Analysis and Visualisations of Edge Localised Modes in Tokamaks

A D McGann, B Dudson

York Plasma Institute, University of York, York, YO10 5DQ, UK

E-mail: `adm518@york.ac.uk`

Abstract. This report outlines work undertaken to extend the features of the 3D plasma fluid simulation code BOUT++ to make it simpler to visualise and analyse its output. The implementation of the visualisation routines is explained, with documentation on how it can be used. A brief introduction is given to the concept of the analysis routine to follow a field line, with details of its implementation and a test to ensure suitablity for purpose.

1. Introduction

The study of Edge Localised Modes (ELMs) in Tokamaks is crucial for the development of efficient and economically plausible Magnetic Confinement Fusion due to their large role in limiting high-performance modes (H-modes) of such devices. Tokamak development has been performed, in part, by computational simulations which allow the study of such phenomenon, and as such create predictions of their behaviour and how they can be controlled. However, these tools often produce output that is difficult to interpret and will require some degree of time and effort dedicated by the researchers to produce meaningful images and parameter values from their work.

From this, a need arises for standard routines for the analysis and visualisations of data produced for the purposes of studying the ouput of simulations on Tokamaks; to reduce time and effort on behalf of researchers, as well as reducing the duplication of efforts. The subject of these works was to extend the functionality of the 3D plasma fluid simulation code BOUT++[1] to include these utilities.

This report outlines, in a roughly chronological order, the development considerations, implementation, documentation and usage case studies for each of the routines that were written for BOUT++ throughout the project.

2. Visualising 3D fields

2.1. Introduction

Because of the 3D capabilities of BOUT++, many of its useful applications can be difficult to display in an intuitive manner, and so the full output can be presented such that much is left to the imagination. This not only makes the interpretation of the data more difficult, but it reduces the visual impact of the work. Therefore, an easy to use and robust method of producing good quality images that displays the information in the data accurately is important.

This section describes the details of the implementation, with methodology and ideas behind the tools created, documentation on how the tools can be used, and some case studies of use.

2.2. Considerations for creating the visualisations

Many of the simulations run using BOUT++ at the time of this works mainly considered a 'slab' geometry for a Tokamak. This is convenient for visualisation, as for analysing a simulation this 'unwrapped' view was seen to be more intuitive than a 'wrapped' view, where the image is representative of the real-space toroidal geometry of a Tokamak.

The suite used to create the images, 'Mayavi 2' [2], was chosen for its versitility in the types of plot possible, the ability to produce interactive applications to work with the data. Technical reasons for this choice include a python interface, maintaining compatibility with existing BOUT++ routines, as well as its 'open source' licensing, which allows for a greater userbase and compliments the open source nature of BOUT++.

2.3. Implementation of visualisation routines

For the purposes required for BOUT++, 'Mayavi' produces the required plots using a 'source and module' model, whereby a set of data is converted into a binary 'source', along with meta-information such as its structure. 'Modules' can be applied to such a data source, which generate the sub-plots from the 'source' to make up the image. The work described here makes use of the 'Mayavi' modules to build up standard routines dedicated to 'typical' outputs of BOUT++. The routines decided to be useful for such a task were:

Creation of interactive images from scalar values

Interactive images are a tool to display simulation data in a way such that the view is never constrained to those generated by a script, or having to adjust values unintuitively to produce the desired view. Instead, the picture is moveable, and the elements within the image, such as slices through data, are moveable, so the important details of the data are presented with maximum effect.

Static images from scalar values

Static images are very useful for those seeking to publish their work in a journal, on a poster or other static media. This is performed by converting the interactive images into static images, which is a scriptable process, described in 2.4.

Dynamic images with a fourth (time) dimension

A usual method of showing time-evolving data is by use of an animation; however, with to the ability to use python scripting behind the Mayavi toolset, the data from which an image is plotted can be changed by the user on demand (as opposed to the constant dataset change of an animation), and automatically replotted with the requested dataset. This is useful for both performing analysis on the data as well as for demonstration. The backend python functionality used to provide this relied on the 'Traits' API, included along with Mayavi 2 in the 'Enthought tool suite'. By applying the appropriate 'trait' to the time index of a 4D array of data, the API creates a dialogue box that presents to the user the time steps at which data is available, and replots the image according to the users' choice.

Section 2.4 describes how to work with the routines that were implemented, as well as small problems that occured during the writing of the routines, and ways they could be avoided.

2.4. Documentation for the routines

This documentation is not a replacement for that of 'Mayavi 2', indeed users seeking to create their own routines should look upon the upstream documentation for guidance.

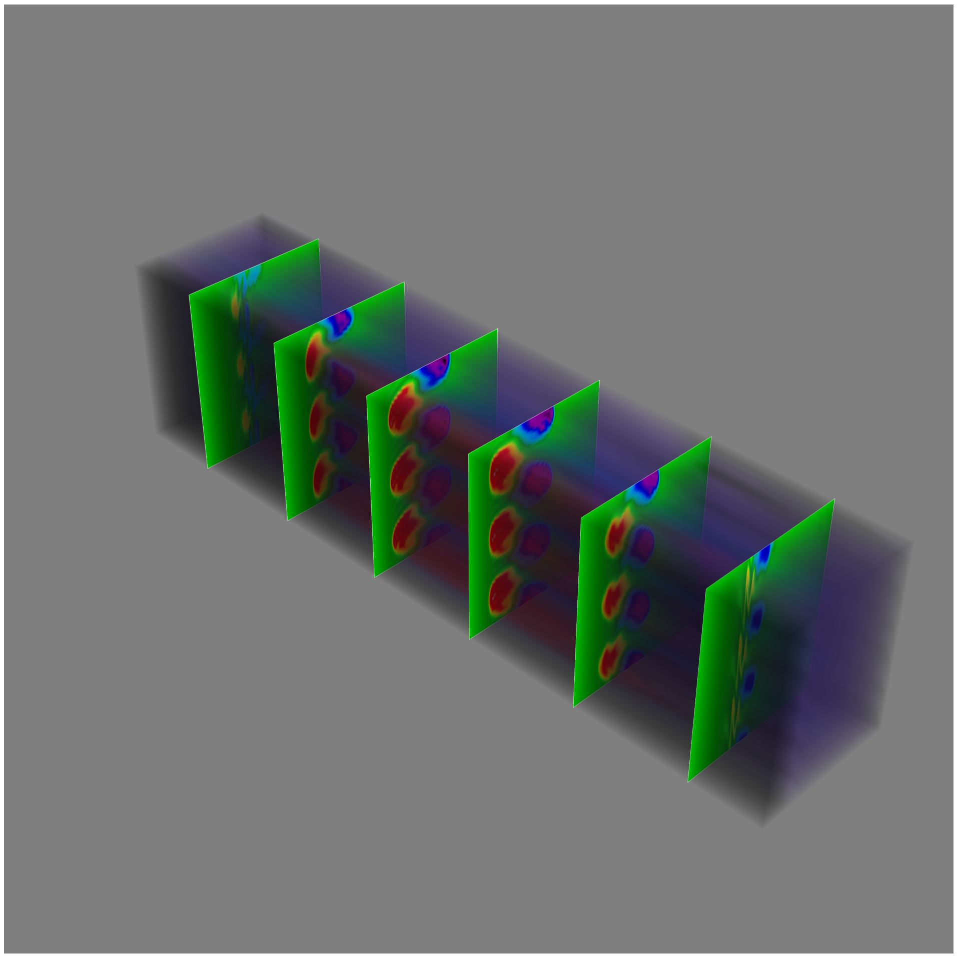


Figure 1. Typical visualisation of the output of BOUT++. Note the inner 'volume' and the outer 'slices

This is here to provide a guide to how to use the tools written for BOUT++ during 'typical' usage.

2.4.1. Generating interactive images

A function was written to automatically generate an interactive image which has a volume of scalar data, intersected by slices of higher detail, such as in figure 2.4.1. In BOUT++, this is the `two_sfields` function. The function takes two arguments, two 3-dimensional scalar fields (the first providing the volume and the second being the slices throughout the volume). The two fields need not be the same data, so data can be contrasted. If the shape of the plot is not suitable, for example if the x-plane is larger than the y-plane, so the data looks stretched, a three-element array can be passed to it with the desired scaling factors in x, y and z.

If this kind of plot does not suit, a custom mayavi script will need to be implemented. To do this the input data should be inserted as a scalar field vtk data type, then the mayavi visualisation tools can be applied, for which the mayavi (specifically 'mlab') documentation should be referred.

2.4.2. Generating static images

Because the mayavi toolset is principally designed for interactive diagrams, there are some quirks to drawing static images. The plot must first be generated using a method as described in 2.4.1. To create a picture using the function '`two_sfields`', the `filename` and `size` keyword arguments must be used. Pass a filename (as a string) and a requested size (for an $x \times y$ pixel image, this argument should be `[x,y]`), to the function and it will produce an output with a format suiting the filename passed to it.

If a different view other than the default is required, or a static image from a custom plot, the BOUT++ function will not suffice. To reorient the figure first the desired coefficients for that rotation must be obtained. To do this, it is recommended to run an interactive figure in the mayavi UI environment:

```
$> mayavi2 -x your_script
```

orient the figure to your liking, and obtain the coefficients by running (in the python shell in the mayavi UI):

```
>>> mlab.view()
```

which returns the 'azimuth', 'elevation', 'distance' and 'roll' coefficients of the image, respectively. These can be used as keyword arguments for `mlab.view` in your script, and will orient your figure as you chose it.

To generate a static output file from these images, the `mlab.savefig` command is used. It produces an image with an extension according to the filename passed to it (for example a .png extension will produce a 'portable network graphic' image). For a greater image size, it was found that (due to the method of enlargening images in mayavi), glitches in the output would occur unless the `magnification` keyword argument was not set to unity, instead setting `size` to enlarge the image. For example, to create a 1920x1080 pixel image:

```
mlab.savefig(figure.png, size=[1920,1080], magnification=1.)
```

would be a suitable line in a script. Furthermore, to avoid having inconvenient window sizes on screen, off-screen rendering should be used:

```
mlab.options.offscreen = True
```

as this avoids artifacts on the image. Refer to the Mayavi documentation for more information on this topic.

2.4.3. Data interactivity

To create an image with a variable data set (such as changing the time step of the data), the Traits API must be used. This is a more complicated task than plotting an interactive figure, as the whole scene must be removed before regenerating the image. An example script is included in BOUT++, (`vis.py` and `vis2.py`) which takes in a 4D set of data, and plots in an interactive window a volume of a single timestamp. For large datasets it reduces the number of data points that are plotted in the volume so features can be identified, and the whole dataset is available on demand to show all the detail.

An experimental feature is in `vis.py`, which attempts to keep the most recently plotted scenes in a list, so as to reduce time between plotting the same time data (for example, if there are interesting features at time = 0 and 10, the two would not have to be redrawn by mayavi every time they were switched between), however there are some unexpected behaviours in mayavi which need to be worked around before this feature should be used.

Automated generation of multiple subplots (such as multiple slices) is not implemented in the routines included in BOUT++, and an elegant solution was not found for this problem. Currently, the only way to perform this would be to manually add all of the indiviual sub-plots to the script by hand (for example:
`mlabscene.children[0].children[`n`] = volume subplot`
where n is the nth sub-plot in the scene, added to the BOUT++ routines).

2.5. Case studies of the visualisations

The work described here has proved to be very popular with those who have 3-dimensional data, and has already seen use in conferences and on promotional material for the Department of Physics at The University of York. Two posters that have been displayed at conferences are shown in figures 2.4.3 and 2.4.3. One of the researchers involved in the creation of figure 2.4.3 stated that the images were useful because they were striking artistically, as well as nicely presenting 3D data, which is a challenge [3].

2.6. Summary of visualisation routines

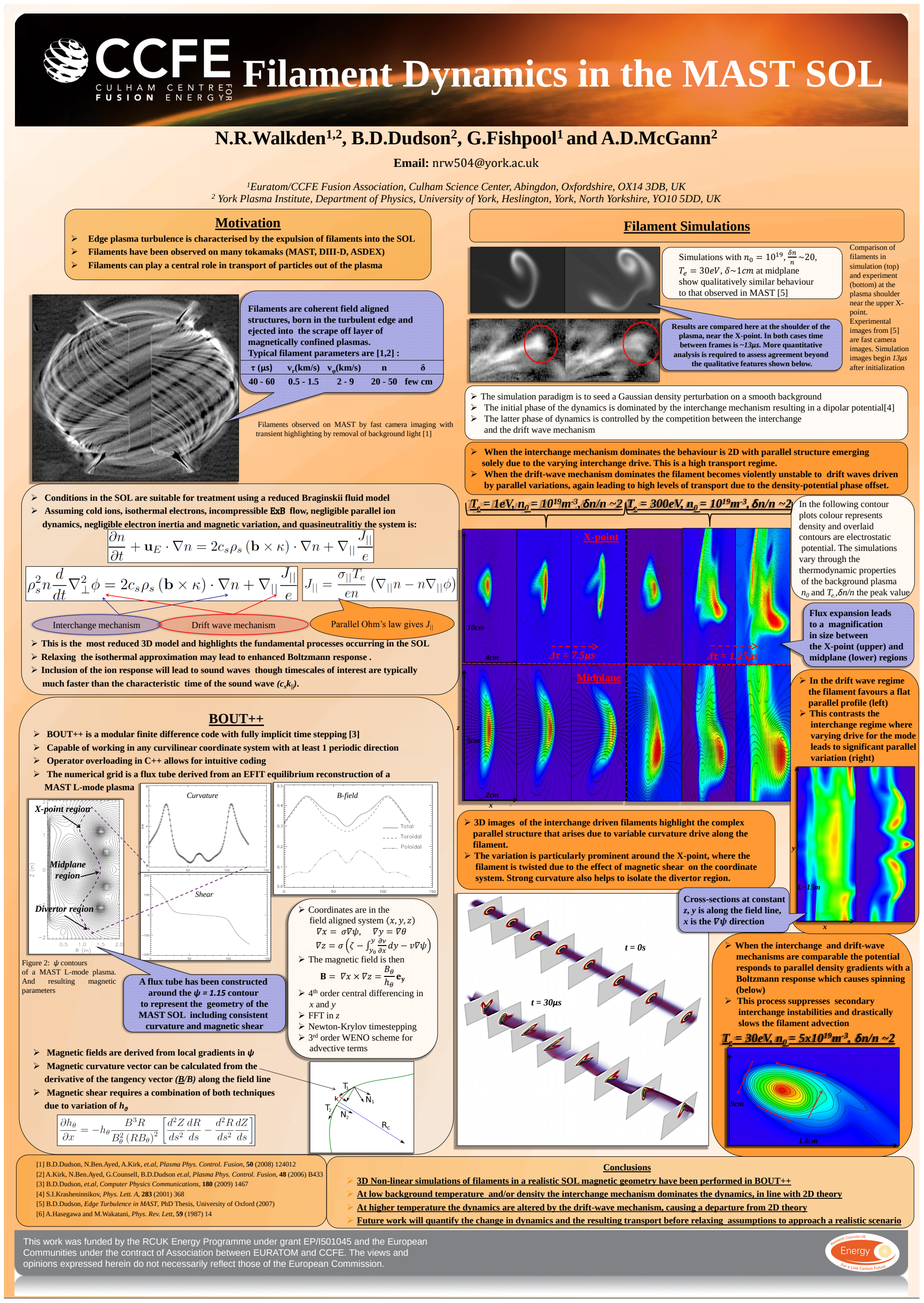
Creating the visualisation routines was one of the main tasks for this project. The plots that were implemented as standard routines were deemed useful in the plasma physics field, so as to maximise usage and minimise duplication of efforts on behalf of researchers using BOUT++; they were well-recieved, and so the routines implemented should become useful tools.

3. Data Analysis routines

3.1. Introduction

BOUT++ has many data analysis routines implemented in IDL, however this is non-free software and requires learning a programming language purely for analysis, as it is not general purpose enough for computationally intensive tasks.

This section describes how some of the functionality of IDL was implemented in the python programming language, which has more modern features and is more general purpose than IDL. Furthermore, some routines were implemented to make BOUT++ easier to use, making some analysis tasks quicker to learn for beginners. Here, the theory of the analysis routines is outlined, as well as the implementation and documentation.



CULHAM CENTRE FOR FUSION ENERGY

Filament Dynamics in the MAST SOL

N.R.Walkden[1,2], B.D.Dudson[2], G.Fishpool[1] and A.D.McGann[2]

Email: nrw504@york.ac.uk

[1]*Euratom/CCFE Fusion Association, Culham Science Center, Abingdon, Oxfordshire, OX14 3DB, UK*
[2] *York Plasma Institute, Department of Physics, University of York, Heslington, York, North Yorkshire, YO10 5DD, UK*

Motivation

- **Edge plasma turbulence is characterised by the expulsion of filaments into the SOL**
- **Filaments have been observed on many tokamaks (MAST, DIII-D, ASDEX)**
- **Filaments can play a central role in transport of particles out of the plasma**

Filaments are coherent field aligned structures, born in the turbulent edge and ejected into the scrape off layer of magnetically confined plasmas.
Typical filament parameters are [1,2] :

τ (μs)	v_r(km/s)	v_φ(km/s)	n	δ
40 - 60	0.5 - 1.5	2 - 9	20 - 50	few cm

Filaments observed on MAST by fast camera imaging with transient highlighting by removal of background light [1]

- **Conditions in the SOL are suitable for treatment using a reduced Braginskii fluid model**
- **Assuming cold ions, isothermal electrons, incompressible ExB flow, negligible parallel ion dynamics, negligible electron inertia and magnetic variation, and quasineutralitiy the system is:**

$$\frac{\partial n}{\partial t} + \mathbf{u}_E \cdot \nabla n = 2c_s\rho_s\left(\mathbf{b}\times\kappa\right)\cdot\nabla n + \nabla_{||}\frac{J_{||}}{e}$$

$$\rho_s^2 n\frac{d}{dt}\nabla_\perp^2\phi = 2c_s\rho_s\left(\mathbf{b}\times\kappa\right)\cdot\nabla n + \nabla_{||}\frac{J_{||}}{e} \qquad J_{||} = \frac{\sigma_{||}T_e}{en}\left(\nabla_{||}n - n\nabla_{||}\phi\right)$$

- **This is the most reduced 3D model and highlights the fundamental processes occurring in the SOL**
- **Relaxing the isothermal approximation may lead to enhanced Boltzmann response .**
- **Inclusion of the ion response will lead to sound waves though timescales of interest are typically much faster than the characteristic time of the sound wave *($c_s k_{||}$)*.**

BOUT++

- **BOUT++ is a modular finite difference code with fully implicit time stepping [3]**
- **Capable of working in any curvilinear coordinate system with at least 1 periodic direction**
- **Operator overloading in C++ allows for intuitive coding**
- **The numerical grid is a flux tube derived from an EFIT equilibrium reconstruction of a MAST L-mode plasma**

Figure 2: ψ contours of a MAST L-mode plasma. And resulting magnetic parameters

A flux tube has been constructed around the *ψ = 1.15* contour to represent the geometry of the MAST SOL including consistent curvature and magnetic shear

- Coordinates are in the field aligned system (x, y, z)
  $\nabla x = \sigma\nabla\psi, \quad \nabla y = \nabla\theta$
  $\nabla z = \sigma\left(\zeta - \int_{y_0}^{y}\frac{\partial v}{\partial x}dy - v\nabla\psi\right)$
- The magnetic field is then
  $\mathbf{B} = \nabla x\times\nabla z = \frac{B_\theta}{h_\theta}\mathbf{e_y}$
- 4th order central differencing in x and y
- FFT in z
- Newton-Krylov timestepping
- 3rd order WENO scheme for advective terms

- **Magnetic fields are derived from local gradients in *ψ***
- **Magnetic curvature vector can be calculated from the derivative of the tangency vector *($\underline{B}$/B)* along the field line**
- **Magnetic shear requires a combination of both techniques due to variation of *h_ϑ***

$$\frac{\partial h_\theta}{\partial x} = -h_\theta\frac{B^3R}{B_\theta^2\left(RB_\theta\right)^2}\left[\frac{d^2Z}{ds^2}\frac{dR}{ds} - \frac{d^2R}{ds^2}\frac{dZ}{ds}\right]$$

Filament Simulations

Simulations with $n_0 = 10^{19}, \frac{\delta n}{n}\sim 20$, $T_e = 30eV, \delta\sim 1cm$ at midplane show qualitatively similar behaviour to that observed in MAST [5]

Results are compared here at the shoulder of the plasma, near the X-point. In both cases time between frames is *~13μs*. More quantitative analysis is required to assess agreement beyond the qualitative features shown below.

Comparison of filaments in simulation (top) and experiment (bottom) at the plasma shoulder near the upper X-point. Experimental images from [5] are fast camera images. Simulation images begin *13μs* after initialization

- The simulation paradigm is to seed a Gaussian density perturbation on a smooth background
- The initial phase of the dynamics is dominated by the interchange mechanism resulting in a dipolar potential[4]
- The latter phase of dynamics is controlled by the competition between the interchange and the drift wave mechanism

- **When the interchange mechanism dominates the behaviour is 2D with parallel structure emerging solely due to the varying interchange drive. This is a high transport regime.**
- **When the drift-wave mechanism dominates the filament becomes violently unstable to drift waves driven by parallel variations, again leading to high levels of transport due to the density-potential phase offset.**

T_e = 1eV, n_0 = $10^{19}m^{-3}$, δn/n ~2 ***T_e = 300eV, n_0 = $10^{19}m^{-3}$, δn/n ~2***

In the following contour plots colour represents density and overlaid contours are electrostatic potential. The simulations vary through the thermodynamic properties of the background plasma n_0 and T_e,δn/n the peak value

Flux expansion leads to a magnification in size between the X-point (upper) and midplane (lower) regions

- **In the drift wave regime the filament favours a flat parallel profile (left)**
- **This contrasts the interchange regime where varying drive for the mode leads to significant parallel variation (right)**

- **3D images of the interchange driven filaments highlight the complex parallel structure that arises due to variable curvature drive along the filament.**
- **The variation is particularly prominent around the X-point, where the filament is twisted due to the effect of magnetic shear on the coordinate system. Strong curvature also helps to isolate the divertor region.**

Cross-sections at constant *z*, *y* is along the field line, *x* is the *$\nabla\psi$* direction

- **When the interchange and drift-wave mechanisms are comparable the potential responds to parallel density gradients with a Boltzmann response which causes spinning (below)**
- **This process suppresses secondary interchange instabilities and drastically slows the filament advection**

T_e = 30eV, n_0 = 5x$10^{19}m^{-3}$, δn/n ~2

[1] B.D.Dudson, N.Ben.Ayed, A.Kirk, *et.al*, *Plasma Phys. Control. Fusion*, **50** (2008) 124012
[2] A.Kirk, N.Ben.Ayed, G.Counsell, B.D.Dudson *et.al*, *Plasma Phys. Control. Fusion*, **48** (2006) B433
[3] B.D.Dudson, *et.al*, *Computer Physics Communications*, **180** (2009) 1467
[4] S.I.Krasheninnikov, *Phys. Lett. A*, **283** (2001) 368
[5] B.D.Dudson, *Edge Turbulence in MAST*, PhD Thesis, University of Oxford (2007)
[6] A.Hasegawa and M.Wakatani, *Phys. Rev. Lett*, **59** (1987) 14

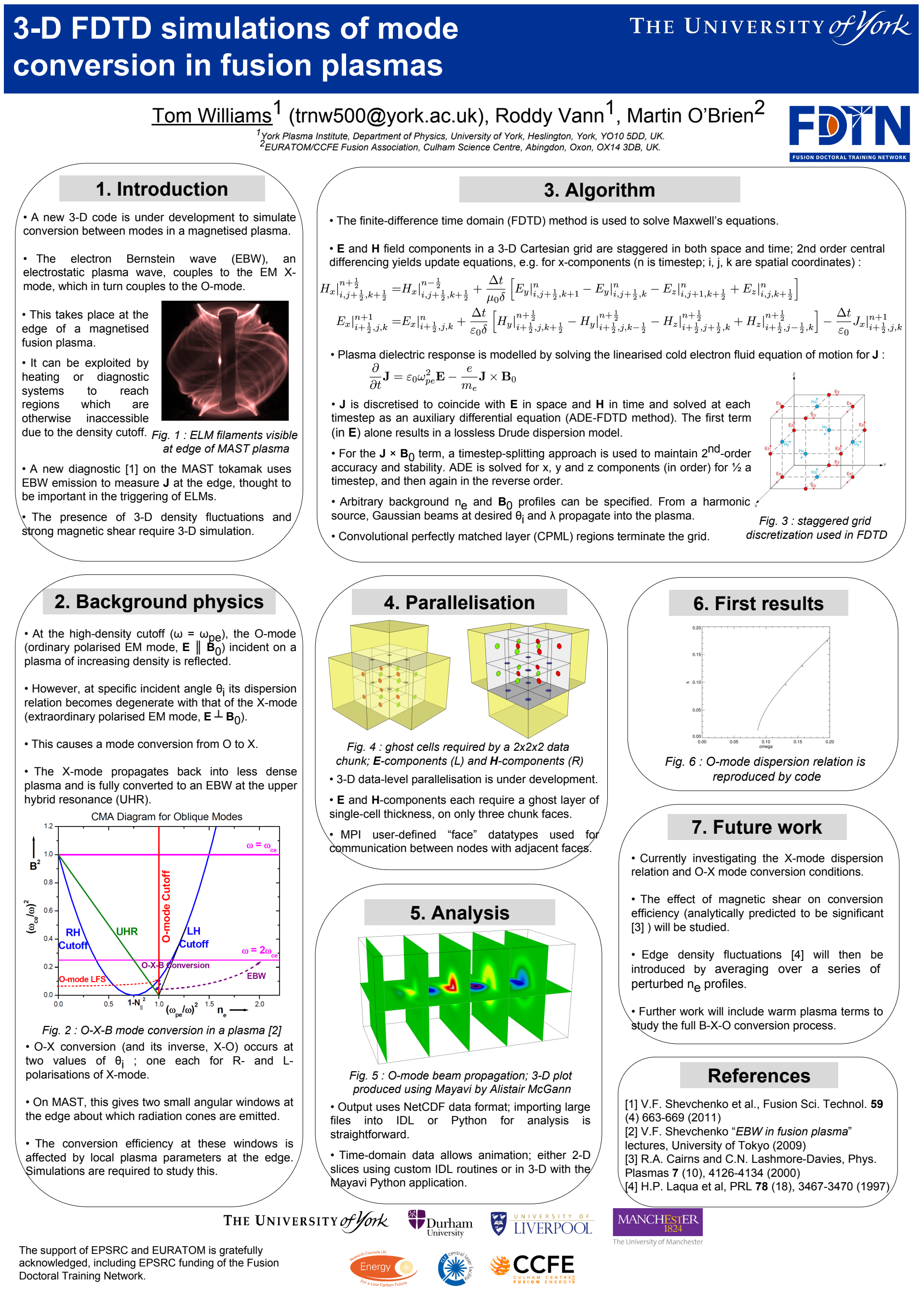
Conclusions

- **3D Non-linear simulations of filaments in a realistic SOL magnetic geometry have been performed in BOUT++**
- **At low background temperature and/or density the interchange mechanism dominates the dynamics, in line with 2D theory**
- **At higher temperature the dynamics are altered by the drift-wave mechanism, causing a departure from 2D theory**
- **Future work will quantify the change in dynamics and the resulting transport before relaxing assumptions to approach a realistic scenario**

This work was funded by the RCUK Energy Programme under grant EP/I501045 and the European Communities under the contract of Association between EURATOM and CCFE. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Research Councils UK Energy For a Low Carbon Future

Figure 2. A poster, courtesy of Nicolas Walkden, demonstrating the use of some of the images created with these routines, on data produced with BOUT++(picture at bottom right)



3-D FDTD simulations of mode conversion in fusion plasmas

THE UNIVERSITY *of York*

Tom Williams[1] (trnw500@york.ac.uk), Roddy Vann[1], Martin O'Brien[2]

[1] *York Plasma Institute, Department of Physics, University of York, Heslington, York, YO10 5DD, UK.*
[2] *EURATOM/CCFE Fusion Association, Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK.*

FDTN FUSION DOCTORAL TRAINING NETWORK

1. Introduction

• A new 3-D code is under development to simulate conversion between modes in a magnetised plasma.

• The electron Bernstein wave (EBW), an electrostatic plasma wave, couples to the EM X-mode, which in turn couples to the O-mode.

• This takes place at the edge of a magnetised fusion plasma.

• It can be exploited by heating or diagnostic systems to reach regions which are otherwise inaccessible due to the density cutoff.

Fig. 1 : ELM filaments visible at edge of MAST plasma

• A new diagnostic [1] on the MAST tokamak uses EBW emission to measure **J** at the edge, thought to be important in the triggering of ELMs.

• The presence of 3-D density fluctuations and strong magnetic shear require 3-D simulation.

3. Algorithm

• The finite-difference time domain (FDTD) method is used to solve Maxwell's equations.

• **E** and **H** field components in a 3-D Cartesian grid are staggered in both space and time; 2nd order central differencing yields update equations, e.g. for x-components (n is timestep; i, j, k are spatial coordinates) :

$$H_x|^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} = H_x|^{n-\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} + \frac{\Delta t}{\mu_0 \delta}\left[E_y|^{n}_{i,j+\frac{1}{2},k+1} - E_y|^{n}_{i,j+\frac{1}{2},k} - E_z|^{n}_{i,j+1,k+\frac{1}{2}} + E_z|^{n}_{i,j,k+\frac{1}{2}}\right]$$

$$E_x|^{n+1}_{i+\frac{1}{2},j,k} = E_x|^{n}_{i+\frac{1}{2},j,k} + \frac{\Delta t}{\varepsilon_0 \delta}\left[H_y|^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k+\frac{1}{2}} - H_y|^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k-\frac{1}{2}} - H_z|^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k} + H_z|^{n+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k}\right] - \frac{\Delta t}{\varepsilon_0} J_x|^{n+1}_{i+\frac{1}{2},j,k}$$

• Plasma dielectric response is modelled by solving the linearised cold electron fluid equation of motion for **J** :

$$\frac{\partial}{\partial t}\mathbf{J} = \varepsilon_0 \omega_{pe}^2 \mathbf{E} - \frac{e}{m_e}\mathbf{J} \times \mathbf{B}_0$$

• **J** is discretised to coincide with **E** in space and **H** in time and solved at each timestep as an auxiliary differential equation (ADE-FDTD method). The first term (in **E**) alone results in a lossless Drude dispersion model.

• For the **J** × $\mathbf{B}_0$ term, a timestep-splitting approach is used to maintain 2nd-order accuracy and stability. ADE is solved for x, y and z components (in order) for ½ a timestep, and then again in the reverse order.

• Arbitrary background n_e and $\mathbf{B}_0$ profiles can be specified. From a harmonic source, Gaussian beams at desired θ_i and λ propagate into the plasma.

• Convolutional perfectly matched layer (CPML) regions terminate the grid.

Fig. 3 : staggered grid discretization used in FDTD

2. Background physics

• At the high-density cutoff (ω = ω_{pe}), the O-mode (ordinary polarised EM mode, **E** ∥ $\mathbf{B}_0$) incident on a plasma of increasing density is reflected.

• However, at specific incident angle θ_i its dispersion relation becomes degenerate with that of the X-mode (extraordinary polarised EM mode, **E** ⊥ $\mathbf{B}_0$).

• This causes a mode conversion from O to X.

• The X-mode propagates back into less dense plasma and is fully converted to an EBW at the upper hybrid resonance (UHR).

CMA Diagram for Oblique Modes

Fig. 2 : O-X-B mode conversion in a plasma [2]

• O-X conversion (and its inverse, X-O) occurs at two values of θ_i ; one each for R- and L-polarisations of X-mode.

• On MAST, this gives two small angular windows at the edge about which radiation cones are emitted.

• The conversion efficiency at these windows is affected by local plasma parameters at the edge. Simulations are required to study this.

4. Parallelisation

*Fig. 4 : ghost cells required by a 2x2x2 data chunk; **E**-components (L) and **H**-components (R)*

• 3-D data-level parallelisation is under development.

• **E** and **H**-components each require a ghost layer of single-cell thickness, on only three chunk faces.

• MPI user-defined "face" datatypes used for communication between nodes with adjacent faces.

5. Analysis

Fig. 5 : O-mode beam propagation; 3-D plot produced using Mayavi by Alistair McGann

• Output uses NetCDF data format; importing large files into IDL or Python for analysis is straightforward.

• Time-domain data allows animation; either 2-D slices using custom IDL routines or in 3-D with the Mayavi Python application.

6. First results

Fig. 6 : O-mode dispersion relation is reproduced by code

7. Future work

• Currently investigating the X-mode dispersion relation and O-X mode conversion conditions.

• The effect of magnetic shear on conversion efficiency (analytically predicted to be significant [3]) will be studied.

• Edge density fluctuations [4] will then be introduced by averaging over a series of perturbed n_e profiles.

• Further work will include warm plasma terms to study the full B-X-O conversion process.

References

[1] V.F. Shevchenko et al., Fusion Sci. Technol. **59** (4) 663-669 (2011)
[2] V.F. Shevchenko "*EBW in fusion plasma*" lectures, University of Tokyo (2009)
[3] R.A. Cairns and C.N. Lashmore-Davies, Phys. Plasmas **7** (10), 4126-4134 (2000)
[4] H.P. Laqua et al, PRL **78** (18), 3467-3470 (1997)

THE UNIVERSITY *of York* Durham University UNIVERSITY OF LIVERPOOL MANCHESTER 1824 The University of Manchester CCFE CULHAM CENTRE FUSION ENERGY

The support of EPSRC and EURATOM is gratefully acknowledged, including EPSRC funding of the Fusion Doctoral Training Network.

Figure 3. A poster, courtesy of Tom Williams, demonstrating the versatility of the plotting software, with a visualisation (centre bottom) of data not produced by BOUT++

3.2. Implementation of the analysis routines

The analysis routines written during this project were:

A fieldline tracer for perturbed magnetic fields

The ability to know where the field lines of a plasma instability, such as those found in Tokamaks, are very important for identifying properties about those instabilaties, such as growth rate. This important factor is so useful to calculate by the users of BOUT++, and should be accessable. Here was implemented a simple, but robust method of calculating the field lines. The theory came from a paper written on the geometry used in BOUT++ simulations [4], and simplified for the purposes of ease of implementation. This simplification is described in 3.3, and the documentation for this is in 3.4.

A user interface for the simple loading of data files

A simple interface to access the grid and output files typical of BOUT++ that allow all of the variables in the output to be laid out, and loaded into a python interface to prevent having to change source code of analysis routines to load in a different dataset was written to allow for faster and easier way of viewing and contrasting different datasets makes the experimental process of using BOUT++ quicker.

3.3. Theory for the field line tracer

The detailed mathematics behind how a field line can be traced without use of contour data is far beyond the scope of this document, and is treated in the guide to tracing field lines guide included with BOUT++, here some of the equations derived within [4], and reduce them to a computational form. The concept of tracing the field line is simple, to take the pertubation of the magnetic field at a point, and add it to the equilibrium field, thus giving the magnetic field throughout the simulated data. The pertubation is calculated from the parallel magnetic field component of the vector potential (an output of BOUT++), and a numerical derivative of this performed to give the magnetic field at one point on a single toroidal angle plane. Because the output of BOUT++ is in flux-aligned co-ordinates, the field to be traced out only depends on the total magnetic field (pertubation and equilibrium) of the magnetic field, and the metric tensor (which can be calculated as part of the output). They are combined as shown in section 3 of [4].

The differentiation of the parallel magnetic field vector potential ($A_{||}$) is performed by cubic 2D interpolation of each poloidal 'slice' of data that is in the output of BOUT++ to obtain a coefficients that can be used to create a continuum of values and first-order derivatives in x and z; cubic 1D interpolation is performed in the y direction to provide full derivatives in all co-ordinates.

3.4. Documentation for the field line tracer

The routine for tracing a field line is implemented in the `field_trace` file of the `boututils` python module included in BOUT++. Necessary arguments are a 3-dimensional dataset of parallel magnetic field vector potential and the relevant BOUT++ grid file for the data, which needs to be imported using the `file_import` function provided as part of BOUT++.

These default arguments will return a 2 dimensional array of the form:
`[(0,1,2),:]`
where the first index denotes the dimension (x, y and z), respectively, and the second contains the co-ordinate value along the corresponding dimension at each poloidal slice. By default, the field line tracer would start at a random point on the (x, z) surface of the data. The `x_0` and `z_0` keyword arguments allow the starting position to be set. This is useful for periodic data, so the starting position can be set as the final position of the field line after the previous tracing step, for example. If these are outside the range of the surface, random points will be chosen.

The final keyword arguments refer to the interpolation coefficients. If the data being analysed is the same, it is recommended to obtain the coefficients of the data once by setting the full_output argument to nonzero:
```
tck = field_trace(data, grid_file, full_output = 1)[1]
```
The array which is returned (as the second output of `field_trace`) can then be used as the `tck` keyword argument. This will avoid unnecessary computation of the same values.

3.5. Testing the field line tracer

The field line tracer was benchmarked by tracing a rational surface of simulated tokamak data. This is a good benchmark as the field line on a rational surface must return to itself after an integer number of torioidal rotations, so if the code is not seen to demonstrate this property is is not accurate enough for use. The data supplied to do this was a toroidal section, $1/64^{th}$ of a whole tokamak. To simulate tracing a whole field line of a tokamak, the routine was run 64 times through this section. The results of this are shown in figure 3.5 and the field line corresponding to this in figure3.5, which show that the field line does indeed return to itself after an equivalent of 12 rotations of the torus. This shows that the code is usable for its purpose.

4. Summary

This project was aimed at adding visualisation routines and analysis routines to BOUT++. This was performed in the Python programming language, for ease of use and to allow those who do not have access to proprietary software to make use of BOUT++. The visualisation routines proved to be very succesful, with the images produced being of production quality.

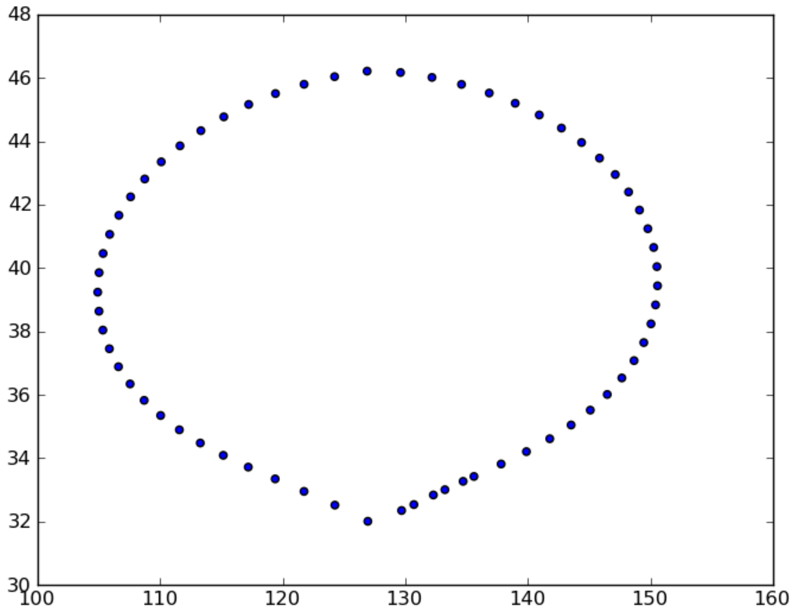


Figure 4. An x,z scatter plot of the values of the field line at every 64 points, at the end of each 'run' though the section of data that is assumed to be the same around all the volume. Note that at the lower section of the ellipse, the data points seem to 'wrap around' each other. This shows that the field line tracer has traced the rational surface accurately, as otherwise it would not have joined up with itself. The kink in the bottom is a result of the shape of the test data used.

The field line tracing analysis routine proved accurate enough for use in BOUT++, and will be included in the plasma simulation code in a future release.

Bibliography

[1] B D Dudson, M V Umansky, X Q Xu, P B Snyder, and H R Wilson. BOUT++: A framework for parallel plasma fluid simulations. *Computer Physics Communications*, 180:1467–1480, 2009.

[2] P. Ramachandran and G. Varoquaux. Mayavi: 3D Visualization of Scientific Data. *Computing in Science & Engineering*, 13(2):40–51, 2011.

[3] R Vann, 2012. Private Communications.

[4] J.P. Sauppe. Tracing magnetic field lines in bout. Technical report, University of Wisconsin-Madison, 2011.

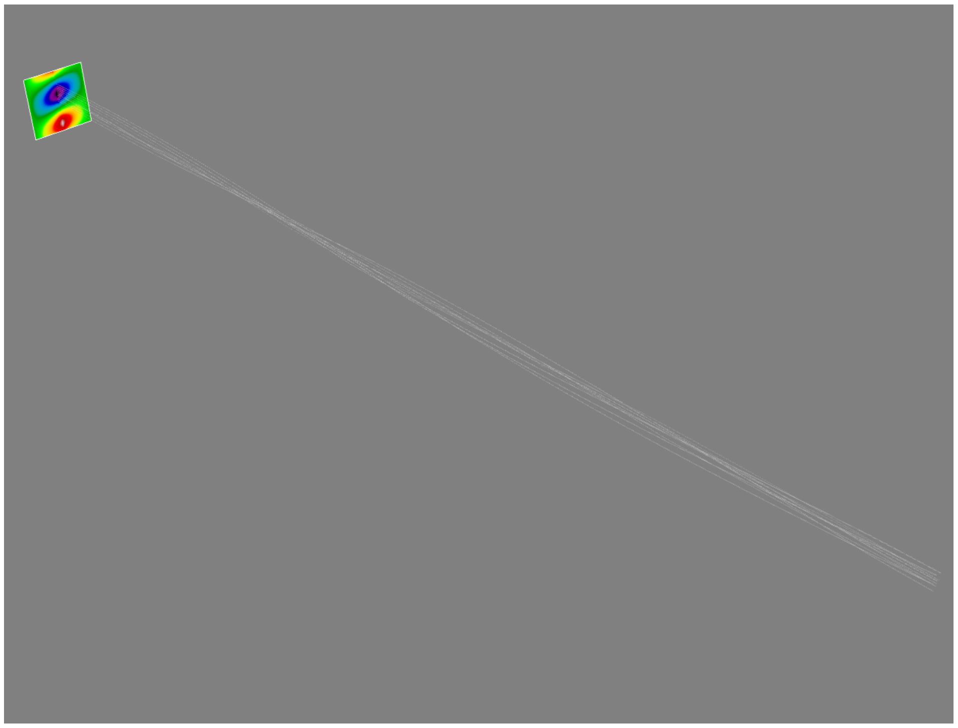


Figure 5. A bundle of field lines produced from random starting points on the initial surface of the parallel magnetic field vector potential