is relevant mostly for low-gas puff H-mode plasmas. SOL widths of L-mode plasmas are generally wider than those of H-modes. Clearly the simplified slab, core-sol, source driven simulations shown above, with their constant ejection of plasma from the turbulent core region must be considered L-mode. As such, we expect that our calculated λ_q is likely to be greater than that predicted by the Goldston heuristic.

Because we are evolving the parallel heat flux for both ions and electrons, unlike in the 2D case, we can trivially calculate the total normalised heat flux both in the case the FLR effects are included and in the drift limit. We then fit an expression for the radial heat flux decay length λ_q in both cases.

$$\lambda_q = q_0 \exp\left(\frac{-\left(R - R_{LCFS}\right)}{\lambda_q}\right) \tag{177}$$

When λ_q is calculated for the simulation presented here we indeed find they are grossly larger than the value predicted by the Goldston heuristic which predicts, for our chosen plasma parameters $\lambda_q \approx\ 2.8mm$. Compared to the Goldston heuristic our calculated $\lambda_q \approx 7.5cm$ is off by more than an order of magnitude. However, as we mentioned above, our simulations are closer to being representative of L-mode shots than H-mode for which the heuristic

is intended. When experimental values are sought in the literature we find that our λ_q values are of the right order of magnitude for some L-mode shots. For example, in [96] where $\lambda_q = 6.8cm$ at the outer divertor was measured using Langmuir probe data. When a range of shots is considered for regression analysis in [97] a range of values for L-mode λ_q in MAST $\approx 1 - 2cm$ are measured. So while our measured λ_q is in approximate agreement with the shot measured in [96] it is overestimated for the range of shots considered in [97] and grossly overestimated compared to the Goldston heuristic.

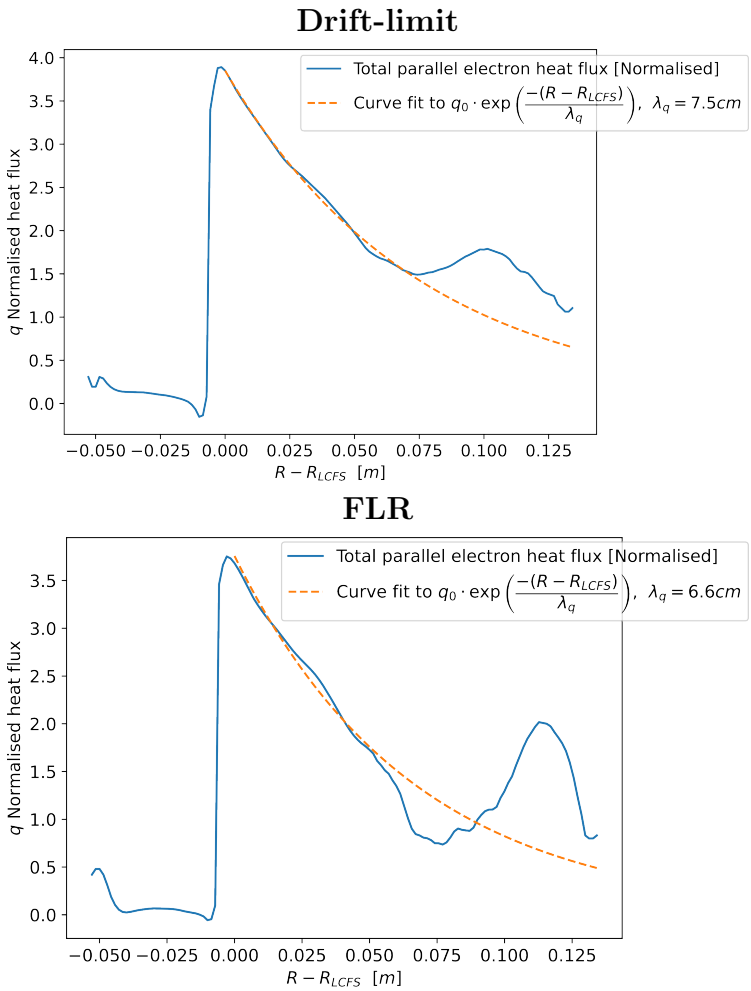

Figure 65: Poloidally averaged, time averaged heat flux density at the target for FLR simulations and in drift-limit, in the drift limit a value for $\lambda_q = 7.5cm$ is obtained whereas in the FLR case $\lambda_q = 6.4cm$. $R - R_{LCFS}$ is the distance from the last closed flux surface

It is however, of interest that our FLR simulation predicts a narrower λ_q than the drift-limit simulation because of the steeper profiles in the near SOL.

One problem that we encounter in these simulations is out placement of

the outer radial boundary. As discussed in section 2.3.5 we have assumed that the dynamics of interest are far from the radial boundaries as they are not physically motivate but rather are a simplification to avoid having to treat the complicated plasma-wall interaction. This assumption held for our filament simulations but for our core-SOL simulation we now see, with our profiles extending radially outward further than intended, that the solution appears to be perturbed near the outer radial wall leading to a bump in the heat flux. Our near-SOL fit for λ_q does not include the region affected by the outer radial boundary, however, to avoid this in future simulations a larger domain size should be employed with a greater radial extent.

7 Conclusions

Understanding the kind of transport associated with filaments is of key importance when designing future reactors with the goal of supplying stable energy reliably. This importance arises from the cross-field transport associated with filaments. If one understands filamentary transport then one is much more adequately equipped to predict and design for relevant cross-field transport. This is crucial in facilitating the design of future divertors and first wall materials. To this end a gyrofluid model with the ability to run in the drift-fluid limit has been implemented in the BOUT++ framework, tested, and used to simulate not just filaments but also self-consistent SOL turbulence.

7.1 Model implementation and verification

The GEM physics model based on the GEM gyrofluid equations [58, 98] has been implemented using the BOUT++ framework with flexibility to include or exclude FLR effects though gyroaveraging. This implementation facilitated isolated filament simulations both in 2D (in section 3) where appropriate closures were derived and implemented, and in 3D (section 4). Filament interaction simulations and finally core-SOL simulations in 2D and 3D were also carried out where a prediction for λ_q was obtained.

7.2 Isolated Filament Simulations & Velocity Scaling Laws

Fluid models have been widely developed and applied to plasma filaments. Here a gyrofluid model has been successfully applied to filament dynamics. Striking agreement has been found between the gyrofluid model taken in the drift-limit and established drift-fluid models, despite being a delta-f model. Furthermore, parameters where FLR effects modify filament dynamics have been identified. To date, drift-fluid models have typically dismissed FLR effects when considering plasma filaments. This assumption, for medium and large filaments, is not unreasonable. However, there are distinct scenarios in which FLR effects have been shown to strongly modify filament behaviour compared to drift-fluid models. Small filaments in the inertial limit have been shown to propagate more slowly and more coherently when FLR effects are

included. This effect is strong enough to disrupt the well-known inertial limit whereby small filaments are expected to propagate with a maximum velocity proportional to the square root of their width. This deviation was grounded in an analytical derivation in section 5

On an individual filament level FLR effects were shown to strongly impact the form and shape of filaments. Namely, the same enhanced coherency found in full-f gyrofluid studies [71, 99] was found in the δf GEM simulations shown here.

7.3 Interacting Filament Simulations & Filament Interactions

Filaments have been shown using drift fluid models to be non-interacting for spacings greater than a few filament widths. Their independence is important for statistical models of filaments. This result has been revisited in section 3.4 and found to hold true even when FLR effects are included. The results of the drift fluid study [43] were reproduced with GEM. It was shown that filament-filament interactions are not strongly impacted by FLR effects and when the filaments separated by more than a few ion Larmor orbits do not strongly interact. This result supports the assumption that filaments can be treated as independent when developing statistical models that attempt to link filament dynamics to radial profiles such as [79]

7.4 Core-SOL Simulations

Core-SOL simulations have been carried out with mixed success. In the case of our 2D simulations an unphysical potential was identified. A mechanism to remedy this unphysical potential and the resultant shear flow for the 2D case has been proposed by way of a Hasegawa-Wakatani closure, following [92].

The 3D core-SOL simulations avoided this spurious potential but were not without issue. Self-consistent turbulence was obtained, for which the mode driving the initial turbulence was identified. The very first mode formed is at a longer wavelength than expected but is attributed to unrealistic initial conditions. After the initial mode generated a radial blowout, turbulence continued self consistently. Power spectra, radial profiles and a prediction for the well-

known heat flux radial decay length λ_q was obtained from the resultant radial profiles. Additionally, a steepening in the near SOL was observed when FLR effects were included. This cannot be attributed to variations in parallel resistivity because the drift-limit simulations experience similar variations but do not show any steepening. It is suggested therefore that the modification to filament dynamics, namely, slower small filaments, and more coherent filaments, led to the variation in radial profile. However, the predicted value, is larger than the generally accepted range of L-Mode λ_q for MAST. We also predict a dependence of λ_q on FLR effects. Core-SOL simulations with FLR effects result in a prediction of λ_q which differs by more than 10% from the drift-fluid prediction. In order to facilitate a better comparison with experimental values, a particular MAST shot should be chosen and simulation parameters matched to it. Additionally, a more realistic geometry should be adopted. The simple slab employed here doesn't capture any of the complexity of the divertor.

7.5 Summary Of Results

The question posed in this thesis is whether FLR effects may impact filament dynamics in such a way that previous drift-fluid results could be considered to have missed some important effects or trends. The answer it seems, is that in certain scenarios FLR effects are particularly important. A robust deviation was found for small inertially-limited filaments both in 2D and 3D and in the cold-ion cold-electron, cold-ion hot-electron, and hot-ion hot-electron cases. This result may be of interest for models that attempt to approximate radial profiles from filament statistic such as with filaments such as [79].

7.5.1 Limitations

The main limitation of GEM for the application we have chosen is its delta-f nature. It would be of interest to repeat these simulations with a full-f gyrofluid model if collisional effects can be included similar fashion as was done with GEM. While, clearly the delta-f nature of GEM is concerning for high amplitude perturbations it is perhaps not unusual for fluid models to contain such limitations. Drift-fluid models for instance will typically include FLR effects only at a reduced level if at all. They also may apply the Boussinesq approximation that isn't strictly valid for large perturbations. Often a cold

ion approximation will be made which is also not strictly valid. All that to say that while GEM has limitations, other models too are not without limitations of their own. The suggestion therefore is that models such as this one may provide a useful, complimentary point of comparison for existing models. The ability of GEM to reproduce the results of drift-fluid models lends credence to the validity of GEM when applied to the systems described here. And the deviations predicted by GEM may provide insight into dynamics not fully captured by drift-fluid models.

7.5.2 Future Work

In section 3.3 we identified a slower than expected convergence rate for our 2D simulations. The method of manufactured solutions should be applied to the GEM physics model. This would formally prove the numerical correctness of the implementation and while it would likely require considerable effort it would pay dividends if maintained when extending the model or adding terms. BOUT++ provides support for MMS testing but, appropriate source terms must be calculated for our moment equations, a task which will be complicated by the gyroaverage operators.

In section 6.1 we identified an issue with the 2D version of GEM applied to core-SOL simulations. A mechanism was proposed to repair the model for this scenario through the derivation of a Hasegawa-Wakatani like closure and is therefore of key importance if the 2D version of GEM is to be applied to core-SOL simulations in the future.

In section 6.2 we identified an issue with our outer radial boundary, this should be remedied either by simply extending the domain radially such that out radial boundary is sufficiently far from the region of interest of the solution or by adopting a more complicated geometry and discarding the slab geometry we used here.

Since a value of λ_q was obtained from the 3D core-SOL simulations it is naturally of interest to now vary parameters that may affect the heat flux decay length. First though, a simulation should be constructed with source terms and plasma parameters that match a chosen MAST L-mode shot for which there is a measurement of λ_q to make a more direct comparison to experimental results than we did in section 6.2

Another feature of this model that wasn't explored in this work is the fact that it can include ion to electron temperature ratios other than $T_i/T_e = 1$. The hot-ion, hot-electron isolated filament simulations of sections 3 and 4 could be repeated with various ratios of ion to electron temperatures to assess the impact of this temperature ratio of filament dynamics. Similarly, electromagnetic effects weren't included in the simulations presented here but are implemented in the physics model. Electromagnetic filament simulations, particularly simulations of interacting filaments would be interesting.

A Code Listings

The below physics model was implemented and revised over the course of the simulations described herein. Additional complexity and higher moments were added as initially simpler subsets of the model were implemented and simulations carried out. Additionally, analysis routines and useful scripts such as the background generation scripts mentioned in section 4.1 were developed. The listing of these supplementary routines is omitted here as they are auxiliary to the physics results discussed above.

Listing 2: GEM physics model implementation

```
#include "gem.hxx"
#include "templates.hxx"

// Allow manual boundary condition at sheath
void Gem::setSheathBoundary(Field3D &var, FieldPerp &val) {
  RangeIterator xrup = mesh->iterateBndryUpperY();
  int j = mesh->yend + 1;
  for (xrup.first(); !xrup.isDone(); xrup.next()) {
    int i = xrup.ind;
    for (int k = 0; k < mesh->LocalNz; ++k) {
        var(i,j,k) = val(i,k);
        var(i, j+1, k) = 3.0*var(i,j,k) - 3.0*var(i,j-1,k) +
   1.0*var(i,j-2,k);
    }
  }
}

void Gem::calcPhi() {
  TRACE("Gem:calcPhi");
  static Field3D rhs = 0.0;
  static Field3D dn = 0.0;
  static Field2D rhosq = 0.0;

  rhs = 0.0;
  dn = 0.0;
  rhosq = 0.0;

  // Currently set up only for two species
  // due to gyroaveraging complications in multi ion scenarios
```

```
29    ASSERT0(n_species==2);
30
31    for (auto s : species_vector) {
32      dn += s->getPhiContribution();
33    }
34
35    dn.applyBoundary("free_o2");
36    mesh->communicate(dn);
37
38    setXGuards(phi, phi_background);
39
40    rhosq = ion->rho*ion->rho;
41    rhosq.applyBoundary("free_o2");
42    mesh->communicate(rhosq);
43    dn *= (ion->tau / ion->a);
44    mesh->communicate(dn);
45    dn.applyBoundary("free_o2");
46    setXGuardsSum(phi, phi_background, dn);
47    phi = phi_solver->solve(dn/rhosq, phi);
48    phi -= dn;
49    setXGuards(phi, phi_background);
50  }
51
52  void Gem::calcApar() {
53    TRACE("Gem::calcApar");
54    static Field3D guess = 0.0;
55    static Field3D rhs = 0.0;
56
57    guess = apar_stag;
58    rhs=0.0;
59
60    rhs.setLocation(CELL_YLOW);
61    const static BoutReal eps=1e-5;
62
63    for (auto s : species_vector) {
64      rhs += s->getAparContribution();
65    }
66
67    // TODO replace std::bind with a lambda function
68    auto f = std::bind(&Gem::calcLeftHandSideApar, this, std::
     placeholders::_1);
```

```
69    // Choice of dim:10 and restart:5 is arbitrary
70    // Should be chosen more rigourously
71    apar_stag = solveGmres(guess, rhs, f, eps, 10, 5);
72
73
74    if (debug) {
75      LHS_apar_apar = calcLeftHandSideApar(apar_stag);
76    }
77  }
78
79  //passed as a callback function for use with the GMRes method
80  Field3D Gem::calcLeftHandSideApar(Field3D& apar_guess) {
81    TRACE("Gem::calcLeftHandSideApar");
82    lhs_apar = 0.0;
83    gyro1_sq_apar = 0.0;
84    gyro2_sq_apar = 0.0;
85
86    apar_guess.applyBoundary("free");
87    for (auto s : species_vector) {
88      gyro1_sq_apar = s->gyro1(apar_guess);
89      gyro2_sq_apar = s->gyro2(apar_guess);
90      gyro1_sq_apar = s->gyro1(gyro1_sq_apar);
91      gyro2_sq_apar = s->gyro2(gyro2_sq_apar);
92      lhs_apar += (s->a * s->beta / s->mu) * (gyro1_sq_apar +
     gyro2_sq_apar);
93    }
94    mesh->communicate(lhs_apar);
95    lhs_apar -= Delp2(apar_guess);
96    return(lhs_apar);
97  }
98
99  void Gem::calcJpar() {
100   TRACE("Gem::calcJpar");
101   jpar_stag = 0.0;
102   for (auto s : species_vector)
103     jpar_stag += s->getJparContribution();
104 }
105
106 Field3D Gem::calcPhiEquilibrium() {
107   // Calculate equilibrium field assuming zero current
108   static Field3D phiEq = 0.0;
```

```
109   phiEq = (ion->mu*electron->tau*electron->Ppar -
110            electron->mu*ion->tau*ion->Ppar);
111   phiEq.applyBoundary("free_o2");
112
113   phiEq /=(electron->mu-ion->mu);
114   FieldPerp phiSheath = extrapSheath(phiEq);
115
116
117   BOUT_FOR(i, phiEq.getMesh()->getRegion3D("RGN_ALL")){
118     phiEq[i] += floatingPot* extrapSheath(electron->Tpar)(i.x()
      , i.z()) -
119       phiSheath(i.x(), i.z());
120
121   }
122   return(phiEq);
123 }
124
125 void Gem::applySheathBoundaryConditions() {
126   ion->sheath_Upar = ion->sheathCouplingFac * extrapSheath(
      electron->Ppar);
127
128   if (force_upar_equal) {
129     electron->sheath_Upar=ion->sheath_Upar;
130     electron->Upar_stag=ion->Upar_stag;
131   }
132   else {
133     electron->sheath_Upar = ion->sheath_Upar -
134         electron->sheathCouplingFac*(extrapSheath(phi -
      floatingPot * electron->Tpar));
135   }
136
137   for (auto s : species_vector){
138     if (s->evolve_A_Upar){
139       setSheathBoundary(s->Upar_stag, s->sheath_Upar);
140     }
141     if (s->evolve_Qpar){
142       s->sheath_Qpar =s->sheathCouplingFac*3* extrapSheath(abs(
      s->tau) * s->Tpar);
143       setSheathBoundary(s->Qpar_stag, s->sheath_Qpar);
144     }
145     if (s->evolve_A_Qper){
```

```
146       s->sheath_Qper = s->sheathCouplingFac* extrapSheath(abs(s
    ->tau) * s->Tper);
147       setSheathBoundary(s->Qper_stag, s->sheath_Qper);
148     }
149   }
150
151
152 }
153
154 int Gem::init(bool restarting) {
155   // read initial conditions
156
157   // add individual species to list of species
158   // and initialise species from options
159   Options *options = Options::getRoot();
160   options = options->getSection("Gem");
161   options->get("n_species", n_species, 2);
162   options->get("calcEquilibrium", calc_equilibrium, 0);
163   options->get("forceUparEqual", force_upar_equal, 0);
164   options->get("filamentFile", filament_file, "filament.nc");
165   options->get("readFilamentFile", read_filament_file, 0);
166   options->get("sheathCouplingFac", sheathCouplingFac, 1);
167   options->get("evolveLogs", evolve_logs, 1);
168   options->get("varycollisionfreq", vary_collision_freq, 1);
169   options->get("driftLimit", drift_limit, 0);
170   options->get("electromagnetic", electromagnetic, 0);
171   // Currently set up only for two species
172   // due to gyroaveraging complications in multi ion scenarios
173   ASSERT0(n_species == 2);
174
175   ASSERT0(!(read_filament_file && calc_equilibrium));
176
177   // initialise each species and store them in a vector for
    easy access
178   for (int i=0; i<n_species; i++){
179     species_vector.push_back(new Species);
180   }
181
182   if (species_vector[0]->is_electron) {
183     ion = species_vector[1];
184     electron = species_vector[0];
```

```
185   }
186   else{
187     ion = species_vector[0];
188     electron = species_vector[1];
189   }
190
191
192   //////////////////
193   // GRID LOADS
194   //////////////////
195
196   // Mesh
197   Field2D Rxy, Bxy, logB, nx, ny, nz, dx, dy, dz;
198   GRID_LOAD(Rxy);  // Major radius [m]
199   GRID_LOAD(Bxy);  // Total B field [T]
200   GRID_LOAD(nx);   // Radial grid cell number
201   GRID_LOAD(dx);   // Radial grid spacing
202   GRID_LOAD(ny);   // Parallel grid cell number
203   GRID_LOAD(dy);   // Parallel grid spacing
204   GRID_LOAD(nz);   // Binormal grid cell number
205   GRID_LOAD(dz);   // Binormal grid spacing
206   GRID_LOAD(logB); // FOR TEST CASE AT LEAST DO THIS
207
208   // Intentionally before norms
209   Lpar = ny * dy;
210   SAVE_ONCE(Lpar);
211   Lperp = (nx-4) * dx;
212   SAVE_ONCE(Lperp);
213   // Average R
214   Lnorm = 0.5 * (max(Rxy, true) + min(Rxy, true));
215   SAVE_ONCE(Lnorm);
216   Bnorm = 0.5 * (max(Bxy, true) + min(Bxy, true));
217   SAVE_ONCE(Bnorm);
218
219   Bxy /= Bnorm;
220
221   // Equilibrium
222   GRID_LOAD(Ni0);
223   Ni0 *= 1e20;
224   Nnorm = 0.5 * (max(Ni0, true) + min(Ni0, true));
225   Ni0 /= Nnorm;
```

```
226   SAVE_ONCE(Nnorm);
227   SAVE_ONCE(Ni0);
228   GRID_LOAD(Te0);
229   Tnorm = 0.5 * (max(Te0, true) + min(Te0, true));
230   Te0 /= Tnorm;
231   SAVE_ONCE(Tnorm);
232   SAVE_ONCE(Te0);
233
234   // Normalisations/physical parameters
235   Cs = sqrt(qe * Tnorm / (Md * ion->getMass()));
236   SAVE_ONCE(Cs);
237   rho_s = sqrt(2.0) * Cs * Md * ion->getMass() / (qe * Bnorm);
238   SAVE_ONCE(rho_s);
239   wci = Cs / rho_s;
240   time_norm = 1.0 / wci;
241   SAVE_ONCE(time_norm);
242   Rxy /= rho_s; // Normalise perp gradient to rho_s
243   output << "\tRxy max is " << max(Rxy, true) << " and min is "
        << min(Rxy, true) << endl;
244   B0 = max(Bxy);
245   R0 = min(Rxy);
246
247   floatingPot = log(sqrt(ion->getMass() / (2 * PI * electron->
       getMass())));
248   for (auto s : species_vector) {
249     BoutReal M_ion, Te, Ne;
250     M_ion = ion->getMass();
251     Te = electron->getTemperature();
252     Ne = electron->getDensity();
253     s->setPars(M_ion, Ne, Te, R0, B0, floatingPot,
254                sheathCouplingFac, Bxy, rho_s);
255   };
256
257
258   // Calculate derived quantities for a check
259   BoutReal beta_e_calc = qe * Tnorm * Nnorm * mu0 / (Bnorm *
       Bnorm);
260   output << "\tCalculated beta_e is " << beta_e_calc << endl;
261
262   // Collisional parameters
263   BoutReal t_e, t_i; // Braginskii collision times
```

```
264
265 // Collision times
266 BoutReal lambda_ei = 24. - log(sqrt(Nnorm / 1e6) / Tnorm);
267 BoutReal lambda_ii = 23. - log(sqrt(2. * Nnorm / 1e6) /
268                       pow(Tnorm*ion->getTemperature(), 1.5));
269
270 t_e = 1. / (2.91e-6 * (Nnorm / 1e6) * lambda_ei * pow(Tnorm,
     -3. / 2));
271 t_i = ion->getMass() / (4.78e-8 * (Nnorm / 1e6) * lambda_ii *
272       pow(Tnorm*ion->getTemperature(), -3. / 2));
273
274 // Normalise
275 BoutReal nu_e, nu_i;
276 nu_e = zeff / (wci * t_e);
277 nu_i = 1. / (wci * t_i);
278
279 electron->setBraginskiiColl(nu_e);
280 ion->setBraginskiiColl(nu_i);
281
282 output << "\tNormalised nu_e = " << nu_e << ", nu_i = " <<
     nu_i << endl;
283
284 // Curvature + normalisation
285 bxcv.covariant = false; // Read contravariant components
286 GRID_LOAD(bxcv);        // Specified components of b0 x kappa
287 GRID_LOAD(sinty);
288 bxcv.z += sinty * bxcv.x;
289 bxcv.x /= Bnorm;
290 bxcv.y *= rho_s * rho_s;
291 bxcv.z *= rho_s * rho_s;
292
293 // Report Normalisations
294 output << "Normalisations\t: " << endl;
295 output << "\tTnorm\t\t: " << Tnorm << endl;
296 output << "\tNnorm\t\t: " << Nnorm << endl;
297 output << "\tLnorm\t\t: " << Lnorm << endl;
298 output << "\tBnorm\t\t: " << Bnorm << endl;
299 output << "\ttime_norm\t: " << time_norm << endl;
300 output << "\tCs\t\t: " << Cs << endl;
301 output << "\trho_s\t\t: " << rho_s << endl;
302 output << "\tLpar\t: " << max(Lpar, true) << " , " << min(
```

```
 Lpar, true) << endl;
303 output << "\tLperp\t: " << max(Lperp, true) << " , " << min(
 Lperp, true) << endl;
304 output << "CalcEquilibrium\t: " << calc_equilibrium << endl;
305
306 /////////////////
307 // MAG. FIELD + METRICS
308 /////////////////
309
310 // Metric components
311
312 mesh->getCoordinates()->Bxy = Bxy;
313 mesh->getCoordinates()->J = 1;
314
315
316 mesh->getCoordinates()->dx /= rho_s;
317 mesh->getCoordinates()->dy /= rho_s;
318 mesh->getCoordinates()->dz /= rho_s;
319 mesh->getCoordinates()->geometry();
320
321 // set location of staggered fields
322 apar_stag.setLocation(CELL_YLOW);
323 phi_stag.setLocation(CELL_YLOW);
324 apar_background.setLocation(CELL_YLOW);
325 gyro1_sq_apar.setLocation(CELL_YLOW);
326 gyro2_sq_apar.setLocation(CELL_YLOW);
327 lhs_apar.setLocation(CELL_YLOW);
328 jpar_stag.setLocation(CELL_YLOW);
329
330 // set boundaries of derived fields
331 phi.setBoundary("phi");
332 phi_stag.setBoundary("phi_stag");
333 apar.setBoundary("apar");
334 apar_stag.setBoundary("apar_stag");
335 jpar.setBoundary("jpar");
336 jpar_stag.setBoundary("jpar_stag");
337
338 // initialise fields to zero
339 phi = 0.0;
340 phi_stag=0.0;
341 apar = 0.0;
```

```
342   apar_stag=0.0;
343   jpar = 0.0;
344   jpar_stag = 0.0;
345   if (debug) {
346     LHS_apar_apar =0;
347     LHS_apar_apar.setLocation(CELL_YLOW);
348   }
349
350   // set up laplacian inversion solvers
351   Options *phi_solver_opts = Options::getRoot()->getSection("
       phi_solver");
352   phi_solver = Laplacian::create(phi_solver_opts);
353   phi_solver->setCoefA(0.0);
354   phi_solver->setCoefD(1.0);
355
356   // add variables to solver
357   for (auto s : species_vector) {
358     auto f = FieldFactory::get();
359     auto opt = Options::getRoot();
360     auto label = s->getLabel();
361     s->N.setBoundary("N_" + s->getLabel());
362     s->N_stag.setBoundary("N_stag_" + s->getLabel());
363     s->Ppar.setBoundary("Ppar_" + s->getLabel());
364     s->Ppar_stag.setBoundary("Ppar_stag_" + s->getLabel());
365     s->Pper.setBoundary("Pper_" + s->getLabel());
366     s->Pper_stag.setBoundary("Pper_stag_" + s->getLabel());
367     s->Ppar_full.setBoundary("Ppar_full_" + s->getLabel());
368     s->Pper_full.setBoundary("Pper_full_" + s->getLabel());
369     s->Upar_stag.setBoundary("Upar_stag_" + s->getLabel());
370     s->Upar.setBoundary("Upar_" + s->getLabel());
371     s->Tpar.setBoundary("Tpar_"+ s->getLabel());
372     s->Tpar_stag.setBoundary("Tpar_stag_"+ s->getLabel());
373     s->Tper.setBoundary("Tper_"+ s->getLabel());
374     s->Tper_stag.setBoundary("Tper_stag_"+ s->getLabel());
375     s->Qpar_stag.setBoundary("Qpar_stag_"+s->getLabel());
376     s->Qpar.setBoundary("Qpar_"+s->getLabel());
377     s->Qper_stag.setBoundary("Qper_stag_"+s->getLabel());
378     s->Qper.setBoundary("Qper_"+s->getLabel());
379
380     if (s->evolve_N) {
381       if (evolve_logs) {
```

```
382         // Solve for log(N) for stability
383         bout_solve(s->logN, ("logN_" + label).c_str());
384         s->N =f->create3D("N_"+ label + ":function", opt, mesh,
     CELL_CENTRE);
385         s->N_stag =f->create3D("N_"+ label + ":function",
386             opt, mesh, CELL_YLOW);
387         // Also dump N
388         dump.addRepeat(s->N, "N_"+label);
389       }
390       else {
391         bout_solve(s->N, ("N_" + label).c_str());
392       }
393     }
394     else {
395       s->N = f->create3D("N_"+ label + ":function", opt, mesh,
    CELL_CENTRE);
396       s->N_stag =f->create3D("N_"+ label + ":function",
397           opt, mesh, CELL_YLOW);
398
399       dump.addOnce(s->N, "N_"+label);
400     }
401
402     if (s->evolve_A_Upar) {
403       bout_solve(s->A_Upar_stag, ("A_Upar_stag_" + label).c_str
    ());
404       dump.addRepeat(s->Upar_stag, "Upar_stag_"+label);
405     }
406     else{
407       s->A_Upar_stag = f->create3D("A_Upar_stag_"+ label + ":
    function",
408           opt, mesh, CELL_YLOW);
409       s->Upar_stag = f->create3D("Upar_stag_"+ label + ":
    function",
410           opt, mesh, CELL_YLOW);
411       dump.addOnce(s->A_Upar_stag, "A_Upar_stag_"+label);
412       dump.addOnce(s->Upar_stag, "Upar_stag_"+label);
413     }
414
415     if (s->evolve_Tpar) {
416       if (evolve_logs) {
417         bout_solve(s->logTpar, ("logTpar_" + label).c_str());
```

```
418         s->Tpar = f->create3D("Tpar_"+ label + ":function",
419             opt, mesh, CELL_CENTRE);
420         dump.addRepeat(s->Tpar, "Tpar_"+label);
421         s->Tpar_stag = interp_to(s->Tpar, CELL_YLOW);
422       }
423       else{
424         bout_solve(s->Tpar, ("Tpar_" + label).c_str());
425       }
426     }
427     else{
428       s->Tpar = f->create3D("Tpar_"+ label + ":function",
429           opt, mesh, CELL_CENTRE);
430       s->Tpar_stag = interp_to(s->Tpar, CELL_YLOW);
431       dump.addOnce(s->Tpar, "Tpar_"+label);
432     }
433
434     if (s->evolve_Tper) {
435       if (evolve_logs) {
436         bout_solve(s->logTper, ("logTper_" + label).c_str());
437         s->Tper = f->create3D("Tper_"+ label + ":function",
438             opt, mesh, CELL_CENTRE);
439         s->Tper_stag = interp_to(s->Tper, CELL_YLOW);
440         dump.addRepeat(s->Tper, "Tper_"+label);
441       }
442       else{
443         bout_solve(s->Tper, ("Tper_" + label).c_str());
444       }
445     }
446     else{
447       s->Tper = f->create3D("Tper_"+ label + ":function",
448           opt, mesh, CELL_CENTRE);
449       s->Tper_stag = f->create3D("Tper_stag_"+ label + ":
    function",
450           opt, mesh, CELL_YLOW);
451     }
452
453     if (s->evolve_Qpar) {
454       bout_solve(s->Qpar_stag, ("Qpar_stag_" + label).c_str());
455       s->Qpar = f->create3D("Qpar_" + label + ":function",
456           opt, mesh, CELL_CENTRE);
457       s->Qpar_stag = f->create3D("Qpar_stag_" + label + ":
```

```
function",
458           opt, mesh, CELL_YLOW);
459     }
460     else{
461       s->Qpar = f->create3D("Qpar_" + label + ":function",
462           opt, mesh, CELL_CENTRE);
463       s->Qpar_stag = f->create3D("Qpar_stag_" + label + ":
function",
464           opt, mesh, CELL_YLOW);
465       dump.addOnce(s->Qpar_stag, "Qpar_stag_"+label);
466     }
467     if (s->evolve_A_Qper) {
468       bout_solve(s->A_Qper_stag, ("A_Qper_stag_" + label).c_str
());
469       s->Qper = f->create3D("Qper_stag_"+label + ":function",
470           opt, mesh, CELL_CENTRE);
471       s->Qper_stag = f->create3D("Qper_stag_"+label + ":
function",
472           opt, mesh, CELL_YLOW);
473       dump.addRepeat(s->Qper_stag, "Qper_stag_"+label);
474     }
475     else{
476       s->A_Qper_stag = f->create3D("A_Qper_stag_" + label + ":
function",
477           opt, mesh, CELL_YLOW);
478       s->Qper = f->create3D("Qper_stag_"+label + ":function",
479           opt, mesh, CELL_CENTRE);
480       s->Qper_stag = f->create3D("Qper_stag_" + label + ":
function",
481           opt, mesh, CELL_YLOW);
482
483       dump.addOnce(s->A_Qper_stag, "A_Qper_stag_"+label);
484       dump.addOnce(s->Qper_stag, "Qper_stag_"+label);
485     }
486
487     s->setSharedFieldPointers(&phi, &apar);
488     s->setSharedStaggeredFieldPointers(&phi_stag, &apar_stag, &
jpar_stag);
489     s->setElectronCollPointer(species_vector);
490     comms += s->getFieldGroup();
491     interpolatedComms += s->getInterpolatedFieldGroup();
```

```
492   }
493   output << "\tField3Ds in comms: " << comms.size_field3d() <<
       endl;
494
495   phi.applyBoundary();
496   phi_stag.applyBoundary();
497   apar.applyBoundary();
498   apar_stag.applyBoundary();
499
500   mesh->communicate(phi, phi_stag, apar, apar_stag);
501
502   for (auto s : species_vector) {
503     s->phi1=0;
504     s->apar1_stag=0;
505     s->phi2=0;
506     s->apar2_stag=0;
507     s->updateDerivedFields();
508     s->splitCompoundFields();
509   }
510
511   phi_background=0.0;
512   apar_background=0.0;
513
514   if (read_filament_file) {
515     if(electromagnetic) {
516       readFromNcFile(apar_background,"apar_stag", filament_file
       );
517     }
518     readFromNcFile(phi_background,"phi", filament_file);
519   }
520
521   if (!restarting && read_filament_file) {
522     if(electromagnetic) {
523       readFromNcFile(apar_stag,"apar_stag", filament_file);
524     }
525     readFromNcFile(phi,"phi", filament_file);
526     phi_stag=interp_to(phi, CELL_YLOW);
527     phi_stag.applyBoundary();
528     mesh->communicate(phi_stag);
529     readFromNcFile(jpar_stag,"jpar_stag", filament_file);
530
```

```
531     for (auto s:species_vector) {
532       s->updateDerivedFields();
533       if(s->evolve_N) {
534         readFromNcFile(s->N, "N_"+s->getLabel(), filament_file)
    ;
535       }
536       if (s->evolve_A_Upar) {
537         readFromNcFile(s->Upar_stag,"Upar_stag_"+s->getLabel(),
     filament_file);
538         s->A_Upar_stag = s->Upar_stag*s->mu + s->beta*s->
    apar1_stag;
539       }
540       if(s->evolve_Tpar){
541         readFromNcFile(s->Tpar, "Tpar_"+s->getLabel(),
    filament_file);
542       }
543       if(s->evolve_Tper){
544         readFromNcFile(s->Tper, "Tper_"+s->getLabel(),
    filament_file);
545       }
546       if(s->evolve_Qpar){
547         readFromNcFile(s->Qpar_stag, "Qpar_stag_"+s->getLabel()
    , filament_file);
548       }
549       if (s->evolve_A_Qper) {
550         readFromNcFile(s->Qper_stag,"Qper_stag_"+s->getLabel(),
     filament_file);
551         s->A_Qper_stag = s->Qper_stag*s->mu  +s->beta*s->
    apar2_stag;
552       }
553     }
554 
555     // Zero vorticity initial conditions
556     electron->N = ion->gyro1(ion->N) -electron->gyro2(electron
    ->Tper) +ion->gyro2(ion->Tper);
557     //ion->N = electron->N-0.5*ion->rho*ion->rho*Delp2(electron
    ->N);
558 
559     for (auto s :species_vector) {
560       s->N.applyBoundary();
561       s->Tpar.applyBoundary();
```

```
562       s->Tper.applyBoundary();
563     }
564     setXGuards(phi, phi_background);
565     calcPhi();
566     phi.applyBoundary();
567   }
568
569   phi.applyBoundary();
570   dump.addRepeat(phi, "phi");
571   if(electromagnetic) {
572     dump.addRepeat(apar_stag, "apar_stag");
573   }
574   else {
575     dump.addOnce(apar_stag, "apar_stag");
576   }
577
578   dump.addRepeat(jpar_stag, "jpar_stag");
579
580   if (debug) {
581     dump.addRepeat(jpar, "jpar");
582     dump.addRepeat(apar, "apar");
583     dump.addRepeat(phi_stag, "phi_stag");
584     dump.addRepeat(LHS_apar_apar, "LHS_apar_apar");
585     dump.addRepeat(electron->electron_coll_stag, "
      electron_coll_stag");
586     for (auto s : species_vector){
587       dump.addRepeat(s->phi1, "phi1_"+s->getLabel());
588       dump.addRepeat(s->phi2, "phi2_"+s->getLabel());
589       dump.addRepeat(s->apar1_stag, "apar1_stag_"+s->getLabel()
      );
590       dump.addRepeat(s->apar2_stag, "apar2_stag_"+s->getLabel()
      );
591     }
592   }
593   // add fields to fieldgroup for communications
594   if (evolve_logs) {
595     for (auto s :species_vector) {
596       s->logN = log(s->N);
597       s->logTpar = log(s->Tpar);
598       s->logTper = log(s->Tper);
599     }
```

```
600   }
601   return(0);
602 }
603
604 int Gem::rhs(BoutReal UNUSED(t)) {
605   // If we're evolving the log of scalar fields then calculate
       fields from
606   // their logs and apply boundaries
607   if (evolve_logs) {
608     for (auto s : species_vector) {
609       s->applyLogLimits();
610       if(s->evolve_N) {
611         s->N=exp(s->logN);
612         s->N.applyBoundary();
613       }
614       if(s->evolve_Tpar) {
615         s->Tpar=exp(s->logTpar);
616         s->Tpar.applyBoundary();
617       }
618       if(s->evolve_Tper) {
619         s->Tper=exp(s->logTper);
620         s->Tper.applyBoundary();
621       }
622       s->applyLimits();
623     }
624   }
625
626   // Communicate after exponentiation
627   mesh->communicate(comms);
628
629   // Calculate electrostatic fields
630   if (calc_equilibrium) {
631     // Calculate equilibrium electrostatic potential assuming
       zero current
632     phi=calcPhiEquilibrium();
633     phi.applyBoundary();
634     mesh->communicate(phi);
635
636     // Calculate potential on staggered grid
637     phi_stag=interp_to(phi, CELL_YLOW);
638     phi_stag.applyBoundary();
```

```
639     mesh->communicate(phi_stag);
640
641     // No electromagnetic effects included in equilibrium case
642     apar = 0.0;
643     apar_stag=0.0;
644
645     // Calculate pressures and gyroaveraged potentials
646     for (auto s : species_vector){
647       s->updateDerivedFields();
648     }
649
650     // Split combined derivatives by subtracting
    electromagnetic component
651     for (auto s : species_vector){
652       s->splitCompoundFields();
653     }
654
655     applySheathBoundaryConditions();
656
657     if (force_upar_equal) {
658       // Forcing zero current
659       electron->Upar_stag = ion->Upar_stag;
660       jpar_stag = 0.0;
661       jpar = 0.0;
662     }
663     else {
664       // Calculate current
665       calcJpar();
666       jpar_stag.applyBoundary();
667       mesh->communicate(jpar_stag);
668       jpar=interp_to(jpar_stag, CELL_CENTRE);
669       jpar.applyBoundary();
670       mesh->communicate(jpar);
671     }
672
673     mesh->communicate(comms);
674
675     for (auto s : species_vector){
676       s->interpolate();
677       s->applyLimits();
678     }
```

```
679
680     mesh->communicate(interpolatedComms);
681
682     if(vary_collision_freq){
683       for(auto s : species_vector) {
684         // Calculate updated braginskii collision frequency
    from updated
685         // temperature and density values
686         s->calcCollFreq(electron);
687       }
688     }
689
690     electron->calcElectronColl();
691     mesh->communicate(electron->electron_coll_stag);
692   }
693   // If we're not calculating 1d profiles
694   else {
695     // Set x guard cells of phi to background for use in
    inversion
696     calcPhi();
697     phi.applyBoundary();
698     mesh->communicate(phi);
699     // Interpolate potential onto staggered grid
700     phi_stag = interp_to(phi, CELL_YLOW);
701     phi_stag.applyBoundary();
702     mesh->communicate(phi_stag);
703
704     if(electromagnetic) {
705       // Calculate electromagnetic field and interpolate onto
706       // centred grid
707       setXGuards(apar_stag, apar_background);
708       calcApar();
709       apar_stag.applyBoundary();
710       mesh->communicate(apar_stag);
711       apar=interp_to(apar_stag, CELL_CENTRE);
712       apar.applyBoundary();
713       mesh->communicate(apar);
714     }
715     else {
716       apar_stag=0.0;
717       apar=0.0;
```

```
718     }
719
720     // Gyroaverage fields and update pressures
721     for (auto s : species_vector) {
722       s->updateDerivedFields();
723     }
724
725     // Split combined derivatives by subtracting
    electromagnetic component
726     for (auto s : species_vector){
727       s->splitCompoundFields();
728     }
729
730     applySheathBoundaryConditions();
731     mesh->communicate(comms);
732     for (auto s : species_vector){
733       s->interpolate();
734     }
735     mesh->communicate(interpolatedComms);
736
737     if (force_upar_equal) {
738       electron->Upar_stag = ion->Upar_stag;
739       electron->Upar = ion->Upar;
740       jpar = 0.0;
741       jpar_stag = 0.0;
742     }
743     else {
744       calcJpar();
745       jpar_stag.applyBoundary();
746       mesh->communicate(jpar_stag);
747       jpar=interp_to(jpar_stag, CELL_CENTRE);
748       jpar.applyBoundary();
749       mesh->communicate(jpar);
750     }
751
752     if(vary_collision_freq){
753       for(auto s : species_vector) {
754         s->calcCollFreq(electron);
755       }
756     }
757
```

```
758     electron->calcElectronColl();
759     mesh->communicate(electron->electron_coll_stag);
760   }
761
762   for (auto s : species_vector) {
763     s->setTimeDerivatives();
764   }
765   return(0);
766 };
767
768 BOUTMAIN(Gem);
```

Listing 3: GEM Species implementation

```
#include "species.hxx"
#include "derivs.hxx"
#include "difops.hxx"
#include "templates.hxx"
#include "bout_types.hxx"
#include "vecops.hxx"

int Species::id = 0;
Species::Species() {
  TRACE("Species constructor");
  my_id = id;
  id++;
  gyroSolver1 = nullptr;
  gyroSolver1_stag = nullptr;

  initFromOptions();
}

Species::~Species() {
  TRACE("Species destructor");
  phi_pt = nullptr;
  apar_pt = nullptr;
  phi_stag_pt = nullptr;
  apar_stag_pt = nullptr;
  jpar_stag_pt = nullptr;
  electron_coll_stag_pt = nullptr;
}

void Species::initFromOptions() {
  TRACE("Initialising parameters from options");

  std::stringstream sec_name;
  sec_name << "species_" << my_id;

  // read shared options and constants
  auto& options = Options::root();
  auto& gem_options = options["GEM"];
  beta              =  gem_options["beta"           ].
   withDefault(1e-3);
  eta               =  gem_options["eta"            ].
```

```
 withDefault(0.51);
calc_equilibrium =  gem_options["calcEquilibrium"].
 withDefault(false);
drift_limit      =  gem_options["driftLimit"     ].
 withDefault(false);
electromagnetic  =  gem_options["electromagnetic"].
 withDefault(0);
evolve_logs      =  gem_options["evolveLogs"     ].
 withDefault(1);

// read species specific options
auto& species_options = options[sec_name.str()];

// TODO
// Do validation on electron parameters
// For example, do not allow positive
// electron charge
is_electron = species_options["is_electron"].withDefault(
 false);

// set which moments are being evolved from options
evolve_N      = species_options["evolve_N"     ].withDefault(
 true);
evolve_A_Upar = species_options["evolve_A_Upar"].withDefault(
 true);
evolve_Tpar   = species_options["evolve_Tpar"  ].withDefault(
 true);
evolve_Tper   = species_options["evolve_Tper"  ].withDefault(
 true);
evolve_Qpar   = species_options["evolve_Qpar"  ].withDefault(
 true);
evolve_A_Qper = species_options["evolve_A_Qper"].withDefault(
 true);

// set numerical constants and physical parameters
N0                    = species_options["N0"
     ].withDefault(1.0);
T0                    = species_options["T0"
     ].withDefault(1.0);
 // Normalised to deuterium mass
mass                  = species_options["mass"
```

```
      ].withDefault(1.0);
// Normalised to electron charge
charge                  = species_options["charge"
      ].withDefault(1.0);
density                 = species_options["density"
      ].withDefault(1.0); // In units of nref
temperature             = species_options["temperature"
      ].withDefault(1.0); // In units of Tref
alpha                   = species_options["alpha"
      ].withDefault(-1.0);
kappa                   = species_options["kappa"
      ].withDefault(-1.0);
pi                      = species_options["pi"
      ].withDefault(-1.0);
eta                     = species_options["eta"
      ].withDefault(-1.0);
NMin                    = species_options["NMin"
      ].withDefault(0.1);
TparMin                 = species_options["TparMin"
      ].withDefault(0.1);
TperMin                 = species_options["TperMin"
      ].withDefault(0.1);
label                   = species_options["Label"
      ].withDefault("unlabled");
include_density_source = species_options["
 include_density_source"].withDefault(false);
include_heat_source     = species_options["include_heat_source
 "    ].withDefault(false);
include_coll_disp       = species_options["include_coll_disp"
      ].withDefault(true);

// set internal settings
auto& diffusion_options = species_options["Diffusion"];
auto diff_par_default   = diffusion_options["diff_par"].
 withDefault(0.0);
auto diff_perp_default = diffusion_options["diff_perp"].
 withDefault(0.0);
for(int i=0; i<6; i++){
  diff_par[i]        = diffusion_options[moment_list[i]]["
 diff_par"].withDefault(diff_par_default);
  diff_perp[i]       = diffusion_options[moment_list[i]]["
```

```
 diff_perp"].withDefault(diff_perp_default);
}

N_stag.setLocation(CELL_YLOW);
A_Upar_stag.setLocation(CELL_YLOW);
Upar_stag.setLocation(CELL_YLOW);
Tpar_stag.setLocation(CELL_YLOW);
Tper_stag.setLocation(CELL_YLOW);
Qpar_stag.setLocation(CELL_YLOW);
A_Qper_stag.setLocation(CELL_YLOW);
Qper_stag.setLocation(CELL_YLOW);
Ppar_stag.setLocation(CELL_YLOW);
Pper_stag.setLocation(CELL_YLOW);
phi1_stag.setLocation(CELL_YLOW);
phi2_stag.setLocation(CELL_YLOW);
apar1_stag.setLocation(CELL_YLOW);
apar2_stag.setLocation(CELL_YLOW);
apar2Guess_stag.setLocation(CELL_YLOW);
electron_coll_stag.setLocation(CELL_YLOW);
nu_stag.setLocation(CELL_YLOW);
Bxy_stag.setLocation(CELL_YLOW);
rho_stag.setLocation(CELL_YLOW);

 // set up comm group
Ppar = 0.0;
Ppar_full = 0.0;
Pper = 0.0;
Pper_full = 0.0;
electron_coll_stag = 0.0;

logN.allocate();
logTpar.allocate();
logTper.allocate();

if (evolve_N) {
  comms.add(logN, N, Ppar, Pper, Ppar_full, Pper_full);
  interpolatedComms.add(N_stag, Ppar_stag, Pper_stag);
}
if (evolve_A_Upar) {
  comms.add(A_Upar_stag, Upar_stag);
  interpolatedComms.add(Upar);
```

```
130   }
131   if (evolve_Tpar) {
132     comms.add(logTpar, Tpar);
133     interpolatedComms.add(Tpar_stag);
134   }
135   if (evolve_Tper) {
136     comms.add(logTper, Tper);
137     interpolatedComms.add(Tper_stag);
138   }
139   if (evolve_Qpar) {
140     comms.add(Qpar_stag);
141     interpolatedComms.add(Qpar);
142   }
143   if (evolve_A_Qper) {
144     comms.add(A_Qper_stag, Qper_stag);
145     interpolatedComms.add(Qper);
146   }
147   if (is_electron) {
148     comms.add(electron_coll_stag);
149   }
150
151 }
152
153 // Set the ratio parameters
154 void Species::setPars(BoutReal &mass_ref, BoutReal &dens_ref,
155                       BoutReal &temp_ref, BoutReal &R0_in,
156                       BoutReal &B0_in, BoutReal &
    floating_pot_in,
157                       BoutReal &sheath_coupling_fac_in, Field2D
     &bxy,
158                       BoutReal &rho_ion) {
159
160   auto& options = Options::root();
161
162   mu = mass / (charge * mass_ref);
163   a = density * charge / (dens_ref);
164   tau = temperature / (charge * temp_ref);
165   sheathCouplingFac = sheath_coupling_fac_in;
166   floatingPot = floating_pot_in;
167
168
```

```
169   rho = sqrt(fabs(mu * tau));
170   rho_stag = interp_to(rho, CELL_YLOW);
171   rho_stag.applyBoundary("free_o2");
172
173   // Gyro 1
174   auto& gyro1_opts = options["gyro1"];
175   gyroSolver1 = Laplacian::create(&gyro1_opts);
176   gyroSolver1->setCoefD(-0.5 * rho * rho);
177   gyroSolver1->setCoefA(1.0);
178
179
180   // Gyro 1 stag
181   gyroSolver1_stag = Laplacian::create(&gyro1_opts, CELL_YLOW);
182   gyroSolver1_stag->setCoefD(-0.5 * rho_stag * rho_stag);
183   gyroSolver1_stag->setCoefA(1.0);
184
185   B0 = B0_in;
186   R0 = R0_in;
187   g = -2.0 / (B0 * R0);
188
189   output << "mu_"                 << label << " : " << mu
                << std::endl;
190   output << "a_"                  << label << " : " << a
                << std::endl;
191   output << "tau_"                << label << " : " << tau
                << std::endl;
192   output << "sheathCouplingFac_" << label << " : " <<
       sheathCouplingFac << std::endl;
193   output << "floatingPot_"        << label << " : " <<
       floatingPot       << std::endl;
194   output << "g"                   << label << " : " << g
                << std::endl;
195   Bxy = bxy;
196
197   Bxy.applyBoundary("free_o2");
198   mesh->communicate(Bxy);
199   Bxy_stag = interp_to(Bxy, CELL_YLOW);
200   Bxy_stag.applyBoundary("free_o2");
201   lgB = log(Bxy);
202   lgB_stag = log(Bxy_stag);
203
```

```
204   FieldFactory f(mesh);
205   std::string N_source_label     = "Sn_"      + label;
206   std::string Tpar_source_label  = "St_par_"  + label;
207   std::string Tper_source_label  = "St_perp_" + label;
208
209   source           = f.create3D("function", &options[
       N_source_label], mesh);
210   source_stag      = f.create3D("function", &options[
       N_source_label], mesh, CELL_YLOW);
211   heat_source_par  = f.create3D("function", &options[
       Tpar_source_label], mesh);
212   heat_source_perp = f.create3D("function", &options[
       Tper_source_label], mesh);
213
214   source      *= rho_ion;
215   source_stag *= rho_ion;
216   heat_source_par  *= rho_ion;
217   heat_source_perp *= rho_ion;
218
219   dump.addOnce(source,          "Sn_"      + label);
220   dump.addOnce(source_stag,     "Sn_stag_" + label);
221   dump.addOnce(heat_source_par, "St_par_"  + label);
222   dump.addOnce(heat_source_perp,"St_perp_" + label);
223 }
224
225 // Gamma_1 gyroaveraging operator
226 Field3D Species::gyro1(const Field3D &fld, const Field3D &guess
       ) {
227   TRACE("species::gyro1(3D)");
228
229   if (calc_equilibrium || drift_limit) {
230     return (fld);
231   }
232
233   CELL_LOC loc_out = fld.getLocation();
234   static Field3D result = 0.0;
235   result.setLocation(loc_out);
236
237   result = myGyroPade1(fld, guess);
238   mesh->communicate(result); // Required
239   result.applyBoundary("free_o2");
```

```
240   return (result);
241 }
242
243 // Gamma_1 gyroaveraging operator without guess
244 Field3D Species::gyro1(const Field3D &fld) {
245   TRACE("species::gyro1(3D)");
246
247   if (calc_equilibrium || drift_limit) {
248     return (fld);
249   }
250
251   CELL_LOC loc_out = fld.getLocation();
252   static Field3D result = 0.0;
253   result.setLocation(loc_out);
254
255   result = myGyroPade1(fld);
256   mesh->communicate(result); // Required
257   result.applyBoundary("free_o2");
258   return (result);
259 }
260
261  // Gamma_2 gyroaveraging operator
262  Field3D Species::gyro2(const Field3D &fld, const Field3D &
     gyro_1_guess,
263                         Field3D & intermediate_guess) {
264    if (calc_equilibrium || drift_limit)
265      return(0.0);
266    else
267      return myGyroPade2(fld, gyro_1_guess, intermediate_guess);
268  }
269
270 // Gamma_2 gyroaveraging operator without guess
271  Field3D Species::gyro2(const Field3D &fld) {
272    if (calc_equilibrium || drift_limit)
273      return(0.0);
274    else
275      return myGyroPade2(fld);
276  }
277
278 Field3D Species::myGyroPade1(const Field3D &fld, const Field3D
     &guess) {
```

```
279    CELL_LOC loc_out = fld.getLocation();
280    switch (loc_out) {
281      case CELL_YLOW : return gyroSolver1_stag->solve(fld, guess
       );
282      default : return gyroSolver1->solve(fld, guess);
283    }
284  }
285
286 Field3D Species::myGyroPade1(const Field3D &fld) {
287    CELL_LOC loc_out = fld.getLocation();
288    switch (loc_out) {
289      case CELL_YLOW : return gyroSolver1_stag->solve(fld);
290      default : return gyroSolver1->solve(fld);
291    }
292  }
293
294  Field3D Species::myGyroPade2(const Field3D &fld,
295                               const Field3D & gyro_1_guess,
296                               Field3D & intermediate_guess) {
297    TRACE("species::myGyroPade2(3D)");
298    CELL_LOC loc_out = fld.getLocation();
299    Field3D result = myGyroPade1(myGyroPade1(fld, gyro_1_guess),
        intermediate_guess);
300    mesh->communicate(result);
301    intermediate_guess = result;
302    switch (loc_out) {
303      case CELL_YLOW : result = 0.5 * rho_stag * rho_stag *
       Delp2(result, loc_out);
304                       break;
305      default : result = 0.5 * rho * rho * Delp2(result, loc_out
       );
306    }
307    mesh->communicate(result);
308    result.applyBoundary("dirichlet_o2");
309    return result;
310  }
311
312 Field3D Species::myGyroPade2(const Field3D &fld) {
313    TRACE("species::myGyroPade2(3D)");
314    CELL_LOC loc_out = fld.getLocation();
315    Field3D result = myGyroPade1(myGyroPade1(fld));
```

```
316     mesh->communicate(result);
317     switch (loc_out) {
318       case CELL_YLOW : result = 0.5 * rho_stag * rho_stag *
    Delp2(result, loc_out);
319                        break;
320       default : result = 0.5 * rho * rho * Delp2(result, loc_out
    );
321     }
322     mesh->communicate(result);
323     result.applyBoundary("dirichlet_o2");
324     return result;
325  }
326
327
328 Field3D Species::curvature(const Field3D &fld, CELL_LOC loc_out
    ) const {
329   TRACE("Species::curvature");
330   // Approximating curvature for a flux tube geometry
331   return (g * DDZ(fld, loc_out));
332 }
333
334 Field3D Species::gradParNLlgB(CELL_LOC loc_out) {
335   TRACE("Species::gradParNLlgB");
336   // Still need to make flux tube approximation here!!
337   // similar to curvature operator
338   switch (loc_out) {
339   case (CELL_YLOW):
340     return (Grad_par(lgB, loc_out) -
341             beta * bracket(apar1_stag, lgB_stag, BRACKET_MODE,
    loc_out));
342   default:
343     return (Grad_par(lgB, loc_out) -
344             beta * bracket(apar1, lgB, BRACKET_MODE, loc_out));
345   }
346 }
347
348 std::string Species::getLabel() { return (label); }
349
350 BoutReal Species::getMass() { return (mass); }
351
352 BoutReal Species::getDensity() { return (density); }
```

```
353
354 BoutReal Species::getTemperature() { return (temperature); }
355
356 void Species::setSharedFieldPointers(Field3D *phi, Field3D *
    apar) {
357   TRACE("Species::setSharedFieldPointers");
358   phi_pt = phi;
359   apar_pt = apar;
360 }
361
362 void Species::setSharedStaggeredFieldPointers(Field3D *phi_stag
    ,
363                                               Field3D *
    apar_stag,
364                                               Field3D *
    jpar_stag) {
365   TRACE("Species::setSharedStaggeredFieldPointers");
366   phi_stag_pt = phi_stag;
367   apar_stag_pt = apar_stag;
368   jpar_stag_pt = jpar_stag;
369 }
370
371 void Species::setElectronCollPointer(const std::vector<Species
    *>& species_vector) {
372   for (auto s : species_vector) {
373     if (s->is_electron) {
374       electron_coll_stag_pt = &s->electron_coll_stag;
375     }
376   }
377 }
378
379 void Species::updateDerivedFields() {
380   TRACE("Species::updateDerivedFields");
381   // Gyroaverage potential
382   if (!calc_equilibrium && !drift_limit) {
383     phi1 = gyro1((*phi_pt));
384     phi2 = gyro2((*phi_pt));
385     phi1_stag = gyro1((*phi_stag_pt));
386     phi2_stag = gyro2((*phi_stag_pt));
387
388     if (electromagnetic) {
```

```
389       apar1_stag = gyro1((*apar_stag_pt), apar1_stag);
390       apar2_stag = gyro2((*apar_stag_pt), apar1_stag,
    apar2Guess_stag);
391       apar1 = interp_to(apar1_stag, CELL_CENTRE);
392       apar1.applyBoundary("free_o2");
393       apar2 = interp_to(apar2_stag, CELL_CENTRE);
394       apar2.applyBoundary("dirichlet_o2");
395     }
396     else {
397       apar1 = 0.0;
398       apar2 = 0.0;
399       apar1_stag = 0.0;
400       apar2_stag = 0.0;
401     }
402     mesh->communicate(phi1_stag, phi2_stag, apar1, apar2);
403   }
404   else {
405     phi1 = *phi_pt;
406     phi1_stag = *phi_stag_pt;
407     phi2 = 0.0;
408     phi2_stag = 0.0;
409     apar1 = *apar_pt;
410     apar1_stag = *apar_stag_pt;
411     apar2 = 0.0;
412     apar2_stag = 0.0;
413   }
414
415   // Update pressures
416   Ppar = N + Tpar;
417   Ppar_full = N * Tpar;
418   Pper = N + Tper;
419   Pper_full = N * Tper;
420   Ppar.applyBoundary();
421   Ppar_full.applyBoundary();
422   Pper.applyBoundary();
423   Pper_full.applyBoundary();
424   mesh->communicate(Ppar, Pper, Ppar_full, Pper_full);
425 }
426
427 void Species::splitCompoundFields() {
428   TRACE("Species::splitCompoundFields");
```

```
429   if (evolve_A_Upar) {
430     Upar_stag = (A_Upar_stag - beta * apar1_stag) / mu;
431     Upar_stag.applyBoundary();
432   }
433   if (evolve_A_Qper) {
434     Qper_stag = (A_Qper_stag - beta * apar2_stag) / mu;
435     Qper_stag.applyBoundary();
436   }
437   mesh->communicate(Upar_stag, Qper_stag);
438 }
439
440 void Species::interpolate() {
441   if (evolve_N){
442     N_stag = interp_to(N, CELL_YLOW, "RGN_NOBNDRY");
443     N_stag.applyBoundary();
444     Ppar_stag = interp_to(Ppar, CELL_YLOW, "RGN_NOBNDRY");
445     Ppar_stag.applyBoundary();
446     Pper_stag = interp_to(Pper, CELL_YLOW, "RGN_NOBNDRY");
447     Pper_stag.applyBoundary();
448   }
449   if (evolve_A_Upar){
450     Upar = interp_to(Upar_stag, CELL_CENTRE, "RGN_NOBNDRY");
451     Upar.applyBoundary();
452   }
453   if (evolve_Tpar) {
454     Tpar_stag = interp_to(Tpar, CELL_YLOW, "RGN_NOBNDRY");
455     Tpar_stag.applyBoundary();
456   }
457   if (evolve_Tper) {
458     Tper_stag = interp_to(Tper, CELL_YLOW, "RGN_NOBNDRY");
459     Tper_stag.applyBoundary();
460   }
461   if (evolve_Qpar) {
462     Qpar = interp_to(Qpar_stag, CELL_CENTRE, "RGN_NOBNDRY");
463     Qpar.applyBoundary();
464   }
465   if (evolve_A_Qper) {
466     Qper = interp_to(Qper_stag, CELL_CENTRE, "RGN_NOBNDRY");
467     Qper.applyBoundary();
468   }
469 }
```

```
470
471 // Return species contribution to RHS of polarisation equation
472 Field3D Species::getPhiContribution() {
473   TRACE("Species::getPhiContribution");
474
475   static Field3D tmp_1 = 0.0;
476   static Field3D tmp_2 = 0.0;
477   static Field3D tmp_3 = 0.0;
478
479   if (evolve_N) {
480     tmp_1 = gyro1(N, tmp_1);
481   }
482   if (evolve_Tper){
483     tmp_3 = gyro2(Tper);
484   }
485
486   return (-a*(tmp_1 + tmp_3));
487 }
488
489 // Return species contribution to RHS of induction equation
490 Field3D Species::getAparContribution() {
491   TRACE("Species::getAparContribution");
492
493   static Field3D tmp_1 = 0.0;
494   static Field3D tmp_2 = 0.0;
495   static Field3D tmp_3 = 0.0;
496   tmp_1.setLocation(CELL_YLOW);
497   tmp_2.setLocation(CELL_YLOW);
498   tmp_3.setLocation(CELL_YLOW);
499   if (evolve_A_Upar) {
500     tmp_1 = gyro1(A_Upar_stag, tmp_1);
501   }
502   if (evolve_A_Qper) {
503     tmp_2 = gyro1(A_Qper_stag, tmp_2);
504     tmp_3 = gyro2(A_Qper_stag, tmp_2, tmp_3);
505   }
506   return (a/mu*(tmp_1 + tmp_3));
507 }
508
509 // Return species contribution to current density
510 Field3D Species::getJparContribution() const {
```

```
511   TRACE("Species::getJparContribution");
512   return (a * Upar_stag);
513 }
514
515 // TODO: Limit collision frequency when temperature is near
      zero
516 void Species::calcCollFreq(Species* electron){
517   nu = nu_0*electron->N/pow(Tpar, 1.5);
518   nu_stag = nu_0*electron->N_stag/pow(Tpar_stag, 1.5);
519
520 }
521
522 void Species::calcElectronColl() {
523   TRACE("Species::calcElectronColl");
524   if (is_electron) {
525     electron_coll_stag =
526         mu * nu_stag *
527         (eta * (*jpar_stag_pt) +
528          (alpha / kappa) * (Qpar_stag + Qper_stag + alpha * (*
      jpar_stag_pt)));
529   }
530   else {
531     electron_coll_stag = 0.0;
532   }
533 }
534
535 void Species::setBraginskiiColl(BoutReal nu_in) {
536   nu_0 = nu_in;
537   nu = nu_0;
538   nu_stag = nu_0;
539   dump.addOnce(nu_0, "nu_0" + label);
540 }
541
542 void Species::addCollisionalDissipation() {
543   TRACE("Species::addCollisionalDissipation");
544   if (evolve_A_Upar) {
545     ddt(A_Upar_stag) += (*electron_coll_stag_pt);
546   }
547
548   if (evolve_Tpar) {
549     ddt(logTpar) -=  2*(nu / (3.0 * pi)) * (Tpar - Tper);
```

```
550   }
551
552   if (evolve_Tper) {
553     ddt(logTper) += (nu / (3.0 * pi)) * (Tpar - Tper);
554   }
555
556   if (evolve_Qpar) {
557     ddt(Qpar_stag) -= nu_stag * (2.5 / kappa) *
558                       (Qpar_stag + 1.28 * (Qpar_stag - 1.5 *
      Qper_stag) +
559                        0.6 * alpha * (*jpar_stag_pt));
560   }
561
562   if (evolve_A_Qper){
563     ddt(A_Qper_stag) -= mu * nu_stag * (2.5 / kappa) *
564                       (Qper_stag - 1.28 * (Qpar_stag - 1.5 *
      Qper_stag) +
565                        0.4 * alpha * (*jpar_stag_pt));
566   }
567 }
568
569 // TODO: Implement Landau damping
570 __attribute__((unused)) void Species::addLandauDamping() {
571   throw BoutException("Error: Landau damping is not yet
      implemented");
572 }
573
574 void Species::addDensitySource() {
575   TRACE("Species::addDensitySource");
576   if (evolve_N) {
577     ddt(logN) += source;
578   }
579   // Assume that the source ions and electrons have zero
      initial momentum
580   if (evolve_A_Upar) {
581     ddt(A_Upar_stag) -= mu * Upar_stag * source_stag/N_stag;
582   }
583 }
584
585 void Species::addHeatSource() {
586   TRACE("Species::addHeatSource");
```

```
587   if (evolve_Tpar) {
588     // This factor of two should be removed and input files
      should be
589     // adjusted
590     ddt(logTpar) += 2*heat_source_par;
591   }
592   if (evolve_Tper) {
593     ddt(logTper) += heat_source_perp;
594   }
595 }
596
597 void Species::addDiffusion() {
598   if (evolve_N) {
599     ddt(logN) += getNumericalDiffusion<0>(N);
600   }
601   if (evolve_A_Upar) {
602     ddt(A_Upar_stag) += mu*getNumericalDiffusion<1>(Upar_stag);
603   }
604
605   if (evolve_Tpar) {
606     ddt(logTpar) += getNumericalDiffusion<2>(Tpar);
607   }
608
609   if (evolve_Tper) {
610     ddt(logTper) += getNumericalDiffusion<3>(Tper);
611   }
612   if(evolve_Qpar){
613     ddt(Qpar_stag) += getNumericalDiffusion<4>(Qpar_stag);
614   }
615   if(evolve_A_Qper){
616     ddt(A_Qper_stag) += mu*getNumericalDiffusion<5>(Qper_stag);
617   }
618 }
619
620 FieldGroup Species::getFieldGroup() { return (comms); }
621
622 FieldGroup Species::getInterpolatedFieldGroup() { return (
      interpolatedComms); }
623
624 void Species::applyLimits(){
625   if(evolve_N){
```

```
626     limitAtLeast<BoutReal>(N, NMin);
627     limitAtLeast<BoutReal>(N_stag, NMin);
628   }
629   if(evolve_Tpar){
630     limitAtLeast<BoutReal>(Tpar, TparMin);
631     limitAtLeast<BoutReal>(Tpar_stag, TparMin);
632   }
633   if(evolve_Tper){
634     limitAtLeast<BoutReal>(Tper, TperMin);
635     limitAtLeast<BoutReal>(Tper_stag, TperMin);
636   }
637 }
638 
639 void Species::applyLogLimits(){
640   if(evolve_logs){
641     if(evolve_N){
642       limitAtLeast<BoutReal>(logN, log(NMin));
643     }
644     if(evolve_Tpar){
645       limitAtLeast<BoutReal>(logTpar, log(TparMin));
646     }
647     if(evolve_Tper){
648       limitAtLeast<BoutReal>(logTper, log(TperMin));
649     }
650   }
651 }
652 
653 void Species::setTimeDerivatives() {
654   if (calc_equilibrium) {
655     if (evolve_N) {
656       ddt(logN) = -Div_par(Upar_stag, CELL_CENTRE);
657     }
658     if (evolve_A_Upar) {
659       ddt(A_Upar_stag) = -Grad_par(phi1 + tau * Ppar, CELL_YLOW
    );
660     }
661 
662     if (evolve_Tpar) {
663       ddt(logTpar) = -Div_par(Upar_stag + Qpar_stag,
    CELL_CENTRE);
664       ddt(logTpar) *= 2.0; // Accounting for factor of 0.5 in
```

```
  equations
  }

  if (evolve_Tper) {
    ddt(logTper) = -Div_par(Qper_stag, CELL_CENTRE);
  }
  if (evolve_Qpar) {
    ddt(Qpar_stag) = -1.5 * tau * Grad_par(Tpar, CELL_YLOW);
    ddt(Qpar_stag) /= mu;
  }

  if (evolve_A_Qper) {
    ddt(A_Qper_stag) = -tau * Grad_par(Tper, CELL_YLOW);
  }
}
else {
  if (evolve_N) {
    ddt(logN) =
        -bracket(phi1, N, BRACKET_MODE, CELL_CENTRE) -
        bracket(phi2, Tper, BRACKET_MODE, CELL_CENTRE) -
        Div_par(Upar_stag, CELL_CENTRE) +
        Bxy * beta * bracket(apar1, Upar / Bxy, BRACKET_MODE,
  CELL_CENTRE) +
        beta * bracket(apar2, Qper, BRACKET_MODE, CELL_CENTRE
  ) +
        curvature(tau * 0.5 * (Ppar + Pper) + phi1 + 0.5 *
  phi2,
                  CELL_CENTRE);
  }

  if (evolve_A_Upar) {
    ddt(A_Upar_stag) =
        -mu * bracket(phi1_stag, Upar_stag, BRACKET_MODE,
  CELL_YLOW) -
        mu * bracket(phi2_stag, Qper_stag, BRACKET_MODE,
  CELL_YLOW) -
        Grad_par(phi1 + tau * Ppar, CELL_YLOW) +
        beta * bracket(apar1_stag, phi1_stag + tau *
  Ppar_stag,
                       BRACKET_MODE, CELL_YLOW) +
        beta * bracket(apar2_stag, phi2_stag + tau *
```

```
Tper_stag,
                        BRACKET_MODE, CELL_YLOW) -
        (phi2_stag + tau * (Tper_stag - Tpar_stag)) *
gradParNLlgB(CELL_YLOW) +
        mu * tau *
            curvature(2.0 * Upar_stag + Qpar_stag + 0.5 *
Qper_stag,
                      CELL_YLOW);
 }

 if (evolve_Tpar) {
   ddt(logTpar) =
        -0.5 * bracket(phi1, Tpar, BRACKET_MODE, CELL_CENTRE)
 -
        Div_par(Upar_stag + Qpar_stag, CELL_CENTRE) +
        Bxy * beta *  bracket(apar1, (Upar + Qpar) / Bxy,
                              BRACKET_MODE, CELL_CENTRE) +
        beta * bracket(apar2, Qper, BRACKET_MODE, CELL_CENTRE
) -
        (Upar + Qper) * gradParNLlgB(CELL_CENTRE) +
        curvature(0.5 * (tau * (Ppar + 2.0 * Tpar) + phi1),
CELL_CENTRE);

   ddt(logTpar) *= 2.0; // Accounting for factor of 0.5 in
equations
 }

 if (evolve_Tper) {
   ddt(logTper) =
        -bracket(phi1, Tper, BRACKET_MODE, CELL_CENTRE) -
        bracket(phi2, N + 2.0 * Tper, BRACKET_MODE,
CELL_CENTRE) -
        Div_par(Qper_stag, CELL_CENTRE) +
        Bxy * beta * bracket(apar1, Qper / Bxy, BRACKET_MODE,
 CELL_CENTRE) +
        beta * bracket(apar2, Upar + 2.0 * Qper, BRACKET_MODE
, CELL_CENTRE) +
        (Upar + Qper) * gradParNLlgB(CELL_CENTRE) +
        curvature(0.5 * (tau * (Pper + 3.0 * Tper) + (phi1 +
4.0 * phi2)),
                  CELL_CENTRE);
```

```
729     }
730
731     if (evolve_Qpar) {
732       ddt(Qpar_stag) =
733           -mu * bracket(phi1_stag, Qpar_stag, BRACKET_MODE,
    CELL_YLOW) -
734           1.5 * tau * (Grad_par(Tpar, CELL_YLOW) +
735           beta * bracket(apar1_stag, Tpar_stag, BRACKET_MODE,
    CELL_YLOW)) +
736           tau * mu *
737               curvature(0.5 * (3.0 * Upar_stag + 8.0 *
    Qpar_stag), CELL_YLOW);
738
739       ddt(Qpar_stag) /= mu;
740     }
741
742     if (evolve_A_Qper) {
743       ddt(A_Qper_stag) =
744           -mu * bracket(phi1_stag, Qper_stag, BRACKET_MODE,
    CELL_YLOW) -
745           mu * bracket(phi2_stag, Upar_stag + 2.0 * Qper_stag,
    BRACKET_MODE,
746                        CELL_YLOW) -
747           tau * (Grad_par(Tper, CELL_YLOW) +
748           beta * bracket(apar1_stag, Tper_stag, BRACKET_MODE,
    CELL_YLOW)) +
749           beta * bracket(apar2_stag, phi1_stag + tau *
    Ppar_stag,
750                          BRACKET_MODE, CELL_YLOW) +
751           beta * bracket(apar2_stag, 2.0 * (phi2_stag + tau *
    Tper_stag),
752                          BRACKET_MODE, CELL_YLOW) -
753           (phi2_stag + tau * Tper_stag - tau * Tpar_stag) *
    gradParNLlgB(CELL_YLOW) +
754           tau * mu * curvature(0.5 * (Upar_stag + 6.0 *
    Qper_stag), CELL_YLOW);
755     }
756   }
757
758   addDiffusion();
759
```

```
760   if (include_density_source){
761     addDensitySource();
762   }
763
764   if (include_heat_source){
765     addHeatSource();
766   }
767
768   if (include_coll_disp){
769     addCollisionalDissipation();
770   }
771
772   if(evolve_logs) {
773     if(evolve_N){
774       ddt(logN) /= N;
775     }
776     if(evolve_Tpar){
777       ddt(logTpar) /= Tpar;
778     }
779     if(evolve_Tper){
780       ddt(logTper) /= Tper;
781     }
782   }
783   else {
784     if(evolve_N)
785       ddt(N) = ddt(logN);
786     if(evolve_Tpar)
787       ddt(Tpar) = ddt(logTpar);
788     if(evolve_Tper)
789       ddt(Tper) = ddt(logTper);
790   }
791 }
```

B Additional 2D Filament simulations

In addition to the hot-electron, hot-ion 2D simulations described in section 3 simulations were also carried out with cold electrons and cold ions as well as hot electrons and cold ions. The results of these simulations are very similar to the hot-electron, hot-ion simulations but the results are included here for completeness.

B.1 Cold Ion Isolated filament simulations

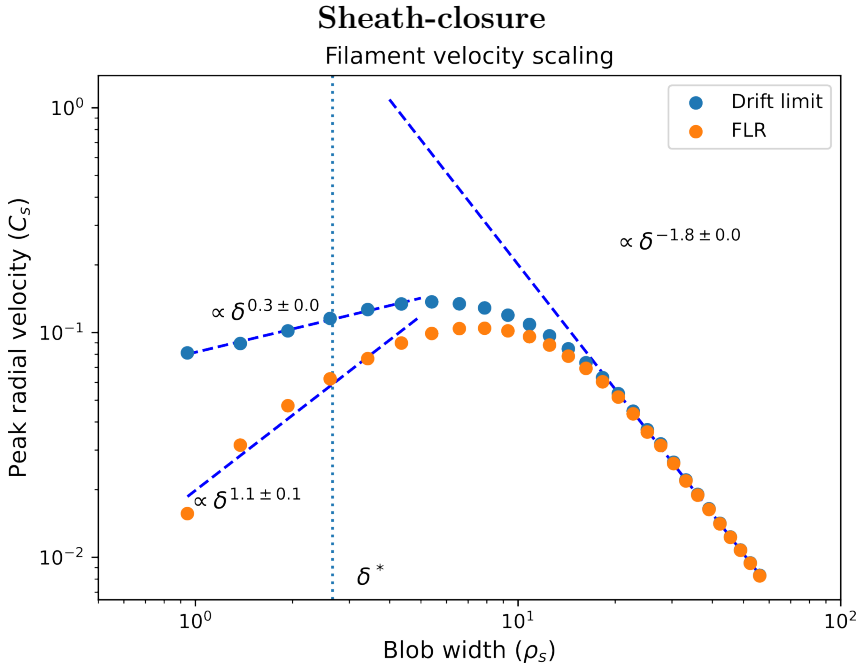

Figure 66: Velocity scaling relation for 2D sheath-closure simulations with cold ions. The inverse square velocity scaling relation is recovered for both drift-reduced and FLR simulations

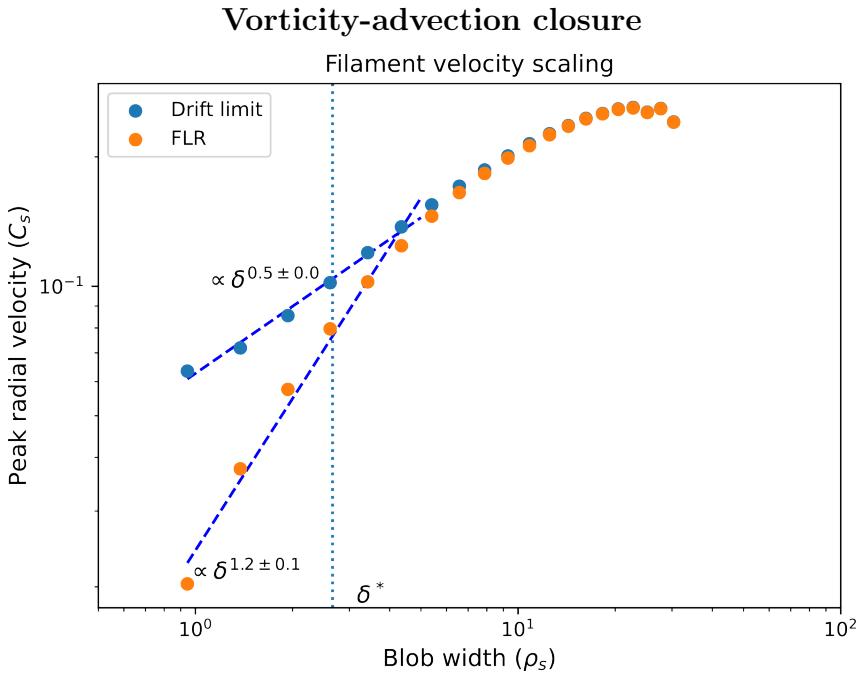

Figure 67: Velocity scaling relation for 2D sheath-closure simulations with cold ions. The square-root velocity scaling relation is recovered for the drift-reduced simulations, but there is a clear deviation for FLR simulations

B.2 Cold Ion Filament interactions

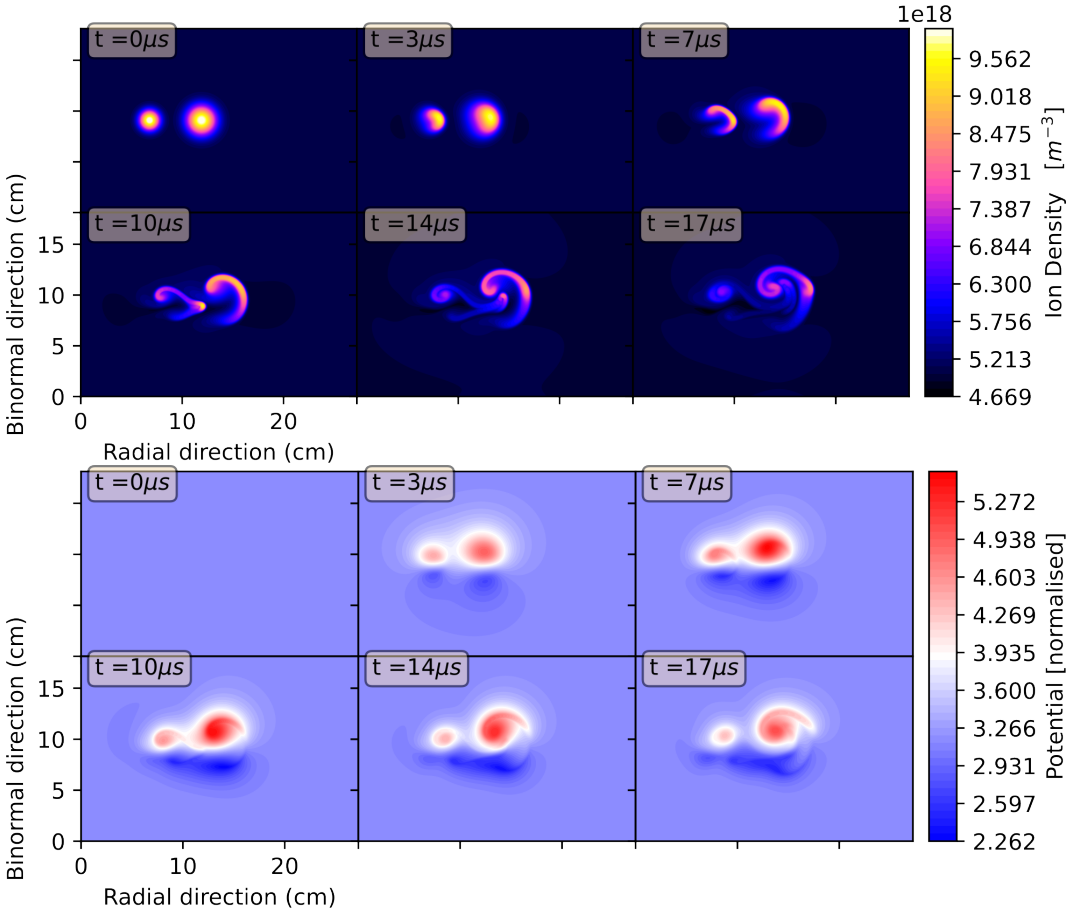

Figure 68: Radially displaced blobs interacting in the drift limit with parameters $(\epsilon, w_A, w_B) = (1.5, 5, 7.5)$

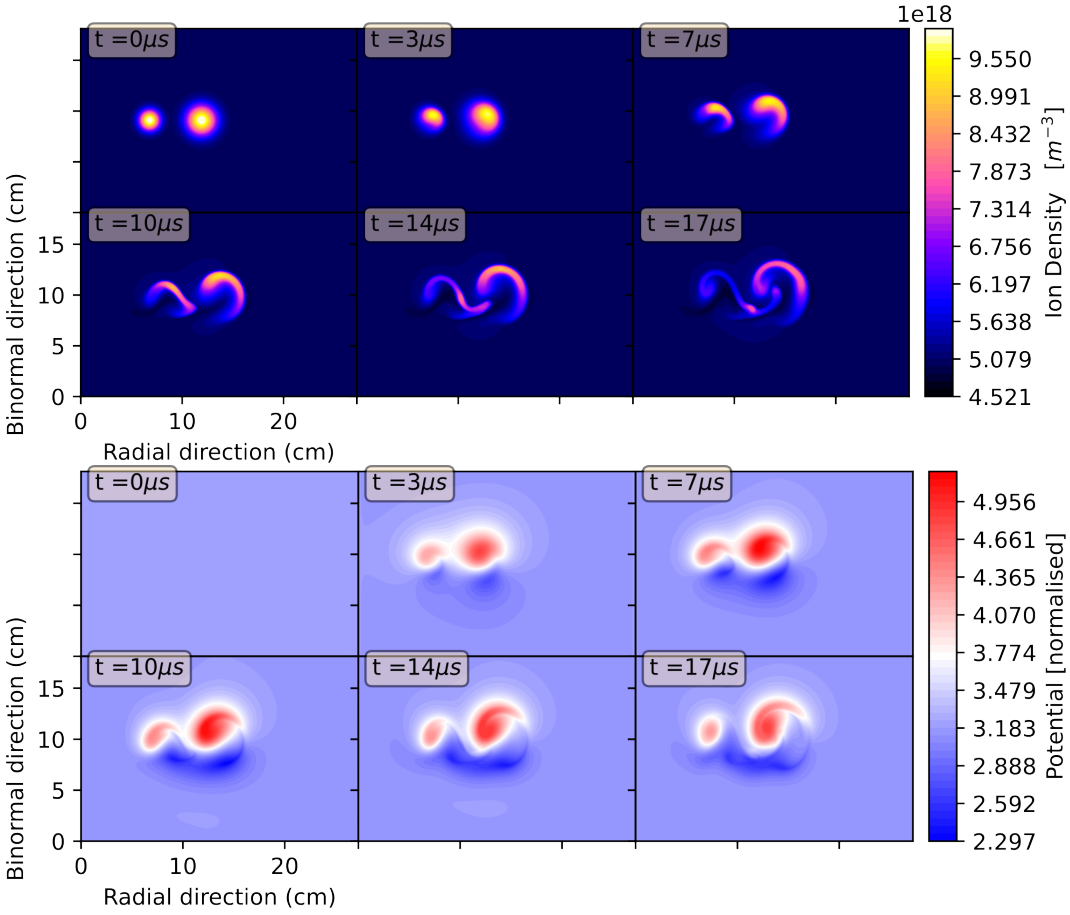

Figure 69: Radially displaced blobs interacting with FLR effects included $(\epsilon, w_A, w_B) = (1.5, 5, 7.5)$

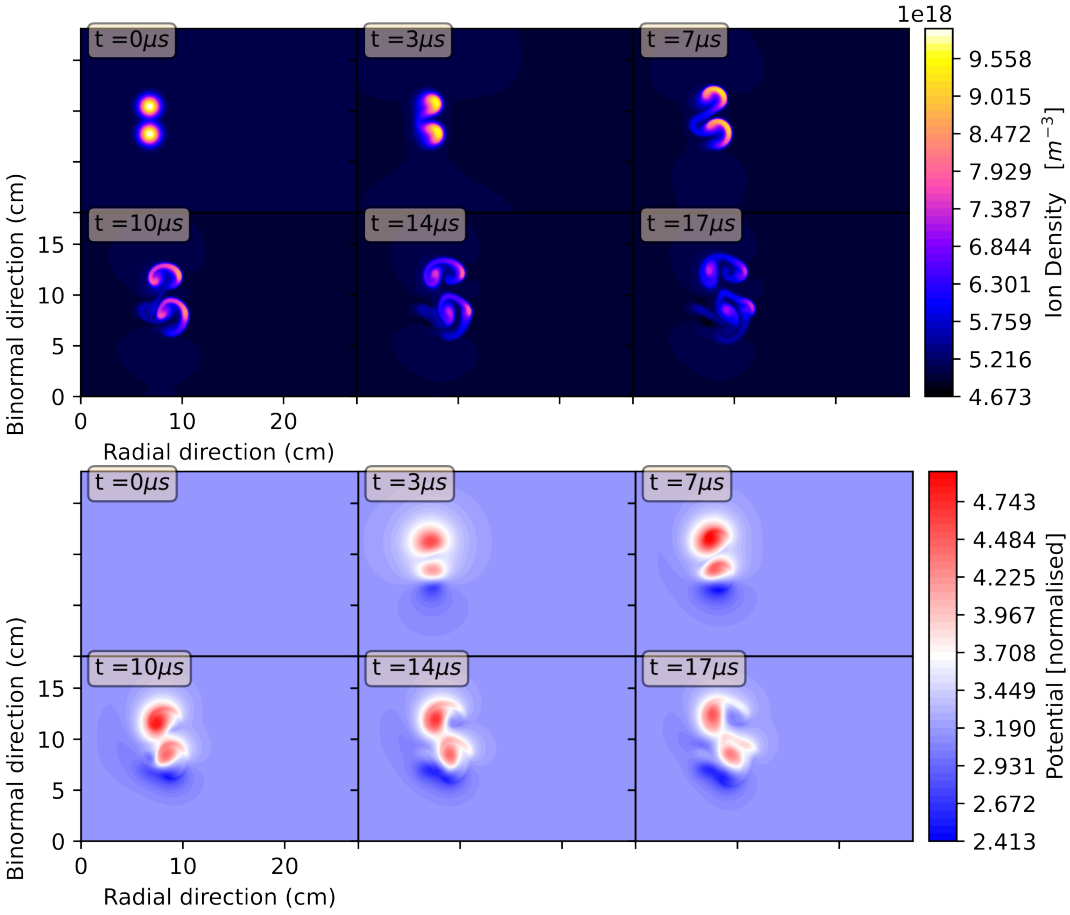

Figure 70: Poloidally displaced blobs interacting in the drift limit $(\epsilon, w_A, w_B) = (1.5, 5, 5)$

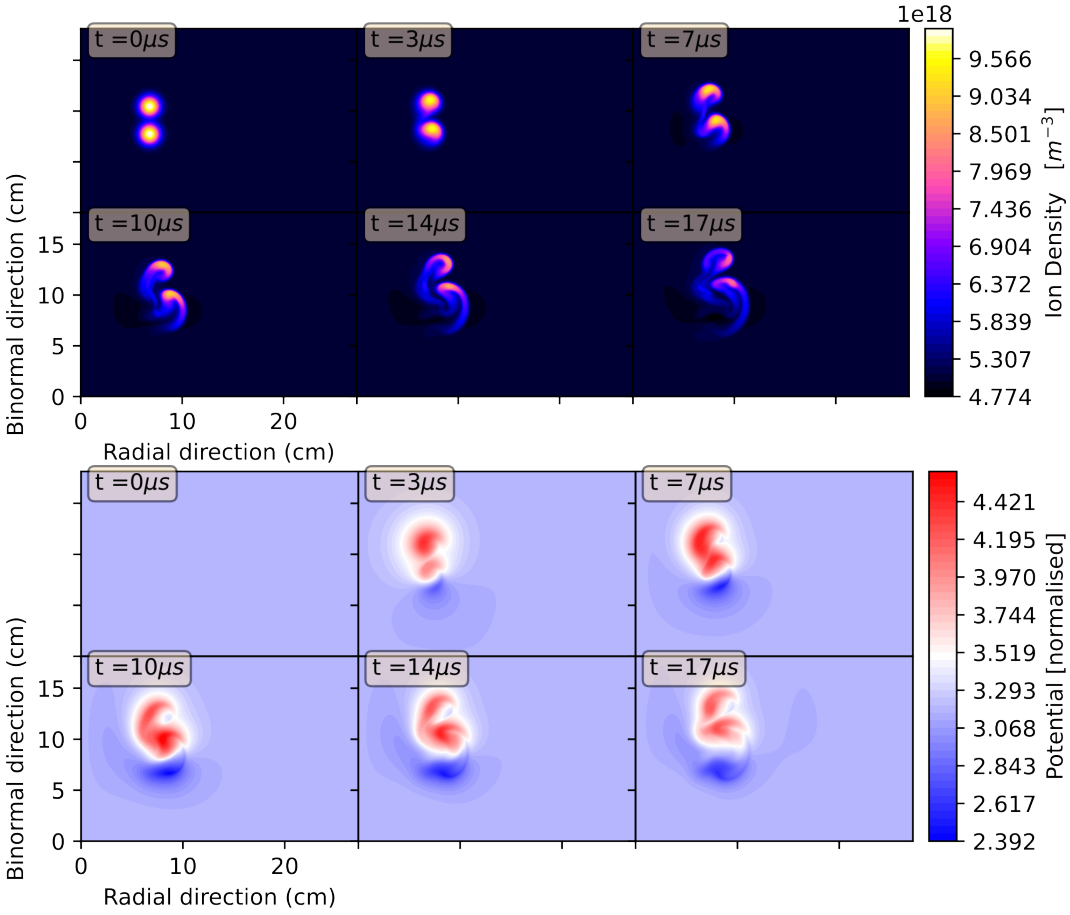

Figure 71: Poloidally displaced blobs interacting with FLR effects included $(\epsilon, w_A, w_B) = (1.5, 5, 5)$

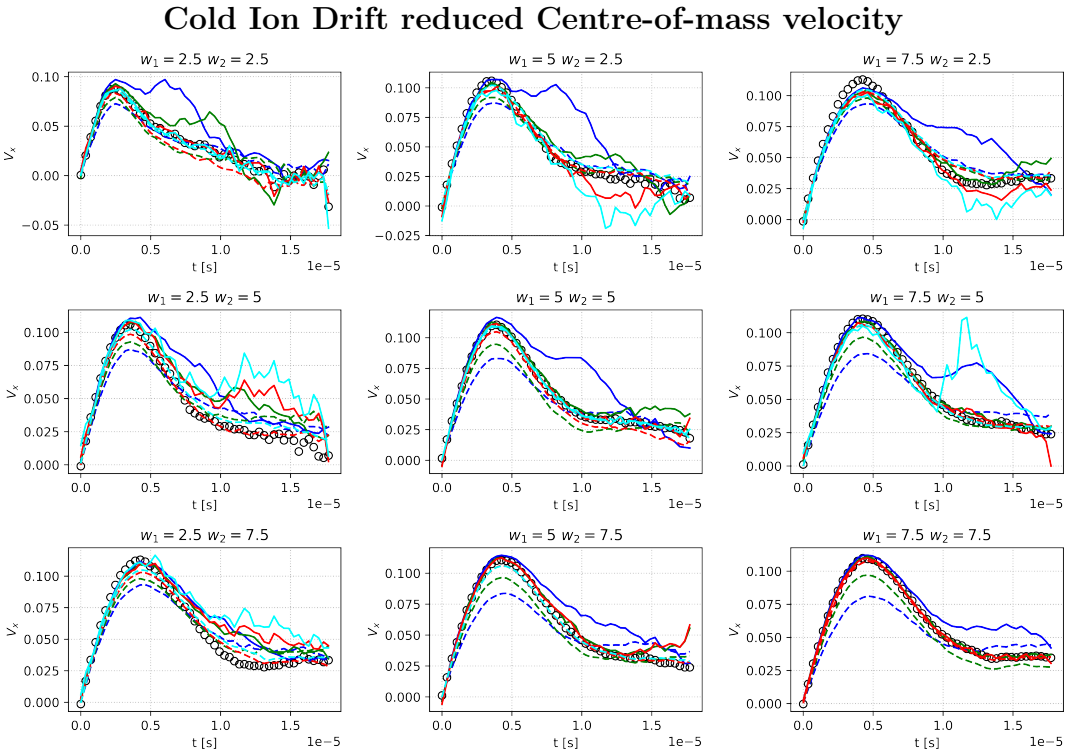

Figure 72: Each subfigure represents a combination of filaments of width w_1, w_2 given in the title. The solid lines represent radially separated filaments, dashed lines represent poloidally separated filaments. The black circles represent the non-interacting reference velocity. Blue, green, red and cyan curves represent ϵ equal to 1.5, 2, 3 and 5 respectively.

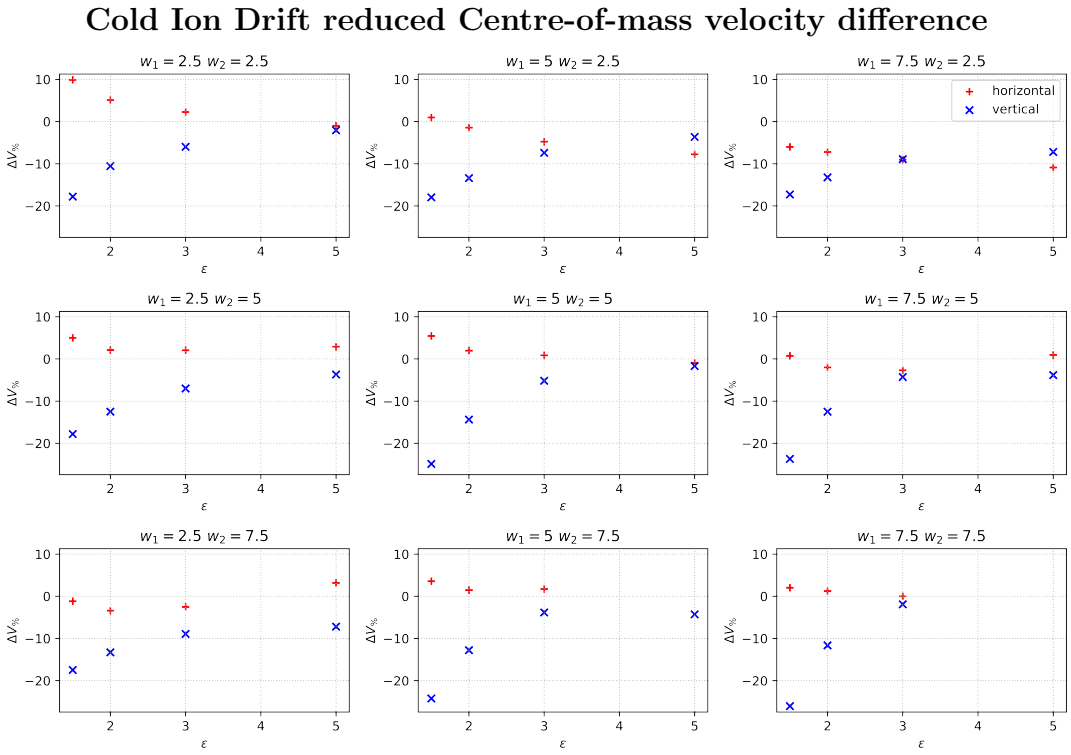

Figure 73: Fractional difference in peak centre-of-mass velocity as a function of separation distance. Blue crosses represent polloidally separated filaments while red pluses represent radially separated filaments.

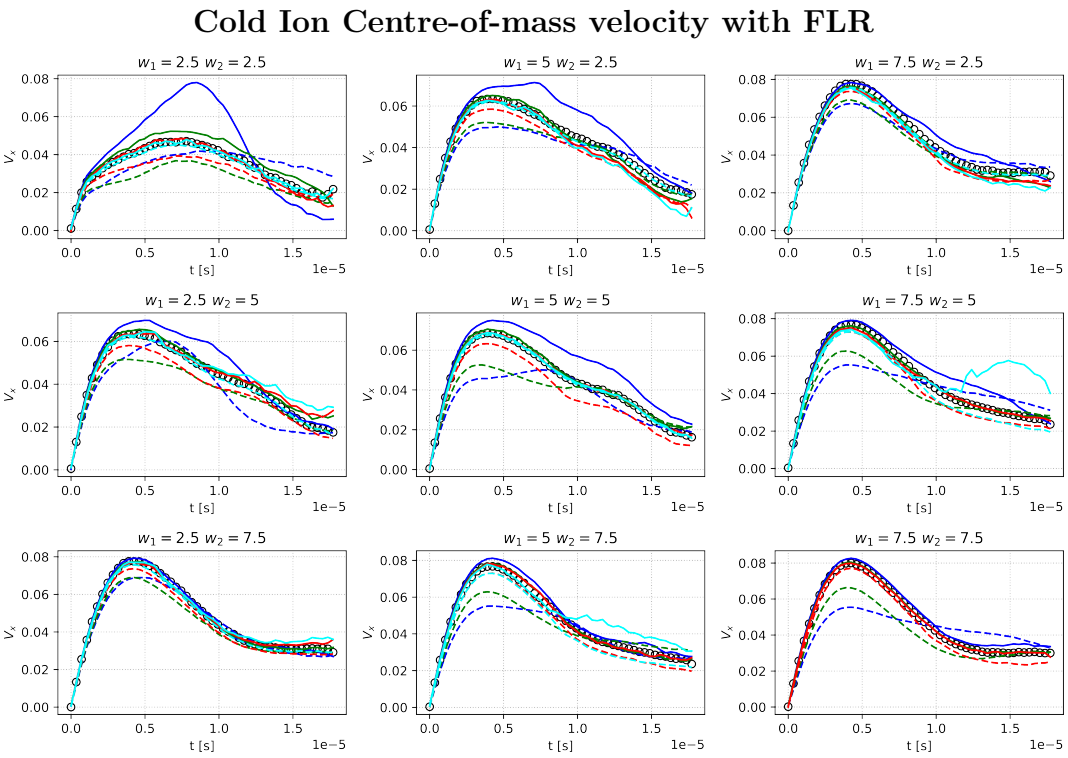

Figure 74: Each subfigure represents a combination of filaments of width w_1, w_2 given in the title. The solid lines represent radially separated filaments, dashed lines represent poloidally separated filaments. The black circles represent the non-interacting reference velocity. Blue, green, red and cyan curves represent ϵ equal to 1.5, 2, 3 and 5 respectively.

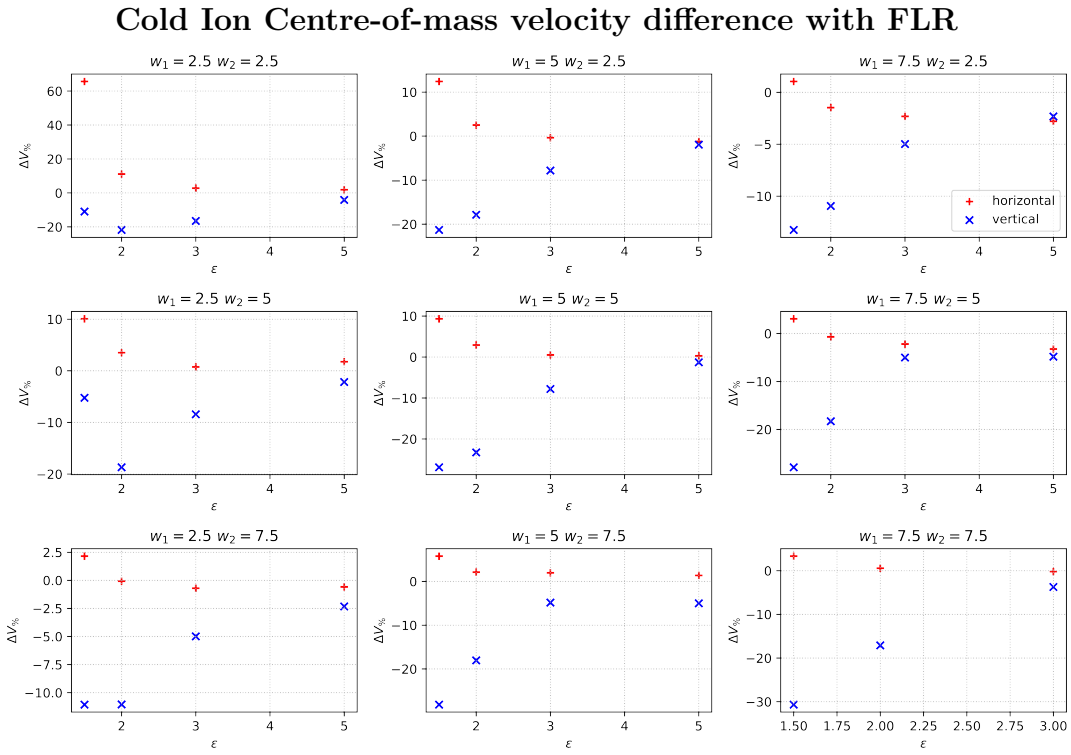

Figure 75: Fractional difference in peak centre-of-mass velocity as a function of separation distance. Blue crosses represent polloidally separated filaments while red pluses represent radially separated filaments.

B.3 Cold Ion, Cold Electron Isolated filament simulations

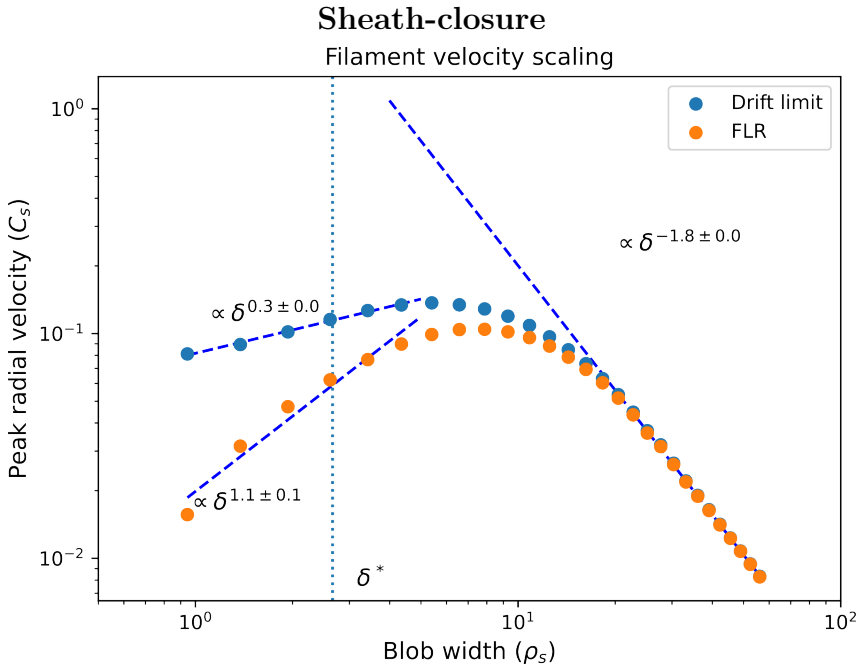

Figure 76: Velocity scaling relation for 2D sheath-closure simulations with cold ions. The inverse square velocity scaling relation is recovered for both drift-reduced and FLR simulations

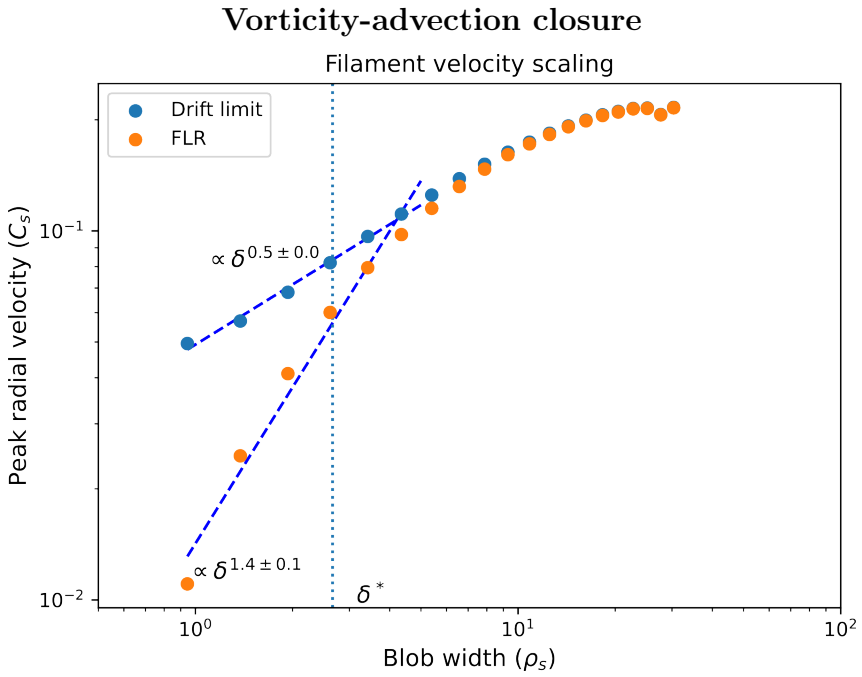

Figure 77: Velocity scaling relation for 2D sheath-closure simulations with cold ions. The square-root velocity scaling relation is recovered for the drift-reduced simulations, but there is a clear deviation for FLR simulations

B.4 Cold Ion, Cold Electron Filament interactions

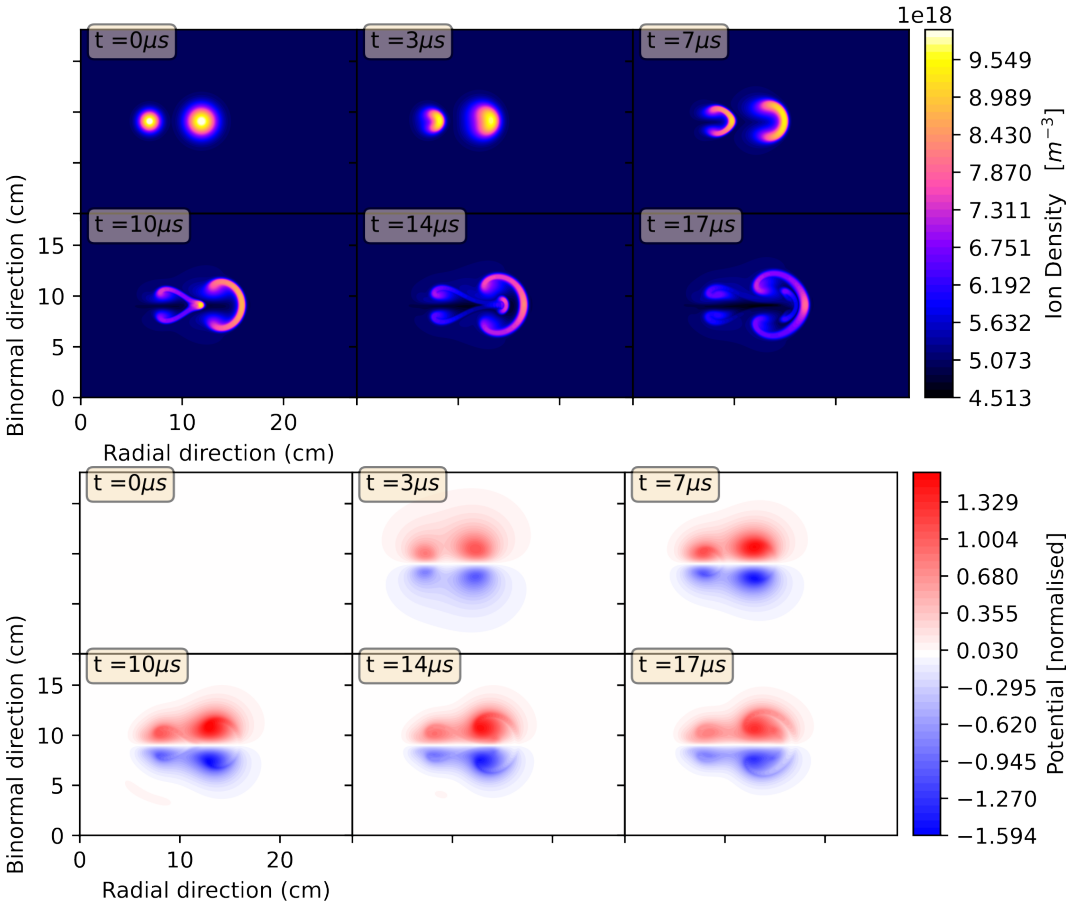

Figure 78: Radially displaced blobs interacting in the drift limit with parameters $(\epsilon, w_A, w_B) = (1.5, 5, 7.5)$

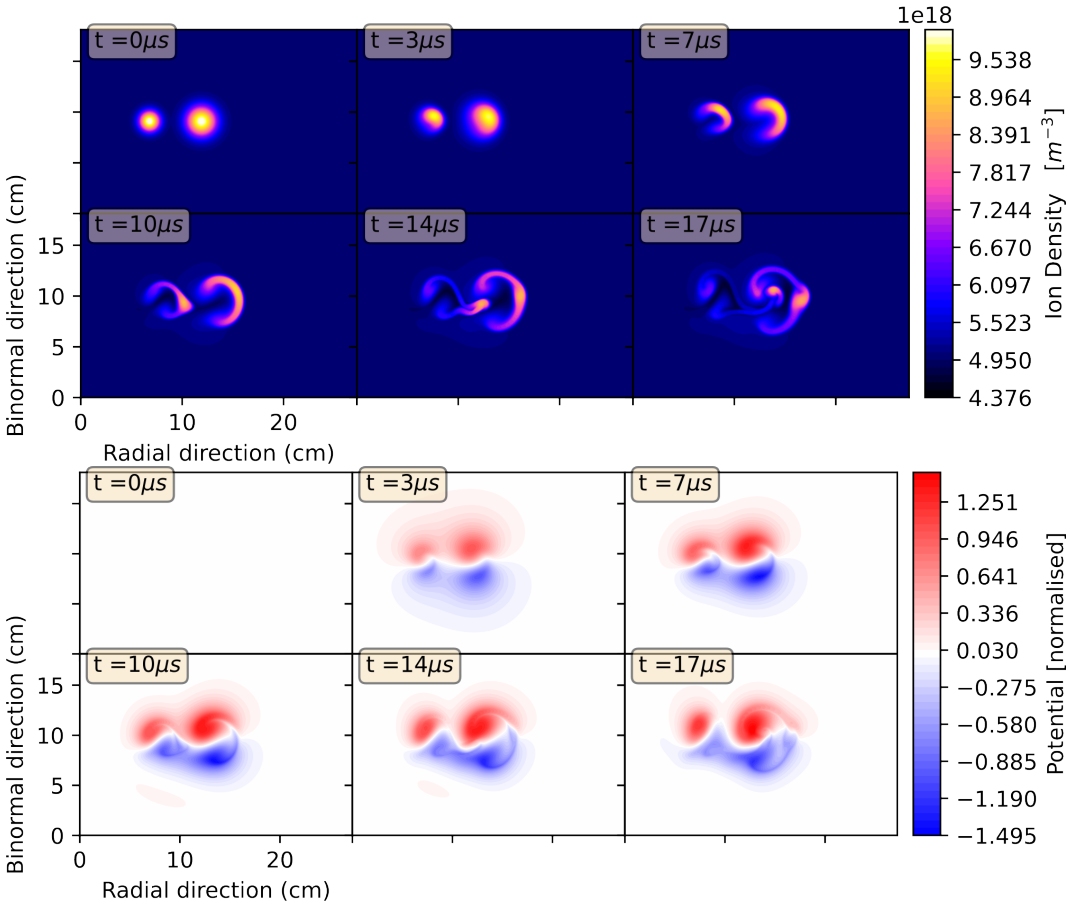

Figure 79: Radially displaced blobs interacting with FLR effects included $(\epsilon, w_A, w_B) = (1.5, 5, 7.5)$

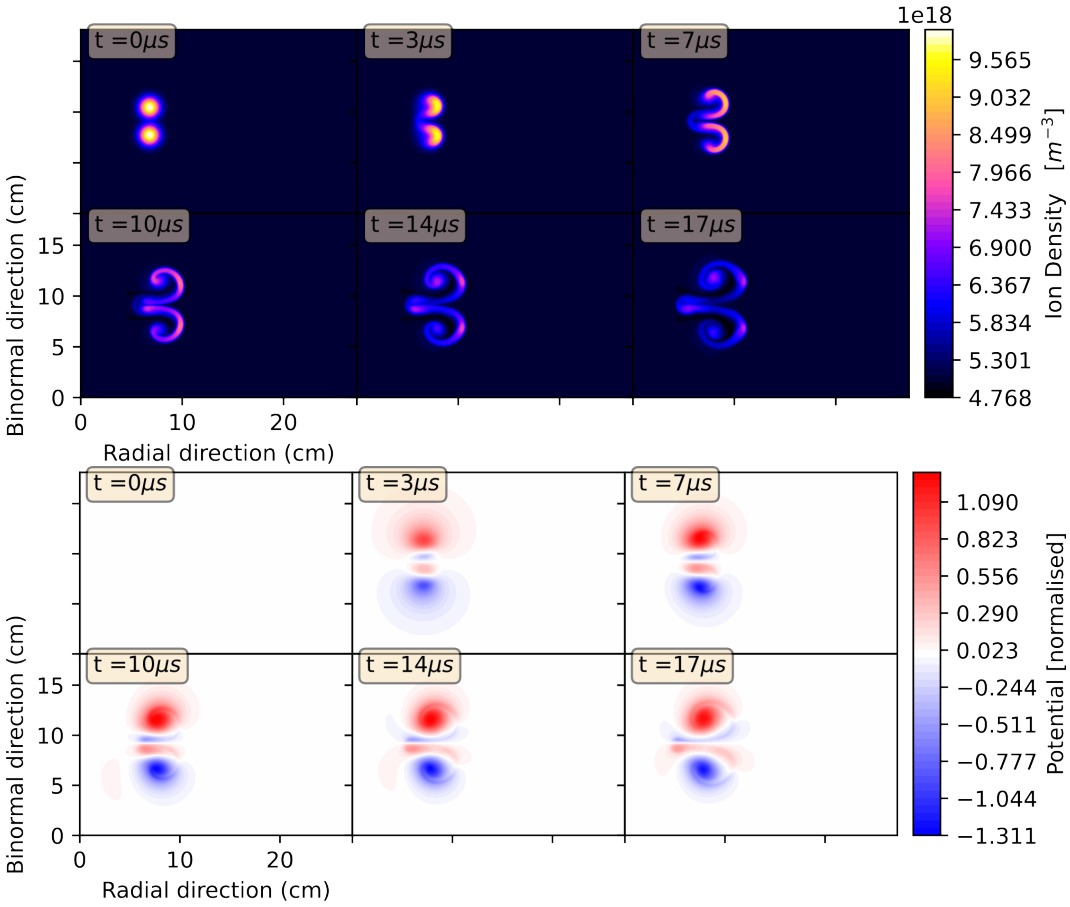

Figure 80: Poloidally displaced blobs interacting in the drift limit $(\epsilon, w_A, w_B) = (1.5, 5, 5)$

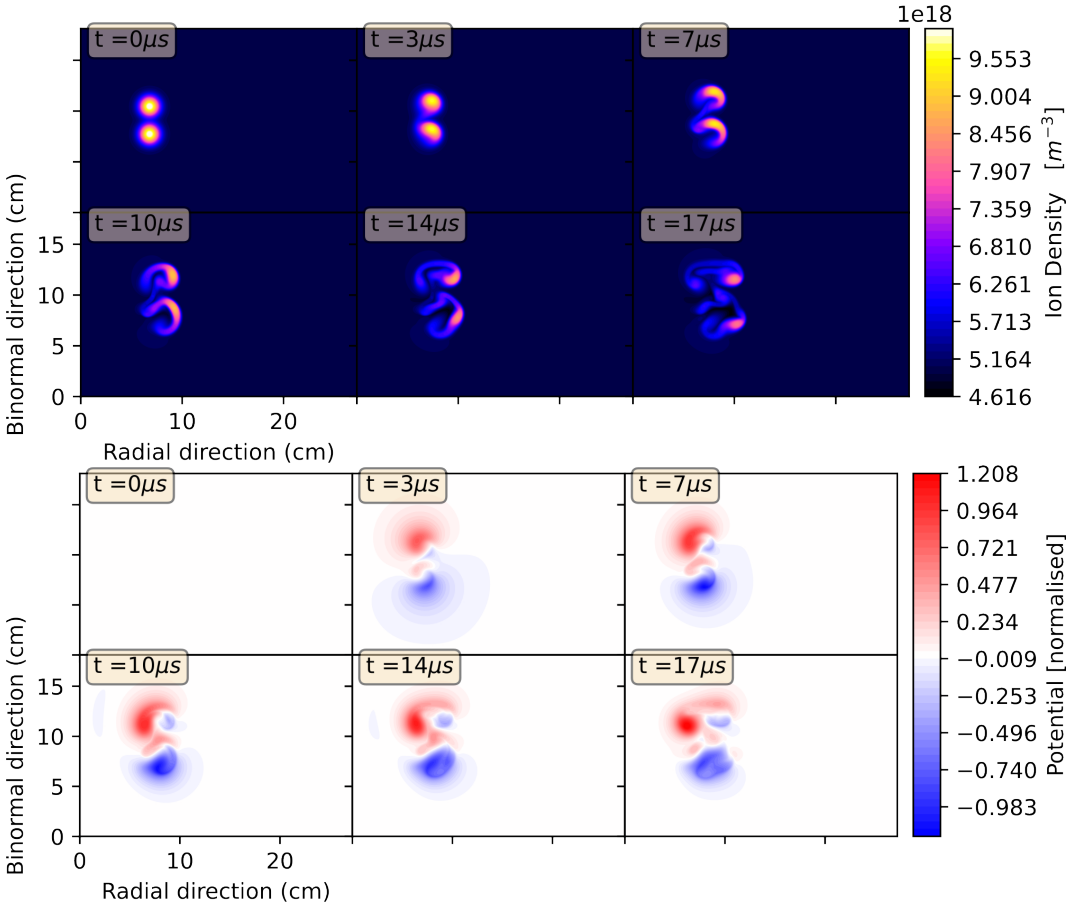

Figure 81: Poloidally displaced blobs interacting with FLR effects included $(\epsilon, w_A, w_B) = (1.5, 5, 5)$

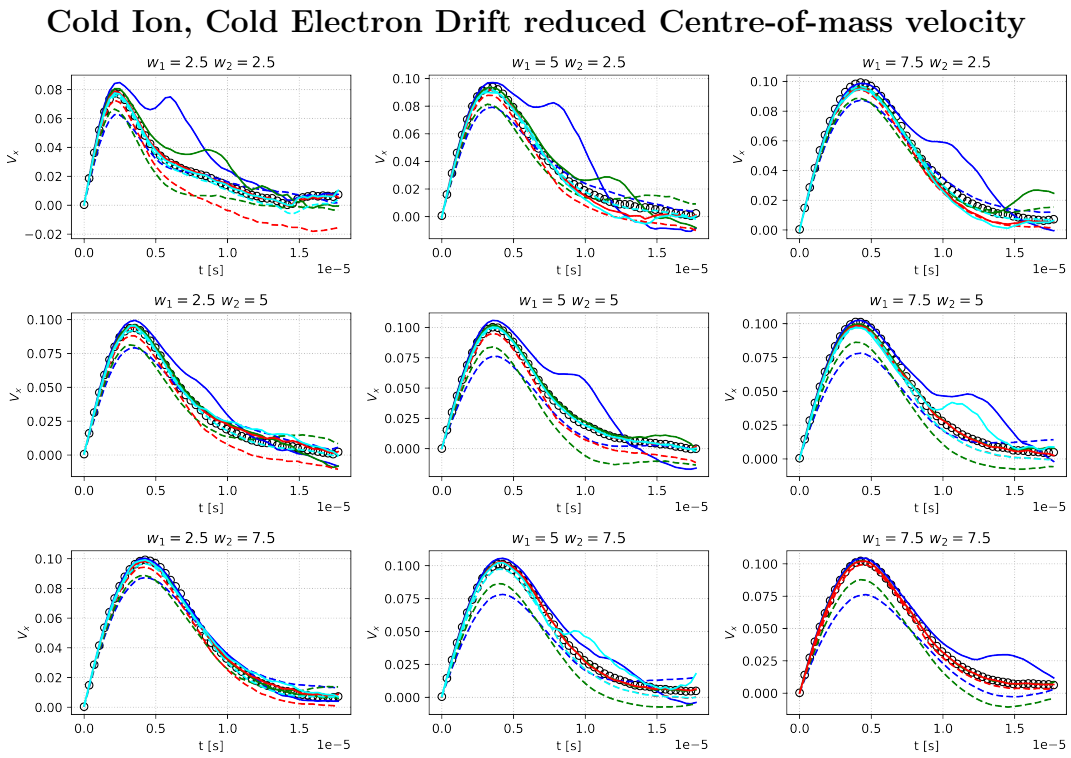

Figure 82: Each subfigure represents a combination of filaments of width w_1, w_2 given in the title. The solid lines represent radially separated filaments, dashed lines represent poloidally separated filaments. The black circles represent the non-interacting reference velocity. Blue, green, red and cyan curves represent ϵ equal to 1.5, 2, 3 and 5 respectively.

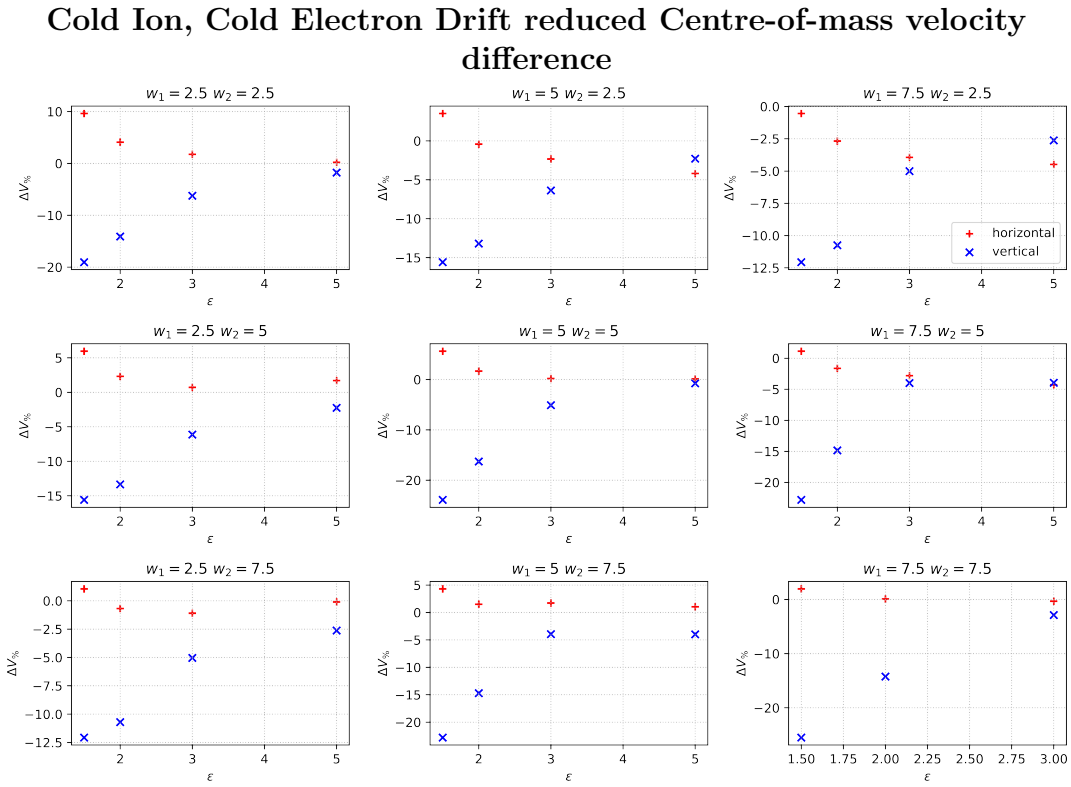

Figure 83: Fractional difference in peak centre-of-mass velocity as a function of separation distance. Blue crosses represent polloidally separated filaments while red pluses represent radially separated filaments.

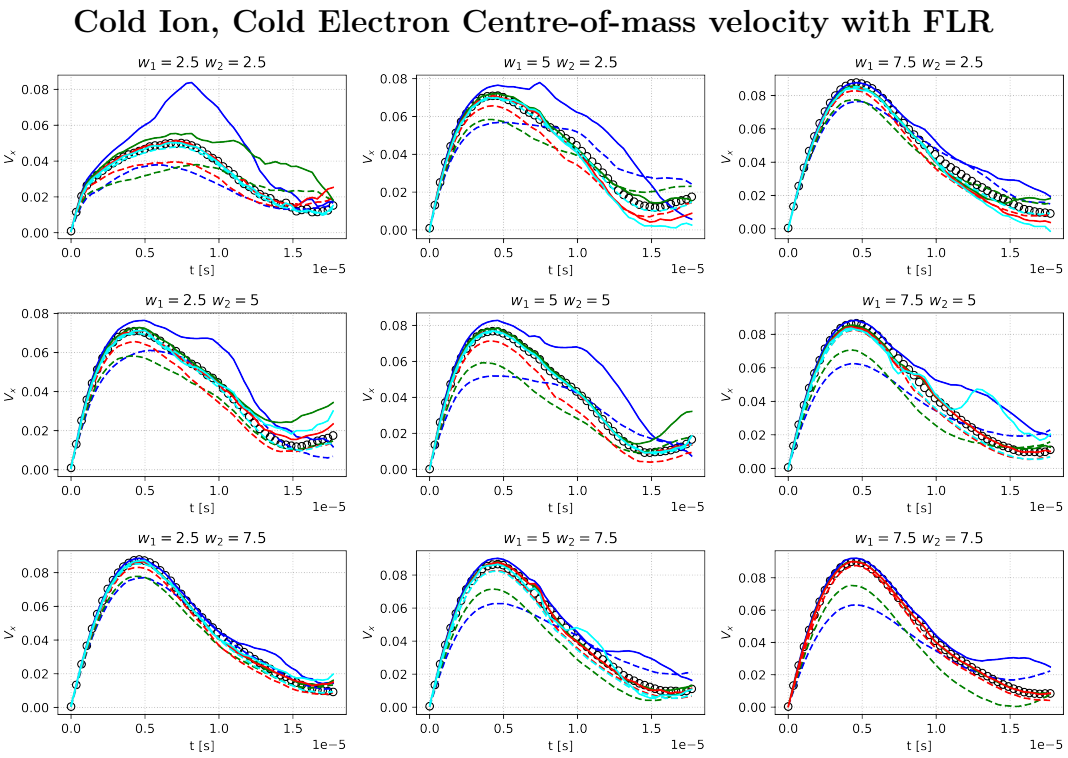

Figure 84: Each subfigure represents a combination of filaments of width w_1, w_2 given in the title. The solid lines represent radially separated filaments, dashed lines represent poloidally separated filaments. The black circles represent the non-interacting reference velocity. Blue, green, red and cyan curves represent ϵ equal to 1.5, 2, 3 and 5 respectively.

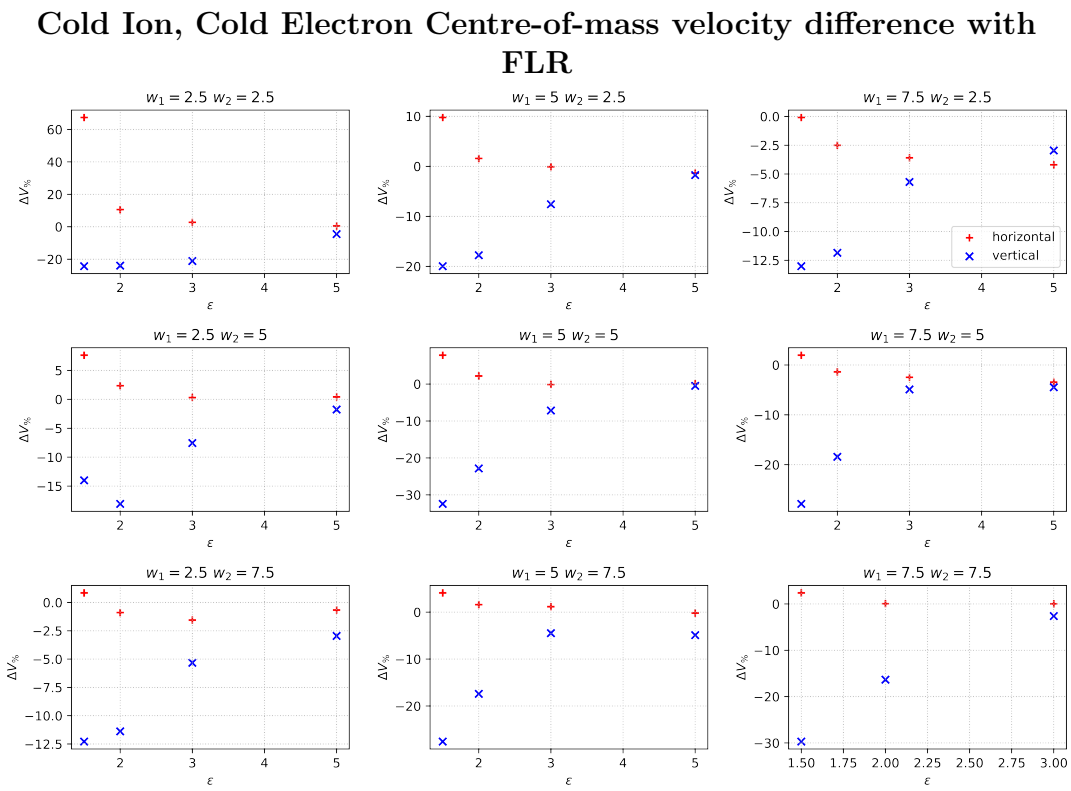

Figure 85: Fractional difference in peak centre-of-mass velocity as a function of separation distance. Blue crosses represent polloidally separated filaments while red pluses represent radially separated filaments.

References

[1] G. A. Jones and K. J. Warner, "The 21st century population-energy-climate nexus," *Energy Policy*, vol. 93, pp. 206–212, 6 2016.

[2] F. Wagner, "Physics of magnetic confinement fusion," *EPJ Web of Conferences*, vol. 54, 2013.

[3] S. Li, H. Jiang, Z. Ren, and C. Xu, "Optimal tracking for a divergent-type parabolic pde system in current profile control," *Abstract and Applied Analysis*, vol. 2014, pp. 1–8, 2014.

[4] S. I. Krasheninnikov, D. A. D'ippolito, and J. R. Myra, "Recent theoretical progress in understanding coherent structures in edge and sol turbulence," *Journal of Plasma Physics*, vol. 74, pp. 679–717, 10 2008.

[5] C. Theiler, I. Furno, P. Ricci, A. Fasoli, B. Labit, S. H. Müller, and G. Plyushchev, "Cross-field motion of plasma blobs in an open magnetic field line configuration," *Physical Review Letters*, vol. 103, p. 065001, 8 2009.

[6] P. Gerland, A. E. Raftery, H. Ševčíková, N. Li, D. Gu, T. Spoorenberg, L. Alkema, B. K. Fosdick, J. Chunn, N. Lalic, G. Bay, T. Buettner, G. K. Heilig, and J. Wilmoth, "World population stabilization unlikely this century," *Science*, vol. 346, pp. 234–237, 2014.

[7] F. Urban, R. M. J. Benders, and H. C. Moll, "Modelling energy systems for developing countries," *Energy Policy*, vol. 35, pp. 3473–3482, 6 2007.

[8] J. Rogelj, M. den Elzen, N. Höhne, T. Fransen, H. Fekete, H. Winkler, R. Schaeffer, F. Sha, K. Riahi, and M. Meinshausen, "Paris agreement climate proposals need a boost to keep warming well below 2 c," *Nature*, vol. 534, pp. 631–639, 6 2016.

[9] M. Sigmond, J. C. Fyfe, and N. C. Swart, "Ice-free arctic projections under the paris agreement," *Nature Climate Change*, vol. 8, pp. 404–408, 5 2018.

[10] S. C. Doney, V. J. Fabry, R. A. Feely, and J. A. Kleypas, "Ocean acidification: The other co 2 problem," *Annual Review of Marine Science*, vol. 1, pp. 169–192, 1 2008.

[11] J. Memmott, P. G. Craze, N. M. Waser, and M. V. Price, "Global warming and the disruption of plant-pollinator interactions," *Ecology Letters*, vol. 10, pp. 710–717, 8 2007.

[12] S. M. Vicente-Serrano, S. Beguería, and J. I. López-Moreno, "A multiscalar drought index sensitive to global warming: The standardized precipitation evapotranspiration index," *Journal of Climate*, vol. 23, pp. 1696–1718, 4 2010.

[13] M. A. Delucchi and M. Z. Jacobson, "Meeting the world's energy needs entirely with wind, water, and solar power," *Bulletin of the Atomic Scientists*, vol. 69, pp. 30–40, 7 2013.

[14] L. Spitzer, "The stellarator concept," *The Physics of Fluids*, vol. 1, pp. 253–264, 11 1958.

[15] J. Wesson and D. J. Campbell, *Tokamaks*. Oxford University Press, 2011.

[16] A. Guthrie and R. K. Wakerling, eds., *The Characteristics Of Electrical Discharges In Magnetic Fields*. McGraw Hill, first edition ed., 1949.

[17] D. Bohm, *Qualitative Description Of The Arc Plasma In A Magnetic Field*. 1949.

[18] F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*. SPRINGER, 3 ed., 2016.

[19] D. H. J. H. J. Goodall, "High speed cine film studies of plasma behaviour and plasma surface interactions in tokamaks," *Journal of Nuclear Materials*, vol. 111-112, pp. 11–22, 11 1982.

[20] S. J. Zweben, "Search for coherent structure within tokamak plasma turbulence," *Physics of Fluids*, vol. 28, pp. 974–982, 6 1985.

[21] M. Endler, H. Niedermeyer, L. Giannone, E. Kolzhauer, A. Rudyj, G. Theimer, and N. Tsois, "Measurements and modelling of electrostatic

fluctuations in the scrape-off layer of asdex," *Nuclear Fusion*, vol. 35, pp. 1307–1339, 11 1995.

[22] C. K. Tsui, J. A. Boedo, J. R. Myra, B. Duval, B. Labit, C. Theiler, N. Vianello, W. A. J. Vijvers, H. Reimerdes, S. Coda, O. Février, J. R. Harrison, J. Horacek, B. Lipschultz, R. Maurizio, F. Nespoli, U. Sheikh, K. Verhaegh, and N. Walkden, "Filamentary velocity scaling validation in the tcv tokamak," *Physics of Plasmas*, vol. 25, p. 072506, 7 2018.

[23] A. Kirk, N. B. Ayed, G. Counsell, B. Dudson, T. Eich, A. Herrmann, B. Koch, R. Martin, A. Meakins, S. Saarelma, R. Scannell, S. Tallents, M. Walsh, H. R. Wilson, N. B. Ayed, G. Counsell, B. Dudson, T. Eich, A. Herrmann, B. Koch, R. Martin, A. Meakins, S. Saarelma, R. Scannell, S. Tallents, M. Walsh, and H. R. Wilson, "Filament structures at the plasma edge on mast," *Plasma Physics and Controlled Fusion*, vol. 48, pp. B433–B441, 12 2006.

[24] N. B. Ayed, A. Kirk, B. Dudson, S. Tallents, R. G. L. Vann, H. R. Wilson, N. B. Ayed, A. Kirk, B. Dudson, S. Tallents, R. G. L. Vann, H. R. Wilson, N. B. Ayed, A. Kirk, B. Dudson, S. Tallents, R. G. L. Vann, and H. R. Wilson, "Inter-elm filaments and turbulent transport in the mega-amp spherical tokamak," *Plasma Physics and Controlled Fusion*, vol. 51, p. 35016, 3 2009.

[25] R. Kube, O. E. Garcia, B. LaBombard, J. L. Terry, and S. J. Zweben, "Blob sizes and velocities in the alcator c-mod scrape-off layer," *Journal of Nuclear Materials*, vol. 438, pp. S505–S508, 7 2013.

[26] J. A. Boedo, P. C. Stangeby, G. R. McKee, G. D. Porter, W. P. West, R. A. Moyer, S. Krasheninnikov, J. G. Watkins, M. A. Mahdavi, D. L. Rudakov, A. W. Leonard, G. Antar, and D. G. Whyte, "Fluctuation-driven transport in the diii-d boundary," *Plasma Physics and Controlled Fusion*, vol. 44, pp. 717–731, 6 2002.

[27] D. A. D'Ippolito, J. R. Myra, and S. I. Krasheninnikov, "Cross-field blob transport in tokamak scrape-off-layer plasmas," *Physics of Plasmas*, vol. 9, pp. 222–233, 1 2002.

[28] S. I. Krasheninnikov, "On scrape off layer plasma transport," *Physics Letters, Section A: General, Atomic and Solid State Physics*, vol. 283, pp. 368–370, 5 2001.

[29] B. Lipschultz, X. Bonnin, G. Counsell, a. Kallenbach, a. Kukushkin, K. Krieger, a. Leonard, a. Loarte, R. Neu, R. Pitts, T. Rognlien, J. Roth, C. Skinner, J. L. Terry, E. Tsitrone, D. Whyte, S. Zweben, N. Asakura, D. Coster, R. Doerner, R. Dux, G. Federici, M. Fenstermacher, W. Fundamenski, P. Ghendrih, a. Herrmann, J. Hu, S. Krasheninnikov, G. Kirnev, a. Kreter, V. Kurnaev, B. LaBombard, S. Lisgo, T. Nakano, N. Ohno, H. D. Pacher, J. Paley, Y. Pan, G. Pautasso, V. Philipps, V. Rohde, D. Rudakov, P. Stangeby, S. Takamura, T. Tanabe, Y. Yang, S. Zhu, a. Kallenbach, a. Kukushkin, K. Krieger, a. Leonard, a. Loarte, R. Neu, R. Pitts, T. Rognlien, J. Roth, C. Skinner, J. L. Terry, E. Tsitrone, D. Whyte, S. Zweben, N. Asakura, D. Coster, R. Doerner, R. Dux, G. Federici, M. Fenstermacher, W. Fundamenski, P. Ghendrih, a. Herrmann, J. Hu, S. Krasheninnikov, G. Kirnev, a. Kreter, V. Kurnaev, B. LaBombard, S. Lisgo, T. Nakano, N. Ohno, H. D. Pacher, J. Paley, Y. Pan, G. Pautasso, V. Philipps, V. Rohde, D. Rudakov, P. Stangeby, S. Takamura, T. Tanabe, Y. Yang, and S. Zhu, "Plasma–surface interaction, scrape-off layer and divertor physics: implications for iter," *Nuclear Fusion*, vol. 47, pp. 1189–1205, 2007.

[30] J. Cheng, L. W. Yan, J. Q. Dong, W. Y. Hong, L. H. Yao, Z. H. Huang, K. J. Zhao, J. M. Gao, C. Y. Chen, B. B. Feng, L. Nie, W. W. Xiao, Q. W. Yang, X. T. Ding, and X. R. Duan, "Divertor heat flux mitigation using supersonic molecular beam injection in type-iii elmy h mode plasmas of hl-2a tokamak," *Journal of Nuclear Materials*, vol. 438, p. 11075046, 2013.

[31] D. A. D'Ippolito, J. R. Myra, and S. J. Zweben, "Convective transport by intermittent blob-filaments: Comparison of theory and experiment," *Physics of Plasmas*, vol. 18, p. 060501, 6 2011.

[32] N. Bisai, A. Das, S. Deshpande, R. Jha, P. Kaw, A. Sen, and R. Singh, "Formation of a density blob and its dynamics in the edge and the scrape-off layer of a tokamak plasma," *Physics of Plasmas*, vol. 12, pp. 1–6, 10 2005.

[33] T. Windisch, O. Grulke, and T. Klinger, "Radial propagation of structures in drift wave turbulence," *Physics of Plasmas*, vol. 13, p. 122303, 12 2006.

[34] G. Fuchert, G. Birkenmeier, B. Nold, M. Ramisch, and U. Stroth, "The influence of plasma edge dynamics on blob properties in the stellarator tj-k," *Plasma Physics and Controlled Fusion*, vol. 55, p. 125002, 12 2013.

[35] F. Greiner, M. Ramisch, N. Mahdizadeh, T. Happel, U. Stroth, and B. Nold, "Generation of intermittent turbulent events at the transition from closed to open field lines in a toroidal plasma," *Physical Review Letters*, vol. 102, p. 255001, 6 2009.

[36] P. Manz, M. Xu, S. H. Müller, N. Fedorczak, S. C. Thakur, J. H. Yu, and G. R. Tynan, "Plasma blob generation due to cooperative elliptic instability," *Physical Review Letters*, vol. 107, p. 195004, 11 2011.

[37] A. V. Nedospasov, "The enhancement of edge turbulence in tokamaks by a limiter current," *Physics of Fluids B*, vol. 5, pp. 3191–3194, 9 1993.

[38] J. R. Myra, D. A. D'Ippolito, X. Q. Xu, and R. H. Cohen, "Resistive x-point modes in tokamak boundary plasmas," *Physics of Plasmas*, vol. 7, pp. 2290–2293, 6 2000.

[39] B. Zhu, M. Francisquez, and B. N. Rogers, "Gdb: A global 3d two-fluid model of plasma turbulence and transport in the tokamak edge," *Computer Physics Communications*, vol. 232, pp. 46–58, 11 2018.

[40] B. D. Dudson and J. Leddy, "Hermes: global plasma edge fluid turbulence simulations," *Plasma Physics and Controlled Fusion*, vol. 59, p. 054010, 5 2017.

[41] P. Manz, T. T. Ribeiro, B. D. Scott, G. Birkenmeier, D. Carralero, G. Fuchert, S. H. Müller, H. W. Müller, U. Stroth, and E. Wolfrum, "Origin and turbulence spreading of plasma blobs," *Physics of Plasmas*, vol. 22, p. 022308, 2 2015.

[42] P. W. Gingell, S. C. Chapman, and R. O. Dendy, "Plasma blob formation by ion kinetic kelvin-helmholtz and interchange instabilities," *Plasma Physics and Controlled Fusion*, vol. 56, p. 035012, 3 2014.

[43] F. Militello, B. Dudson, L. Easy, A. Kirk, and P. Naylor, "On the interaction of scrape off layer filaments," *Plasma Physics and Controlled Fusion*, vol. 59, p. 125013, 12 2017.

[44] D. Schwörer, N. R. R. Walkden, H. Leggate, B. D. D. Dudson, F. Militello, T. Downes, and M. M. M. Turner, "Influence of plasma background including neutrals on scrape-off layer filaments using 3d simulations," *Nuclear Materials and Energy*, 3 2017.

[45] L. Easy, F. Militello, J. Omotani, B. Dudson, E. Havlíčková, P. Tamain, V. Naulin, and A. H. Nielsen, "Three dimensional simulations of plasma filaments in the scrape off layer: A comparison with models of reduced dimensionality," *Physics of Plasmas*, vol. 21, p. 122515, 12 2014.

[46] W. Lee, J. R. Angus, M. V. Umansky, and S. I. Krasheninnikov, "Electromagnetic effects on plasma blob-filament transport," *Journal of Nuclear Materials*, vol. 463, pp. 765–768, 8 2015.

[47] O. E. Garcia, N. H. Bian, and W. Fundamenski, "Radial interchange motions of plasma filaments," *Physics of Plasmas*, vol. 13, p. 082309, 8 2006.

[48] N. R. Walkden, L. Easy, F. Militello, and J. T. Omotani, "Dynamics of 3d isolated thermal filaments," *Plasma Physics and Controlled Fusion*, vol. 58, p. 115010, 10 2016.

[49] S. I. Krasheninnikov, A. Y. Pigarov, S. A. Galkin, G. Q. Yu, D. A. D'Ippolito, J. R. Myra, D. R. Mccarthy, W. M. Nevins, T. D. Rognlien, X. Q. Xu, J. A. Boedo, D. L. Rudakov, M. J. Schaffer, W. P. West, and D. G. Whyte, "Blobs and cross-field transport in the tokamak edge plasmas," in *19th Fusion Energy Conference 14 - 19 October 2002 Lyon, France*, pp. IAEA–CN–94/TH/4–1, 2003.

[50] J. R. Myra, D. A. Russell, D. A. D'Ippolito, and D. A. D'Ippolito, "Collisionality and magnetic geometry effects on tokamak edge turbulent transport. i. a two-region model with application to blobs," *Physics of Plasmas*, vol. 13, p. 112502, 11 2006.

[51] J. C. Maxwell, "Xxv. on physical lines of force," *Philosophical Magazine Series 4*, vol. 21, pp. 161–175, 1861.

[52] S. I. Braginskii and M. A. Leontovich, "Reviews of plasma physics," 1965.

[53] A. J. Brizard and T. S. Hahm, "Foundations of nonlinear gyrokinetic theory," *Reviews of Modern Physics*, vol. 79, pp. 421–468, 4 2007.

[54] J. J. Ramos, "General expression of the gyroviscous force," *Physics of Plasmas*, vol. 12, pp. 1–7, 2005.

[55] M. A. Beer and G. W. Hammett, "Toroidal gyrofluid equations for simulations of tokamak turbulence," *Physics of Plasmas*, vol. 3, pp. 4046–4064, 1996.

[56] G. W. Hammett and F. W. Perkins, "Fluid moment models for landau damping with application to the ion-temperature-gradient instability," *Physical Review Letters*, vol. 64, pp. 3019–3022, 6 1990.

[57] G. W. Hammett, W. Dorland, and F. W. Perkins, "Fluid models of phase mixing, landau damping, and nonlinear gyrokinetic dynamics," *Physics of Fluids B*, vol. 4, pp. 2052–2061, 1992.

[58] B. D. Scott, "Free-energy conservation in local gyrofluid models," *Physics of Plasmas*, vol. 12, pp. 1–18, 10 2005.

[59] B. Scott, "Derivation via free energy conservation constraints of gyrofluid equations with finite-gyroradius electromagnetic nonlinearities," *Physics of Plasmas*, vol. 17, 2010.

[60] G. Knorr, F. R. Hansen, J. P. Lynov, H. L. Pécseli, and J. J. Rasmussen, "Finite larmor radius effects to arbitrary order," *Physica Scripta*, vol. 38, pp. 829–834, 1988.

[61] F. R. Hansen, G. Knorr, J. P. Lynov, H. L. Pecseli, and J. J. Rasmussen, "A numerical plasma simulation including finite larmor radius effects to arbitrary order," *Plasma Physics and Controlled Fusion*, vol. 31, pp. 173–183, 2 1989.

[62] S. Hamaguchi and W. Horton, "Ion temperature gradient driven turbulence in the weak density gradient limit," *Physics of Fluids B*, vol. 2, pp. 3040–3046, 12 1990.

[63] W. Dorland and G. W. Hammett, "Gyrofluid turbulence models with kinetic effects," *Physics of Fluids B*, vol. 5, pp. 812–835, 3 1993.

[64] H. G. W, M. A. Beer, W. Dorland, G. W. Hammett, M. A. Beer, W. Dorland, S. C. Cowley, S. A. Smith, H. G. W, M. A. Beer, W. Dorland, G. W. Hammett, M. A. Beer, W. Dorland, S. C. Cowley, and S. A. Smith, "Developments in the gyrofluid approach to tokamak turbulence simulations," *Plasma Physics and Controlled Fusion*, vol. 35, p. 973, 1993.

[65] X. Q. Xu, P. W. Xi, A. Dimits, I. Joseph, M. V. Umansky, T. Y. Xia, B. Gui, S. S. Kim, G. Y. Park, T. Rhee, H. Jhang, P. H. Diamond, B. Dudson, and P. B. Snyder, "Gyro-fluid and two-fluid theory and simulations of edge-localized-modes," *Physics of Plasmas*, vol. 20, p. 056113, 5 2013.

[66] D. Dickinson, "Wp14-frf-ccfe constructing a pedestal evolution model," 2016.

[67] W. Dorland, "Gyrofluid models of plasma turbulence," p. 222, 1993.

[68] T. T. Ribeiro and B. Scott, "Gyrofluid turbulence studies of the effect of the poloidal position of an axisymmetric debye sheath," *Plasma Physics and Controlled Fusion*, vol. 50, p. 055007, 5 2008.

[69] T. T. Ribeiro and B. Scott, "Tokamak turbulence computations on closed and open magnetic flux surfaces," *Plasma Physics and Controlled Fusion*, vol. 47, pp. 1657–1679, 10 2005.

[70] J. Madsen, "Full-f gyrofluid model," *Physics of Plasmas*, vol. 20, p. 072301, 7 2013.

[71] M. Held, "Full-f gyro-fluid modelling of the tokamak edge and scrape-off layer," 2017.

[72] B. D. Scott, "Gem – an energy conserving electromagnetic gyrofluid model," *arXiv*, p. 27, 1 2005.

[73] M. Wiesenberger, "Gyrofluid computations of filament dynamics in tokamak scrape-off layers by," 2014. NULL.

[74] B. D. D. Dudson, M. V. V. Umansky, X. Q. Q. Xu, P. B. B. Snyder, and H. R. R. Wilson, "Bout++: A framework for parallel plasma fluid simulations," *Computer Physics Communications*, vol. 180, pp. 1467–1480, 9 2009. ¡b¿From Duplicate 2 (¡i¿BOUT++: A framework for parallel plasma fluid simulations¡/i¿ - Dudson, B.D. D.; Umansky, M.V. V.; Xu, X.Q. Q.; Snyder, P.B. B.; Wilson, H.R. R.)¡br/¿¡/b¿¡br/¿¡b¿From Duplicate 1 (¡i¿BOUT++: A framework for parallel plasma fluid simulations¡/i¿ - Dudson, B D; Umansky, M V; Xu, X Q; Snyder, P B; Wilson, H R)¡br/¿¡/b¿¡br/¿NULL.

[75] B. D. Dudson, M. V. Umansky, X. Q. Xu, P. B. Snyder, and H. R. Wilson, "Bout++: a framework for parallel plasma fluid simulations," *Computer Physics Communications*, vol. 180, pp. 1467–1480, 10 2008.

[76] L. Easy, "Three dimensional simulations of scrape-off layer filaments," 2016.

[77] D. Schwörer, "On the influence of background including neutrals on the dynamics of 3d scrape-off layer filaments in fusion devices — enhanced reader," 2020.

[78] P. C. Stangeby, *The plasma boundary of magnetic fusion devices*. Institute of Physics Pub, 2000.

[79] F. Militello, T. Farley, K. Mukhi, N. Walkden, and J. T. Omotani, "A two-dimensional statistical framework connecting thermodynamic profiles with filaments in the scrape off layer and application to experiments," *Physics of Plasmas*, vol. 25, 5 2018.

[80] J. R. Angus, M. V. Umansky, and S. I. Krasheninnikov, "Effect of drift waves on plasma blob dynamics," *Physical Review Letters*, vol. 108, p. 215002, 5 2012.

[81] N. R. Walkden, B. D. Dudson, and G. Fishpool, "Characterization of 3d filament dynamics in a mast sol flux tube geometry," *Plasma Physics and Controlled Fusion*, vol. 55, p. 105005, 10 2013.

[82] D. Schwörer, N. R. Walkden, H. Leggate, B. D. Dudson, F. Militello, T. Downes, and M. M. Turner, "Influence of plasma background on 3d scrape-off layer filaments," *Plasma Physics and Controlled Fusion*, vol. 61, p. 025008, 2 2019.

[83] J. Olsen, J. Madsen, A. H. Nielsen, J. J. Rasmussen, and V. Naulin, "Temperature dynamics and velocity scaling laws for interchange driven, warm ion plasma filaments," *Plasma Physics and Controlled Fusion*, vol. 58, p. 044011, 4 2016.

[84] W. Fundamenski, O. E. Garcia, V. Naulin, R. A. Pitts, A. H. Nielsen, J. J. Rasmussen, J. Horacek, and J. P. Graves, "Dissipative processes in interchange driven scrape-off layer turbulence," *Nuclear Fusion*, vol. 47, pp. 417–433, 5 2007.

[85] L. Einkemmer and M. Wiesenberger, "A conservative discontinuous galerkin scheme for the 2d incompressible navier-stokes equations," *Computer Physics Communications*, vol. 185, pp. 2865–2873, 11 2014.

[86] A. Kendl, "Gyrofluid vortex interaction," *Plasma Physics and Controlled Fusion*, vol. 60, p. 025017, 2 2018.

[87] H. Hasegawa and S. Ishiguro, "Microscopic effect on filamentary coherent structure dynamics in boundary layer plasmas," *Plasma*, vol. 1, pp. 1–7, 3 2018.

[88] N. R. Walkden, B. D. Dudson, L. Easy, G. Fishpool, and J. T. Omotani, "Numerical investigation of isolated filament motion in a realistic tokamak geometry," *Nuclear Fusion*, vol. 55, p. 113022, 10 2015.

[89] K. T. Tsang, "Finite larmor radius stabilization of ballooning modes in tokamaks," *Physics of Fluids*, vol. 24, pp. 2017–2021, 6 1981.

[90] F. Militello, W. Fundamenski, V. Naulin, and A. H. Nielsen, "Simulations of edge and scrape off layer turbulence in mega ampere spherical tokamak plasmas," *Plasma Physics and Controlled Fusion*, vol. 54, p. 095011, 8 2012.

[91] N. R. Walkden, F. Riva, B. D. Dudson, C. Ham, F. Militello, D. Moulton, T. Nicholas, and J. T. Omotani, "3d simulations of turbulent mixing in a simplified slab-divertor geometry," *Nuclear Materials and Energy*, vol. 18, pp. 111–117, 1 2019.

[92] T. E. G. Nicholas, "Reduced simulations of scrape-off-layer turbulence," 2021.

[93] T. Eich, B. Sieglin, A. Scarabosio, W. Fundamenski, R. J. Goldston, and A. Herrmann, "Inter-elm power decay length for jet and asdex upgrade: Measurement and comparison with heuristic drift-based model," *Physical Review Letters*, vol. 107, p. 215001, 11 2011.

[94] C. J. Lasnier, M. A. Makowski, J. A. Boedo, S. L. Allen, N. H. Brooks, D. N. Hill, A. W. Leonard, J. G. Watkins, and W. P. West, "Scaling of divertor heat flux profile widths in diii-d," *Journal of Nuclear Materials*, vol. 415, pp. S353–S356, 8 2011.

[95] R. J. Goldston, "Heuristic drift-based model of the power scrape-off width in low-gas-puff h-mode tokamaks," *Nuclear Fusion*, vol. 52, p. 013009, 12 2011.

[96] J. W. Ahn, G. F. Counsell, and A. Kirk, "L-mode sol width scaling in the mast spherical tokamak," *Plasma Physics and Controlled Fusion*, vol. 48, p. 1077, 7 2006.

[97] A. J. Thornton, A. Kirk, and M. Team, "Scaling of the scrape-off layer width during inter-elm h modes on mast as measured by infrared thermography," *Plasma Physics and Controlled Fusion*, vol. 56, 6 2014.

[98] B. D. Scott, A. Kendl, and T. Ribeiro, "Nonlinear dynamics in the tokamak edge," *Contributions to Plasma Physics*, vol. 50, pp. 228–241, 2010.

[99] M. Held, M. Wiesenberger, J. Madsen, and A. Kendl, "The influence of temperature dynamics and dynamic finite ion larmor radius effects on seeded high amplitude plasma blobs," *Nuclear Fusion*, vol. 56, p. 126005, 9 2016.