UK Atomic Energy Authority

ExCALIBUR

Fluid Referent Models

D2.6.2

This report describes work for ExCALIBUR project NEPTUNE at Milestone 2.6.2. It comprises the documents 2047356-TN-01[1] (11 pp), 2047356-TN-02-2[2] (20 pp), 2047356-TN-03-2[3] (12 pp), and 2047356-TN-04-2[4] (14 pp), which deal with work assigned task numbers 0.1, 1.1, 1.2, and 2.1 respectively, as of November 23, 2021. The first of these describes the basic infrastructure established for NEPTUNE; the latter three describe tests for and implementations of elliptic solvers and tests of fluid models, all within the BOUT++ software framework [5] maintained by the reports' authors. The reports have the status of living documents and are available from the NEPTUNE github repository `https://github.com/ExCALIBUR-NEPTUNE/`.

The report 0.1, titled *Environment*, describes infrastructure for coordinating NEPTUNE activities, comprising the establishment of a development environment and a framework for evaluating parallel scaling. The development environment includes a Slack workspace, a public github repository for code and documentation, and a private github repository for non-public documents. ReadTheDocs has been set up to build and host documentation automatically. Correctness tests are intended to be run via github actions, mirroring the approach used by the developers of the Nektar++ software framework [6], in which Docker images are used to run the test suite. Frameworks for performance testing are considered, with the conclusion that a new tool is required for NEPTUNE, building on desirable aspects identified in existing systems. Community-building work is detailed, this including progress made by the BOUT++ team at the VECMA hackathon held in January 2021 and accounts of relevant events attended, including performance analysis and SYCL workshops. These activities are supported by regular meetings with other grantees e.g. Oxford / Warwick (theory) and STFC (preconditioners).

The report 1.1, titled *Elliptic solver tests*, gives an explanation of how elliptic problems arise in plasma physics, with reference to relevant geometries. These problems generally arise for components of the electromagnetic four-potential in the non-relativistic limit in which the displacement current can be neglected. It is explained that the equation for the potential may take two- or three-dimensional form, depending on the pitch angle of the magnetic field - in the former case the gradients parallel to the magnetic field are neglected. Several classes of correctness test are listed: round-trip, analytic, and the method of manufactured solutions, with root-mean-square and maximum absolute error error norms used to assess accuracy. A discussion of performance tests focuses on issues specific to the plasma physics domain e.g. there may be large numbers of repeated solves during time-integration, which are serial unless parallel-in-time methods are used. Further sections discuss tests in geometries of increasing complexity, beginning with simple slabs, through the introduction of an artificial pole to simulate an X-point, to tokamak-like toroidal geometries; example BOUT++ input files are provided. Finally, two time-evolving systems are examined: the shear Alfvén wave (a coupled electromagnetic / charged particle wave travelling in the direction of the magnetic field) and the geodesic acoustic wave (an axisymmetric plasma oscillation occurring in toroidal geometries). Some example solutions of the resulting time-dependent elliptic problems are presented, and energy conservation is adopted as a measure of physical correctness and numerical stability.

Elliptic solver implementations in BOUT++, and existing preconditioner techniques for these, are described in the report 1.2, titled *Elliptic solver implementation*. A range of dimensionalities from one to three is considered, with the lower-dimensional solutions being useful for preconditioning the harder three-dimensional problem (thus presenting an example of a physics-informed preconditioner); one further simplification is to assume a constant plasma density - known by analogy as the Boussinesq approximation - when deriving a preconditioner.

One-dimensional solvers assume a constant density in toroidal angle, giving a system of second-order ordinary differential equations in the radial coordinate. Second-order central differencing results in a complex tridiagonal system per mode, which may be solved either by direct or iterative methods. Two-dimensional solvers (enacted in the toroidal planes, enabling the geometric factors to be constant in one direction - the toroidal) use either the Naulin preconditioning strategy (using a one-dimensional solver to correct iteratively for non-constant density), a geometric multigrid implementation, or solvers from PETSc. Three-dimensional solvers include implementations in PETSc and a relatively new Hypre implementation supporting GPU operation.

The Alfvén wave test case outlined in the previous report 1.1 uses a two-dimensional iterative PETSc solver in an X-point geometry taken from an experimental DIII-D tokamak equilibrium, giving a time-dependent problem involving an elliptic solve at each time step. Direct solvers are compared to an iterative method with regard to numerical stability - stability issues were found with longer-running direct solve simulations and probably result from the lack of diagonal dominance and the accumulation of individually small errors in each solve. Note that the strategy of using the direct solver as a preconditioner for the iterative results in a stable method.

New tests on the 23-PFLOP Lassen system in collaboration with LLNL find that some portions of the code show significant speed-up on a GPU accelerator (in summary, solve time and matrix assembly show respective speed improvement factors of 6.9 and 6 on four NVIDIA V100 GPUs compared to 40 IBM Power9 CPUs) but the global performance increase factor is only about 1.6, due to other portions of the code either not having been ported to GPUs or even performing worse thereupon. The results demonstrate also that matrices and preconditioners need to be used repeatedly (while stored *in situ* on the GPU) in order to realize the maximum possible gains in execution speed.

The aim of the report 2.1, titled *1D fluid model tests*, is to present tests of a one-dimensional fluid solver with UQ and realistic boundary conditions; due to the UQ requirement of repeated runs, the tests are designed to run quickly.

Analytic single-species tests are intended to exercise individual parts of the solver in the absence of couplings (therefore representing unit tests). A simplified one-dimensional scenario, with diffusion purely in the direction of the magnetic field, represents heat conduction (nonlinear because the diffusion coefficient for the temperature depends on the temperature via the usual Braginskii theory); various boundary conditions are tested, including fixed-temperature end points, fixed input power at one end point, or a more involved method approximating radiative energy losses.

Similar tests in higher-dimensional geometries are delegated to a reference. One-dimensional treatments of fluid fields are used to represent density, energy, and momentum in the direction of the magnetic field; these are supplemented with choices of boundary condition including no-outflow, free-outflow, and a model representing the plasma sheath (a region of positive charge density near a boundary caused by the rapid outflow of electrons), and choices of source term.

A subsequent section deals with particle recycling: ions striking the boundary tend to re-enter the plasma as neutral species after acquiring electrons from the wall, before being ionized again and returning to the wall to re-acquire electrons, a process which can repeat *ad infinitum*. This can be modelled as a source of particles near the boundary, together with an energy sink term to represent losses due to ionization and radiative processes. A more sophisticated approach is to simulate the plasma as two coupled fluids, one charged and one not - this basically means that the 'source' is a time-evolving part of the system state as opposed to being added in by hand. Effective reaction rates are derived from an atomic physics database. Adding neutrals means that a one-dimensional representation becomes less appropriate, due to the absence of magnetic confinement for such a species. Further complexity emerges from the low-collisionality regime that pervades over much of the tokamak volume, with the non-small Knudsen number reflecting the need for a kinetic treatment: this could be implemented using the open-source SOLPS code, or, less readily, obtaining and using the non-open-source EIRENE code.

A final section includes a discussion of multiple species i.e. deuterium, tritium, helium ash, and impurities such as tungsten and beryllium, all of which are potentially important in a full simulation. One simple step is to allow species to have separate electron and ion temperatures; here, quasi-neutrality means that separate electron and ion equations for density and velocity are not necessary (i.e. only the energy balance equation is paired). More generally, it is conceded that the plethora of interacting, interconverting states represented by the various species and their ionization, internal excitation, and combination constitutes a large and currently incompletely-understood problem.

Acknowledgement

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.

References

[1] B. Dudson, P. Hill, E. Higgins, D. Dickinson, S. Wright, and D. Moxey. Task 0.1 Environment. Technical Report 2047356-TN-01, UKAEA Project Neptune, 2021.

[2] B. Dudson, P. Hill, E. Higgins, D. Dickinson, S. Wright, and D. Moxey. Task 1.1 Elliptic solver tests. Technical Report 2047356-TN-02-2, UKAEA Project Neptune, 2021.

[3] B. Dudson, P. Hill, E. Higgins, D. Dickinson, S. Wright, and D. Moxey. Task 1.2 Elliptic solver implementation. Technical Report 2047356-TN-03-2, UKAEA Project Neptune, 2021.

[4] B. Dudson, P. Hill, E. Higgins, D. Dickinson, S. Wright, and D. Moxey. Task 2.1 1D fluid model tests. Technical Report 2047356-TN-04-2, UKAEA Project Neptune, 2021.

[5] B.D. Dudson. BOUT++ website. `https://boutproject.github.io/`, 2020. Accessed: June 2020.

[6] D. Moxey et al. Nektar++ website. `https://www.nektar.info`, 2020. Accessed: June 2020.

UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0057
	Issue:	1.00
	Date:	November 23, 2021
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	Wayne Arter Ed Threlfall BD	N/A N/A	November 23, 2021 November 23, 2021
Reviewed By:	Rob Akers Advanced Computing Dept. Manager		November 23, 2021
Approved By:	Rob Akers Advanced Computing Dept. Manager		November 23, 2021

T/NA083/20
Fluid Referent Models

Task 0.1 Environment

Ben Dudson, Peter Hill, Ed Higgins, David Dickinson, and Steven Wright

University of York

David Moxey

University of Exeter

March 29, 2021

Contents

1 Executive summary 1

2 Development environment 2

2.1 Slack 2

2.2 GitHub Organisation 2

2.3 ReadTheDocs 3

3 Community building 3

3.1 Hackathons 3

3.1.1 VECMA 3

3.2 Workshops 6

3.2.1 Performance Analysis 6

3.2.2 HPC Development using C++ and SYCL 7

3.2.3 Towards Exascale Simulation of Integrated Engineering Systems at Extreme Scales 7

3.3 Cross-task coordination 7

4 Testing 8

4.1 Correctness testing 8

4.2 Performance testing 8

4.2.1 Existing Solutions 10

1 Executive summary

This document describes activities from January to March 2021, to coordinate community activities under the ExCALIBUR-Neptune project. The deliverable for this work package was to set up the development environment (version control, continuous development/integration, automated testing and documentation services, coding standards). To achieve this we have set up development services, organised and attended community activities. The other part of this work package is to set up a testing framework for evaluating parallel scaling on e.g. Archer2 / Viking / Bede. In this period we have carried out a search of existing tools, and created an outline of the Neptune performance testing system design. This system will be implemented in task 0.2 (2021/22).

2 Development environment

2.1 Slack

We set up a Slack (`https://slack.com/intl/en-gb/`) work-space, `excalibur-neptune`, as a discussion and communication tool across all of the groups involved in ExCALIBUR-NEPTUNE. We considered alternatives such as Zulip (`https://zulip.com/`), which has similar features of asynchronous text messaging, group private messages, separate topics or channels, and file sharing. Zulip has some advantages over Slack, most notably it is free to host one's own instance (or rather, "free as in 'puppy'": the product is free, but there are still costs involved in the setup, maintenance and running of the service on one's own infrastructure). The main reason for choosing Slack over Zulip was the degree of familiarity people are likely to have with Slack: Zulip has a different model of conversations, somewhat similar to email, while Slack has a more traditional "chatroom" model. Many researchers are already members of at least one Slack workspace, so there is little incremental cost and a shallow learning curve to joining an additional workspace.

2.2 GitHub Organisation

Working with Research Software Engineers (RSEs) at UKAEA, we set up the GitHub organisation ExCALIBUR-NEPTUNE, creating two repositories:

- `Documents` – private repository archiving non-public documents, such as bid documents and reports.

- `Neptune` – public repository to collate Neptune components and documentation

We are managing access to these repositories as well as to the organisation as whole, inviting members of other groups to become organisation members, giving them permissions to create and manage their own repositories under the organisation. Through this mechanism, as of the start of March 2021, two proxy-apps (minepoch and Nektar-driftwave) are hosted under the ExCALIBUR-NEPTUNE organisation.

2.3 ReadTheDocs

We have set up ReadTheDocs (`https://readthedocs.org/`) to automatically build and host documentation in the `Neptune` repository. ReadTheDocs is built on Sphinx (`https://www.sphinx-doc.org/en/master/`) which reads and parses ReStructuredText (`http://docutils.sourceforge.net/rst.html`) files.

3 Community building

3.1 Hackathons

3.1.1 VECMA

The VECMA Hackathon ran from the 19th to 22nd of January 2021, and served as an introduction to the VECMA toolkit of uncertainty quantification (UQ) tools. We used two BOUT++ models, a simple 1D heat conduction model and a more complicated 2D "blob" model, as bases with which to learn the EasyVVUQ (`https://easyvvuq.readthedocs.io/en/dev/`) tool. We gained experience with using EasyVVUQ for BOUT++, and started understanding some of the challenges UQ is going to present to the ExCALIBUR-NEPTUNE project. During the Hackathon, we had conversations with the UQ group and fed our findings back to them.

In order to use EasyVVUQ with BOUT++, we had to first write a custom "encoder" and "decoder" in Python. The encoder turns a Python `dict` of input values into an input file that BOUT++ can read, while the decoder reads BOUT++ output file(s) into a Python `dict`. This was trivial to implement given BOUT++'s existing pre- and post-processing tools.

We next followed existing EasyVVUQ examples and tutorials to get a basic UQ workflow setup using the simple 1D model. These were easy to follow and to adapt to our model, and were greatly helped by access to EasyVVUQ developers and experts during the hackathon.

The 1D heat conduction model evolves the following equation in time, t:

$$\frac{\partial T}{\partial t} = \nabla_{||}(\chi \partial_{||} T) \tag{1}$$

where T is the temperature and χ is heat conductivity. BOUT++ grids are always 3D, even if some dimensions only have a single point. Here, we use 100 points in y, the parallel direction, and 1 point in both x and z. The initial condition is given by a Gaussian in y:

$$T(t=0) = A \exp[-(y-y_0)^2/(2w^2)]/(w\sqrt{2\pi}) \tag{2}$$

where A is the amplitude, y_0 the Gaussian centre, and w is the width of the Gaussian. Thus, we have four input parameters: χ, A, y_0 and w.

Overall, EasyVVUQ proved easy to do basic UQ and get results out. We initially used Polynomial Chaos Expansion (PCE) with this model, varying just χ and A, and measuring $T(y, t = 10)$. This model takes only a few seconds to run, and using 3rd order PCE resulted in 16 simulations, taking only a couple of minutes total. The PCE tools in EasyVVUQ have built-in tools for plotting the moments and Sobol indices, making simple analyses trivial. However, even using this simple model immediately uncovered some subtleties: it is (currently) not possible to give EasyVVUQ more information about the expected distribution. We know that T must always be positive, but when varying χ over multiple orders of magnitude, some simulations see T very quickly go to zero. The distribution of simulations can be heavily weighted close to zero, and the resulting distribution of T computed by the PCE analysis can have significant amplitude at negative T, which is nonphysical. Similar difficulties are anticipated in any system where uncertainty in an independent or input variable varies over multiple orders of magnitude. One solution is to instead measure $\ln(T)$ instead of T, which enforces the positivity condition, but at the expense of making the resulting uncertainties more difficult to interpret. A more robust solution would be for EasyVVUQ to be able to incorporate additional *a priori* knowledge of the dependent variables.

The "blob2D" model is more complex, having spatial variation in two dimensions (x and z), and two evolving variables, the vorticity ω and the plasma density n:

$$\frac{\partial \omega}{\partial t} = -[\phi, n] + 2\frac{\partial n}{\partial z}\frac{\rho_s}{R_c} + D_n \nabla_\perp^2 n \tag{3}$$

$$\frac{\partial n}{\partial t} = -[\phi, \omega] + 2\frac{\partial n}{\partial z}\frac{\rho_s}{nR_c} + D_\omega \nabla_\perp^2 \omega \frac{1}{n} \tag{4}$$

$$\omega = \nabla^2 \phi \tag{5}$$

where ϕ is the electrostatic potential, $[\cdot, \cdot]$ is the Poisson bracket, $\rho_s = \sqrt{eT_{e0}m_i}/(eB_0/m_i)$ is the Bohm gyro-radius, e the electron charge, T_{e0} the initial electron temperature, m_i the ion mass, B_0 the magnetic field, R_c is the radius of curvature, D_n is the density diffusion coefficient, and D_ω is the viscous diffusion coefficient. This model is significantly more complex than the conduction model with many more input parameters, and takes several minutes to complete 50 timesteps on 16 cores. We used this model to investigate using EasyVVUQ on expensive turbulence models, varying the initial amplitudes of T_{e0}, the scale of the initial density perturbation, n_0, D_n, and D_ω. Because the output of this model is 2D in space, plus time, we used a number of lower-dimensional diagnostics as measurements instead. These consisted of the (x, z) position of the both the peak density and its centre of mass, as well as the velocity of this point.

The first thing to note is that the PCE sampler requires $(N + 1)^M$ samples, where N is the order of polynomials used, and M is the number of parameters being varied. For our model where we are varying $M = 4$ parameters and using $N = 3$ we need 256 simulations. This is prohibitive on a local machine, but EasyVVUQ includes several mechanisms for running simulations in parallel, including on clusters with job/queue managers such as SLURM. One mechanism is via `dask` (`https://dask.org/`), which works best for simple parallelisation on a local machine, and especially for Python kernels. The `dask_jobqueue` (`https://jobqueue.dask.org/en/latest/`) package extends this to SLURM clusters, but this proved to be very difficult to use for MPI parallelised programs. The last mechanism, `ExecuteSLURM`, takes a template SLURM job script and replaces variables with concrete values, and offers some control over the number of jobs to submit at once. This turned out to the be the most robust of the three mechanisms tried, and although it did not offer much benefit over a hand-written parameter scan, it is likely to be of more use when using hierarchical sparse grid sampling, where the parameter scan is incrementally refined. One thing to note is that while EasyVVUQ can launch simulations in parallel, the decoding – reading the output of the simulations – still happens in serial. Therefore if there is any cost to the decoding, it is probably wise to have this done as part of the simulation.

Lastly, we also explored using stochastic collocation (SC) instead of PCE. There was some pain to this. While the EasyVVUQ sampling and analysis objects are easily swapped between these two methods, the later analysis and plotting of results differ significantly. This means that a workflow written for one method needs several changes in order to convert it to use the other. Mostly these differences are due to incomplete or unimplemented methods, and it is expected that these differences disappear as EasyVVUQ matures.

3.2 Workshops

3.2.1 Performance Analysis

This ExCALIBUR-affiliated performance analysis workshop (`https://tinyurl.com/performanceanalysis2021`) held online workshops on the 21st 22nd of January, and 18th Feb.

Most of this work has been done by Joseph Parker and John Omotani (CCFE), with input and discussions from the BOUT++ team at York and elsewhere. BOUT++, and in particular the STORM model, has been instrumented with a number of tools including Intel VTune and Score-P. The results of these workshops are collected in a github repository (`https://github.com/boutproject/cs-performance-tuning-workshop`).

3.2.2 HPC Development using C++ and SYCL

SYCL is one of the potential technologies for developing performance portable software under ExCALIBUR-NEPTUNE. We therefore joined workshops organised by Codeplay on 7th Jan and 17th Feb 2021. This included tutorials, exercises, and help from Codeplay to install and use SYCL compilers.

3.2.3 Towards Exascale Simulation of Integrated Engineering Systems at Extreme Scales

21 – 22 January, 2021: This was an ExCALIBUR meeting, at which Ben Dudson gave a talk "Coupling Codes at Exascale for the ExCALIBUR UKAEA NEPTUNE Nuclear Fusion Project". That talk presented an overview of the challenges, some representative examples of integrated simulations and code coupling in fusion, and invited participation in and input into the Neptune project.

3.3 Cross-task coordination

We have given talks at the main Neptune events, including the kickoff meeting 14th Jan, and wrap-up on 16th March, and regular progress update meetings.

Regular meetings have also been held with Oxford/Warwick group, and the STFC preconditioners group. Separate meetings have also been held with groups to discuss PinT and UQ bids and activities, to consider how these fit into the Neptune work plan. We have also participated in and in some cases led discussions in the ExCALIBUR-Neptune Slack workspace, to coordinate with the other Neptune tasks.

4 Testing

4.1 Correctness testing

Since Github is used to host the Excalibur-Neptune code, we propose to use Github actions for correctness and regression testing, as well as enforcement of coding styles and simple static code analysis. These tests can be triggered on pushes or pull requests, and can be configured to block merging into main branches unless passed.

The approach used by Nektar++ appears promising, in which Docker images are built and then used to run the test suite. If a test fails, this means that the same docker image can be downloaded and run on a developer's machine. This helps reproduce and identify errors, which might otherwise only occur on the testing server.

Github actions runs on virtual servers, with typically one or two cores, and inconsistent performance. This makes it unsuitable for performance testing, for which a bespoke solution is being developed, described in the next section.

4.2 Performance testing

We have started writing the specification for a system for monitoring performance of ExCALIBUR-NEPTUNE components and proxy-/mini-apps (hereafter collectively "apps"). Some requirements:

- Can be run manually, but amenable to automation: we want to be able to track the performance history of a given app, while still maintaining the flexibility to run *ad hoc* experiments across different app and hardware configurations;
- Flexible output: the performance metrics that we want to measure and track may change over time or between apps or experiments. We don't want to define a rigid schema now only to need to continually change it later;
- App agnostic: ExCALIBUR-NEPTUNE will be made of many compo-

nents, with many proxy-apps developed along the way, and making use of a variety of performance profiling tools. We want a single performance testing framework that can handle all of this variation;

In order to satisfy the first requirement, we have chosen to start developing a "push" framework, where data is collected on a machine and pushed to the data repository, rather than a "pull" framework where a server launches jobs on remote machines and pulls the data. This leaves open the option of automating the performance testing and converting the "push" framework (at least partly) into a "pull" one.

Our proposed framework consists of several components:

- *Test configuration files*: define an individual run of an app
- *Runner*: reads *test configuration files* and runs an app
- *Performance data files*: output from an individual run of an app
- *Uploader*: collates performance data files and uploads them to the data repository
- *Data repository*: stores performance data files
- *Dashboard*: interprets and displays data from data repository

Test configuration files need to be human readable, as these will be written by humans. This rules out formats such as JSON, which are suitable for machine-machine transfer of data, but are not human-friendly. There are a variety of text file formats that would be suitable; we are currently looking at TOML (https://toml.io/en/). The schema of these configuration files is still a work in progress, but there are several requirements:

- App executable location
- App input file(s) location(s)
- Performance tool (optional)
- *Performance data file* output location

The runner would read the test configuration files, and launch the app, possibly wrapped or instrumented with a separate performance profiling tool. This set up would allow automatic scanning for configuration files, and so expanding the performance test suite could be done through simply adding a new file.

The performance data files should be structured text files of some form, most likely JSON, to facilitate interoperability. This gives us the most flexibility in terms of moving to more rigid schemas later on or databases.

The data repository will be a plain GitHub repository. The uploader can then be a simple wrapper around git.

4.2.1 Existing Solutions

TheMatrix (https://github.com/devitocodes/thematrix) is a similar project for the Devito (https://www.devitoproject.org) symbolic finite difference library. TheMatrix runs performance benchmarks of Devito on a variety of hardware hosted in the Azure cloud service, and uses Airspeed Velocity (https://asv.readthedocs.io/en/stable/) to visualise the results. While meeting several of the ExCALIBUR-NEPTUNE requirements, there are a few major downsides. Firstly, Airspeed Velocity is limited to profiling Python tools/kernels only. This is essentially a show-stopper for ExCALIBUR-NEPTUNE in terms of being able to monitor the performance of proxy-apps written in several different languages. The other significant point is that TheMatrix is built around running the tests on the Azure platform, which does not currently meet the needs of ExCALIBUR-NEPTUNE.

Another similar project is Gingko Performance Explorer (GPE) (https://ginkgo-project.github.io/gpe/), another performance monitoring solution tied to a particular numerical package, Gingko (https://ginkgo-project.github.io). GPE is designed to be run automatically as part of a Continuous Integration/Continuous Development (CI/CD) process. After a commit to the development branch of the Gingko project, a GitHub Actions runner starts GPE, which pushes jobs to HPC systems, where tests are run and the performance is measured. GPE then periodically "checks in" to the HPC to see if the jobs have finished, and if so, collates the results. A particularly interesting feature of GPE is the data visualisation, which allows custom queries to be written and visualised directly in a web browser.

From an initial survey it seems that there is no existing solution which meets all the needs of ExCALIBUR-NEPTUNE. Partial solutions exist, and aspects of these will be adopted, to inform the design of the Excalibur-Neptune system. Since we are designing this to be a generic tool, it is likely to also be of interest to a wider community.

Excalibur-Neptune report 2047356-TN-02-2

Task 1.1 Elliptic solver tests

Ben Dudson, Peter Hill, Ed Higgins, David Dickinson, and Steven Wright

University of York

David Moxey

University of Exeter

June 16, 2021

Contents

1 Executive summary **2**

2 Elliptic solvers in plasma equations **2**

2.1 Embedded domain 4

3 Tests of elliptic solvers **5**

3.1 Correctness tests 5

3.2 Performance tests 6

4 Tests in slab geometries **7**

4.1 Slab with a pole 8

5 Tests in tokamak geometries **8**

5.1 Tokamak geometries 10

5.1.1 Numerical tokamak equilibria 10

5.1.2 Analytic tokamak equilibria 10

6 Time-evolving systems **11**

6.1 Shear Alfvèn wave 11

6.1.1 Energy conservation 12

6.2 Geodesic Acoustic Wave 14

6.2.1 Simplified model 14

6.2.2 Analytic solution 16

6.2.3 Mass and energy conservation 17

7 Conclusions **19**

8 References **20**

1 Executive summary

This report gives a brief overview of the origin of elliptic problems in plasma physics models, and the tokamak geometry they are solved in. Drawing mainly on experience with BOUT++, approaches to testing of both correctness and performance, and the pros and cons of them are discussed. A series of geometries which can be used for testing are described, starting with slabs of increasing complexity, and then tokamak geometries. Finally two simple time-dependent sets of equations are suggested, one for the shear Alfvèn wave, and the other the Geodesic Acoustic Wave (GAM). These provide ways to test the long-term numerical stability of the elliptic solver, together with the boundary conditions parallel and perpendicular to the magnetic field.

2 Elliptic solvers in plasma equations

Elliptic problems appear in plasma equations typically in the electromagnetic fields, in the limit where the displacement current is neglected and light speed goes to infinity. Two common components are solving for the electrostatic potential ϕ, and the parallel component of the vector potential $A_{||} = \mathbf{b} \cdot \mathbf{A}$ where $\mathbf{b} = \mathbf{B}/\left|\mathbf{B}\right|$ is the unit vector along the magnetic field $\mathbf{B}$.

The electrostatic potential is calculated from a fluid vorticity ω, or analogously in gyro-fluid and gyro-kinetic models, from ion and gyro-centre density differences. These give rise to an equation of the form

$$\nabla \cdot \left(\frac{n}{B^2}\nabla_\perp \phi\right) = \omega \tag{1}$$

where n is the plasma density, which varies in time and space. This system is often simplified by replacing n by a constant. By analogy with buoyancy-driven flows, this is called the Boussinesq approximation. The operator $\nabla_\perp = \nabla - \mathbf{b}\mathbf{b}\cdot\nabla$ is the component of the gradient perpendicular to the magnetic field. It arises because the ion polarisation drift, which ultimately gives rise to ω, depends on the electric field perpendicular to the magnetic field.

The electric field parallel to the magnetic field is determined by quite different physics, being mainly determined by the electron dynamics rather than the ions. The electromagnetic potential $A_{||} = \mathbf{b}\cdot\mathbf{A}$ is related to the current along the magnetic field:

$$\begin{aligned}
\mathbf{B} &= \nabla\times\mathbf{A} \\
\mathbf{J} &= \frac{1}{\mu_0}\nabla\times\mathbf{B} \\
&= \frac{1}{\mu_0}\nabla\times\nabla\times\mathbf{A} \\
&= \frac{1}{\mu_0}\left[\nabla\left(\nabla\cdot\mathbf{A}\right) - \nabla^2\mathbf{A}\right]
\end{aligned}$$

The Coulomb gauge is chosen, setting $\nabla\cdot\mathbf{A} = 0$. The current along the magnetic field is therefore

$$J_{||} = \mathbf{b}\cdot\mathbf{J} = -\frac{1}{\mu_0}\mathbf{b}\cdot\nabla^2\mathbf{A} \tag{2}$$

This elliptic operator is slightly different from equation 1 above, but often the following approximation is used:

$$\mathbf{b}\cdot\nabla^2\mathbf{A} \simeq \nabla\cdot\left(\nabla A_{||}\right) \tag{3}$$

In addition derivatives of $A_{||}$ along the magnetic field are often neglected, so that the same operator as equation 1 can be used.

The potential $A_{||}$ appears in models through the perturbed magnetic field

$$\delta\mathbf{B} = \nabla\times\left(\mathbf{b}A_{||}\right) \tag{4}$$

It also appears in the component of the electric field parallel to the magnetic field:

$$E_{||} = \mathbf{b}\cdot\mathbf{E} = -\mathbf{b}\cdot\nabla\phi - \frac{\partial A_{||}}{\partial t} \tag{5}$$

which appears in an Ohm's law equation for the current along the magnetic field.

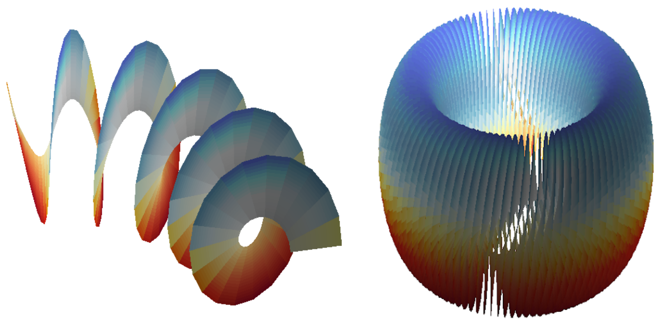

Figure 1: Domain perpendicular to the magnetic field in a torus. For illustration only. Left: A part of the domain; Right: Space filling after many toroidal turns.

2.1 Embedded domain

The elliptic equation to be solved for ϕ in equation 1 may appear to be a 2D problem, because it depends on the component of the electric field perpendicular to the magnetic field. Indeed if the magnetic field were constant in space then this would be a 2D system. In a tokamak however the magnetic field varies in space, and generally has components in both toroidal and poloidal directions. This means that the 2D domain does not in general close on itself, but forms a spiral which fills the toroidal volume. This is illustrated in figure 1. In many situations the tilt of the 2D plane in the toroidal direction can be neglected, using the fact that variation along the magnetic field (which is predominantly toroidal) is generally small. Some important exceptions are:

- Low-n (toroidal mode number) instabilities and waves. For these the assumption that parallel gradients are small relative to perpendicular can break down, and the corrections become important.
- In low aspect-ratio ("spherical") tokamaks, the pitch of the magnetic field can become quite large: At the outboard midplane of MAST and MAST-U, the pitch can be nearly 45^o.

In either of these cases solving for the potential may require a 3D solve, rather than a decoupled set of 2D solves. It seems likely that in most cases the corrections should be small, and that 2D solves would be a good preconditioner for solving the full 3D problem.

3 Tests of elliptic solvers

As for many components of scientific high performance codes, tests need to address both the correctness and performance of the implementation.

3.1 Correctness tests

In all correctness tests it is important to define a figure of merit. This should be quantitative, and sufficiently well defined that it can be automated. This enables a pass/fail criterion to be established, and the test integrated into continuous integration workflows.

It is usually useful to combine both a measure of a global average error (such as root-mean-square, l_2 norm), and a measure which is sensitive to large localised errors (such as maximum absolute error, l_∞ norm). This enables both the overall accuracy of the scheme to be assessed, and also helps identify problems with specific areas of the mesh such as boundaries.

Round-trip tests: The simplest tests to implement are those which use a forward operator to check the accuracy of an inversion method. This type of test is often fast, and useful as check while developing the code. There are some subtleties in this which can lead to spurious issues: The forward operator must use the exact same discretisation as the inverse operator being tested. The forward operator itself should also be checked for correctness. Manual inspection of a forward operator is usually more straightforward than an inverse operator, but is not a substitute for an actual quantitative test.

Analytic solution tests: In simple geometries (such as slabs), and for particular choices of the density n in equation 1, an analytic solution can be found. In these cases the numerical solution can be compared against the analytic, to calculate the error. To properly check the method, a convergence test should be performed, using multiple grids with different resolutions, and the order of convergence compared against the theoretical order of the method used. Testing the order of convergence in this way can be challenging in some cases: Depending on the system, it can be that numerical round-off begins to affect the error, before the error enters the asymptotic regime.

Manufactured solution tests: Simple analytic tests have the benefit of being easy to implement, but are typically limited in their thoroughness: In a slab, for example, many metric tensor components are zero which in a realistic simulation would be non-zero. In that case the slab test is not able to tell whether the parts of the code which depend on those metric components are implemented correctly. The challenge for testing is that realistic cases which exercise all parts of the code typically do not have analytic solutions (otherwise there would be little point in using the code). Fortunately since the system to be tested is relatively straightforward, an analytic solution can be chosen (manufactured) for ϕ, differentiated analytically to calculate the ω function. Values for ω can then be calculated on a grid, inverted and compared to the original expression. As part of this, a non-trivial geometry needs to be generated analytically. This need not necessarily be of the same form as real simulations (e.g. a tokamak), provided that it exercises the same parts of the code (preferably, all the code).

3.2 Performance tests

The elliptic solver is often the main bottleneck to parallel scaling of plasma physics codes, certainly in existing codes such as BOUT++. This is because it involves global communications, or at least communications over a large region of the domain, and this inversion is done frequently as part of the time advance.

Elliptic solvers are critical to many areas of scientific computing, certainly not unique to plasma physics. Experience with BOUT++ however, has been that the plasma use case is different from others in ways which are important for performance: Libraries are often optimised for solving very large systems of equations (millions of d.o.f), for applications where a relatively small number of such large systems may need to be solved. In typical plasma turbulence simulations, however, the size of each system may be relatively small (10^4 d.o.f for a 2D system; 100s of degrees of freedom per solve if decomposed into 1D systems), but a turbulence simulation may require 10^5 or 10^6 such solves. Since the systems to be solved are part of the time evolution, these 10^5 or 10^6 solves are essentially serial unless parallel-in-time techniques are used, and must be solved efficiently. Since the quantity being solved for is evolving in time, solutions tend to be close to previous solutions, and so iterative schemes tend to be effective.

It is more important that performance tests use problems which are close to those which will be solved in production systems, than it is for correctness tests. Different problem sizes hit different thresholds in cache sizes, message sizes, work intensity, network capacity etc. Problems in slab geometry, which are simpler to solve than realistic situations, may have different convergence characteristics, and not exercise the preconditioner, changing the proportion of time spent in parts of the solver. As discussed above, a characteristic of typical problems is that they are solving systems which start from a generally good initial guess, the value at the previous time step. This should be taken into account in the performance testing.

Specific solution algorithms, their scaling, theoretical and measured performance are not discussed in this report beyond the comments above. These will be addressed in task 1.2 and 1.3.

4 Tests in slab geometries

A series of slab simulations of gradually increasing complexity can be used in build a code up step-by-step, and help to identify problems early before moving on to more complex systems.

- A 2D domain, with fixed (Dirichlet) boundary conditions on all sides, and the magnetic field directed perpendicular to the domain. This is a simple problem which is probably included as an example in most finite difference or finite element packages.
- The magnetic field vector **b** is not perpendicular to the plane the potential ϕ is being solved on. This introduces some non-zero metric terms, and can be made more challenging by varying the angle of the magnetic field with position.
- The boundaries can be modified so that the mesh is periodic in one direction, but continues to have boundaries in the other. In a tokamak the periodic direction would correspond to the poloidal or toroidal directions (depending on the coordinate system chosen), and the other direction to the radial direction from inside (core) to outside (edge).

- A variation on the above is to introduce a corner into the problem, so that the domain is periodic over part of the boundary (in the core), but has fixed boundaries over the rest of the boundary (the scrape-off layer).

4.1 Slab with a pole

It was found in developing a BOUT++ implementation, that the above slab geometries were not sufficiently challenging to be able to exercise a GPU preconditioner (Hypre). To make the problem more challenging, and closer to realistic tokamak geometry, a test case was developed which has a pole in the coordinate system close to (but outside) the mesh edge. This mimics the singularity at the X-point which occurs in field-aligned coordinates.

In field-aligned coordinates one of the coordinates is aligned with the magnetic field. In this case the y coordinate basis vector $\mathbf{e}_y$ is aligned to $\mathbf{B}$, so therefore the gradients of the x and z coordinates are perpendicular to $\mathbf{B}$. A choice used in BOUT++ and other plasma simulation codes is a Clebsch coordinate system, characterised by $\mathbf{B} = \nabla z \times \nabla x$. For details see the BOUT++ manual section on field-aligned coordinates [1].

The radius from a pole, r is used to set non-zero metric tensor components

$$\begin{aligned} g_{xx} &= \mathbf{e}_x \cdot \mathbf{e}_x = 1/r^2 & (6)\\ g_{yy} &= \mathbf{e}_y \cdot \mathbf{e}_y = 1 + 1/r^2 & (7)\\ g_{zz} &= \mathbf{e}_z \cdot \mathbf{e}_z = 1 & (8)\\ g_{yz} &= \mathbf{e}_y \cdot \mathbf{e}_z = 1/r & (9) \end{aligned}$$

and so coordinate Jacobian $J = 1/r$.

In BOUT++ the input file for this configuration is shown in figure 2.

5 Tests in tokamak geometries

These tests approach realistic production cases, introducing a toroidal geometry with sheared magnetic field in doubly-periodic domains, and then by introducing

```
# Mesh size
Lx = 10
Ly = 10

# Number of grid cells
nx = 20
ny = 32

# mesh spacing
dx = Lx / (nx - 4) # Account for 4 guard cells in X
dy = Ly / ny

# Location of the pole in the coordinates
pole_x = Lx + 1.0
pole_y = Ly + 1.0

# Distance from the pole
# Note here "x" is normalised to [0,1] on the grid; "y" normalised
    to [0,2*pi]
r = sqrt((pole_x - x * Lx)^2 + (pole_y - y * Ly / (2*pi))^2)

# This mimics the metric tensor close to the X-point in a tokamak
# by here setting poloidal field Bp ~ r

g11 = r^2
g22 = 1
g33 = 1 + 1 / r^2
g12 = 0
g13 = 0
g23 = -1/r

J = 1 / r

g_11 = 1 / r^2
g_22 = 1 + 1 / r^2
g_33 = 1
g_12 = 0
g_13 = 0
g_23 = 1 / r
```

Figure 2: Part of a BOUT++ input file for a slab with a pole. See BOUT++ manual for details of the code [2].

an X-point into the domain. For each geometry two kinds of tests can be performed:

- A single solve (for correctness), or small number of solves with slowly varying input (for performance).
- Time-evolving a relatively simple system of equations, in which the elliptic solve is a key part. The purpose of this is to identify potential issues with

slowly accumulating errors. These errors may be below tolerance in a single solve, but interact with the time evolving system in a way which leads to numerical instability.

5.1 Tokamak geometries

Tokamak grids are typically generated from a numerical equilibrium, and typically those are provided from experiment at low resolution (e.g. 64x64 for the whole poloidal cross-section). Higher resolution inputs can be generated using a free boundary Grad-Shafranov solver, but generating a sequence of simulation meshes from Grad-Shafranov solutions for a convergence test remains challenging. Not conceptually difficult, but the capability to do it has not been implemented and would require some considerable effort to do in a rigorous way.

For convergence tests, the easiest approach would seem to be to use an analytic solution, either to the Grad-Shafranov solution itself, or a simplified solution which is not a Grad-Shafranov solution but shares important characteristics (such as an X-point).

5.1.1 Numerical tokamak equilibria

If an analytic equilibrium and convergence to small tolerances is not required, then numerical solutions are available for many different tokamaks, both real and conceptual. A common starting point is circular cross-section geometries, which don't have an X-point. Examples include the 'cbm18' series of equilibria generated by P.Snyder (GA). Alternatively an equilibrium can be generated by a free-boundary equilibrium code such as EFIT, EFIT++, Helena, or FreeGS[3].

5.1.2 Analytic tokamak equilibria

The Grad-Shafranov equation is a nonlinear partial differential equation, and so finding analytic solutions is non-trivial. Some have however been found: The Soloviev solutions are widely used, but only include closed flux surfaces (the plasma core), not the separatrix and scrape-off layer. In addition these solutions typically have jumps or current sheets at the edge which make them problematic to extend into the vacuum

Cerfon & Freidberg found a set of analytic equilibria which include a separatrix and vacuum region outside the plasma [4]. That algorithm has been implemented in Python by John Omotani [5]. This could be used as input to a mesh generator for convergence studies.

6 Time-evolving systems

These are simple systems of equations which are intended to be relatively quick to simulate, but which can help identify problems with numerical methods or boundary conditions at an early stage.

Since we wish to identify numerical instabilities and error accumulation, we choose systems which are stable: These systems of equations support waves which are either stable or damped. Any growing mode can therefore be identified as a numerical artefact.

6.1 Shear Alfvèn wave

This is an electromagnetic wave along the magnetic field. Due to the variation of the magnetic field in a tokamak, there is a radial (across the field) phase velocity, and an initially coherent mode will tend to mix and dissipate due to the model and numerical damping.

The set of equations evolves the vorticity U and parallel vector potential $A_{||}$. It appears in normalised form as:

$$\begin{aligned}
\frac{\partial U}{\partial t} &= \nabla_{||} j_{||} && (10)\\
\frac{\partial A_{||}}{\partial t} &= -\partial_{||}\phi - \eta j_{||} + \mu_{||e}\nabla_{||}^2 j_{||} && (11)\\
j_{||} &= -\frac{2}{\beta_e}\nabla_\perp^2 A_{||} && (12)\\
\nabla_\perp^2\phi &= U && (13)
\end{aligned}$$

where $\nabla_{||} f = \nabla\cdot\left(\mathbf{b}f\right)$ is the divergence of a flow along the magnetic field, and $\partial_{||} = \mathbf{b}\cdot\nabla$ is the gradient along the magnetic field. The factor β_e is the ratio of electron pressure to magnetic pressure, and is typically around 10^{-4} in the plasma edge. Dissipation is included in the form of resistivity η and parallel electron viscosity $\mu_{||e}$. The resistivity would typically be small ($< 10^{-2}$), and electron viscosity much smaller than that (usually neglected).

6.1.1 Energy conservation

The equations for this wave contain only energy sinks (dissipation), and so oscillations can only grow if there are sources of energy from either numerical instability or boundary fluxes. To calculate the energy in the system, multiply the vorticity equation by ϕ and the $A_{||}$ equation by $j_{||}$, then integrate over the simulation volume.

$$\begin{aligned}
\phi U &= \phi\nabla\cdot\nabla_\perp\phi = \nabla\cdot\left(\phi\nabla_\perp\phi\right) - \underbrace{\nabla\phi\cdot\nabla_\perp\phi}_{\left|\nabla_\perp\phi\right|^2} && (14)\\
\frac{\partial}{\partial t}\left(\phi U\right) &= \phi\frac{\partial U}{\partial t} + U\frac{\partial \phi}{\partial t} && (15)\\
&= \phi\nabla_{||}j_{||} + \nabla_\perp^2\phi\frac{\partial\phi}{\partial t} && (16)\\
\nabla_\perp^2\phi\frac{\partial\phi}{\partial t} &= \nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla_\perp\phi\right) - \underbrace{\nabla\frac{\partial\phi}{\partial t}\cdot\nabla_\perp\phi}_{\frac{1}{2}\frac{\partial}{\partial t}\left(\left|\nabla_\perp\phi\right|^2\right)} && (17)
\end{aligned}$$

and so

$$\frac{\partial}{\partial t}\left[\nabla\cdot\left(\phi\nabla_\perp\phi\right) - \left|\nabla_\perp\phi\right|^2\right] = \phi\nabla_{||}j_{||} + \nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla_\perp\phi\right) - \frac{1}{2}\frac{\partial}{\partial t}\left(\left|\nabla_\perp\phi\right|^2\right) \quad (18)$$

Putting these together gives an equation for the evolution of the kinetic energy in the E×B motion:

$$\frac{1}{2}\frac{\partial}{\partial t}\left(\left|\nabla_\perp\phi\right|^2\right) = -\phi\nabla_{||}j_{||} + \nabla\cdot\left(\phi\nabla_\perp\frac{\partial\phi}{\partial t}\right) \quad (19)$$

On the left is the energy in the E×B motion, since the factors of ion mass and density are constant here, and have been normalised out. The first term on the right is a transfer of energy from electromagnetic energy, and the second term is a flux of energy from the boundary. Integrating this equation over a volume, the divergence term on the right will become a surface integral over the boundary:

$$\frac{\partial}{\partial t}\int_V \frac{1}{2}\left|\nabla_\perp\phi\right|^2 dV = -\int_V \phi\nabla_{||}j_{||}dV + \oint_S \phi\nabla_\perp\frac{\partial\phi}{\partial t}\cdot d\mathbf{S} \tag{20}$$

and so the flux of energy through the boundary is zero if ϕ is zero, or if the component of $\nabla_\perp\phi$ normal to the boundary is constant in time.

Similarly, the $A_{||}$ equation gives:

$$\begin{aligned}
j_{||}A_{||} &= \frac{2}{\beta_e}A_{||}\nabla_\perp^2 A_{||} = \frac{2}{\beta_e}\nabla\cdot\left(A_{||}\nabla_\perp A_{||}\right) - \frac{2}{\beta_e}\underbrace{\nabla A_{||}\cdot\nabla_\perp A_{||}}_{\left|\nabla_\perp A_{||}\right|^2} && (21)\\
\frac{\partial}{\partial t}\left(j_{||}A_{||}\right) &= j_{||}\frac{\partial A_{||}}{\partial t} + A_{||}\frac{\partial j_{||}}{\partial t} && (22)\\
&= j_{||}\left(\partial_{||}\phi + \eta j_{||}\right) + \frac{2}{\beta_e}A_{||}\nabla\cdot\left(\nabla_\perp\frac{\partial A_{||}}{\partial t}\right) && (23)\\
&= j_{||}\partial_{||}\phi + \eta j_{||}^2 + \frac{2}{\beta_e}\nabla\cdot\left(A_{||}\nabla_\perp\frac{\partial A_{||}}{\partial t}\right) - \frac{2}{\beta_e}\frac{1}{2}\frac{\partial}{\partial t}\left|\nabla_\perp A_{||}\right|^2 && (24)
\end{aligned}$$

and so:

$$\frac{\partial}{\partial t}\left[\frac{2}{\beta_e}\nabla\cdot\left(A_{||}\nabla_\perp A_{||}\right) - \frac{2}{\beta_e}\left|\nabla_\perp A_{||}\right|^2\right] = j_{||}\partial_{||}\phi + \eta j_{||}^2 + \frac{2}{\beta_e}\nabla\cdot\left(A_{||}\nabla_\perp\frac{\partial A_{||}}{\partial t}\right) - \frac{2}{\beta_e}\frac{1}{2}\frac{\partial}{\partial t}\left|\nabla_\perp A_{||}\right|^2 \tag{25}$$

$$\frac{1}{\beta_e}\frac{\partial}{\partial t}\left|\nabla_\perp A_{||}\right|^2 = -j_{||}\partial_{||}\phi - \eta j_{||}^2 + \frac{2}{\beta_e}\nabla\cdot\left(\frac{\partial A_{||}}{\partial t}\nabla_\perp A_{||}\right) \tag{26}$$

Integrating over the volume:

$$\frac{\partial}{\partial t}\int_V \frac{1}{\beta_e}\left|\nabla_\perp A_{||}\right|^2 dV = -\int_V j_{||}\partial_{||}\phi dV - \int_V \eta j_{||}^2 dV + \oint_S \frac{2}{\beta_e}\frac{\partial A_{||}}{\partial t}\nabla_\perp A_{||}\cdot d\mathbf{S} \tag{27}$$

Like the E×B energy equation (eq 20), the flux of energy through the boundary is zero if $A_{||} = \text{const}$ or $\nabla_\perp A_{||}\cdot\mathbf{S} = 0$.

The exchange of energy between E×B and electromagnetic forms is through

$\phi\nabla_{||}j_{||}$ and $j_{||}\partial_{||}\phi$. These should balance for energy to be conserved:

$$\phi\nabla_{||}j_{||} = \phi\nabla\cdot\left(\mathbf{b}j_{||}\right) = \nabla\cdot\left(\mathbf{b}\phi j_{||}\right) - j_{||}\underbrace{\mathbf{b}\cdot\nabla\phi}_{\partial_{||}\phi} \tag{28}$$

Hence there are no fluxes of energy through the parallel boundary if $j_{||}$ or ϕ are zero at the boundary.

From this we conclude:

- $\phi = 0$ or $\nabla_\perp\phi\cdot\mathbf{S} = \text{const}$ for conservation of energy
- $A_{||} = \text{const}$ or $\nabla_\perp A_{||}\cdot\mathbf{S} = 0$ for conservation of energy
- There is no requirement that the form of $\nabla^2_\perp$ in vorticity and $j_{||}$ equations is the same, since the only term which is common to both equations is $\phi\nabla_{||}j_{||} \leftrightarrow j_{||}\partial_{||}\phi$

6.2 Geodesic Acoustic Wave

The Geodesic Acoustic Mode (GAM) is an axisymmetric ($n = 0$) sound wave which occurs through oscillations in the radial current. It involves an up-down ($m = 1$) asymmetry in the density, together with a potential which is approximately constant on flux surfaces ($m = 0$). It can arise from a simple system of equations, but because it involves currents across the magnetic field, it exercises the radial boundary conditions. If an annulus is simulated, so that there is a boundary to the inner core, then treatment of that core boundary can be important, to ensure that there is no net current, or that radial boundary layers don't form.

6.2.1 Simplified model

The GAM oscillation arises because radial gradients of zonal ($n = 0$, $m \simeq 0$) electrostatic potential causes E$\times$B advection of density in the poloidal direction. Because the magnetic field strength varies between inboard and outboard sides, E$\times$B advection is faster on the outboard side than the inboard. This leads to rarefaction and compression at the top and bottom of the device (depending on flow direction). The change in pressure at top and bottom of the device alters the diamagnetic current across the magnetic field, which then modifies the electrostatic potential.

We require equations for current continuity and density n:

$$\begin{aligned}\nabla\cdot\left[\frac{m_in}{B^2}\frac{\partial\nabla_\perp\phi}{\partial t}\right] &= \nabla\cdot\left[p\nabla\times\left(\frac{\mathbf{b}}{B}\right)\right]+\nabla_{||}j_{||} &(29)\\ \frac{\partial n}{\partial t} &= -\nabla\cdot\left[n\frac{\mathbf{b}\times\nabla\phi}{B}\right] &(30)\end{aligned}$$

The first (vorticity) equation includes a current along the magnetic field, $j_{||}$. This is not required to derive the wave analytically, but is required in a numerical simulation to ensure that ϕ remains approximately constant along the magnetic field. For this purpose a simple electrostatic Ohm's law can be used:

$$j_{||} = -\frac{1}{\eta}\mathbf{b}\cdot\nabla\phi \tag{31}$$

Decreasing η will make the potential relax more quickly to a constant along flux surfaces.

The usual approach to solving this is to use

$$\nabla\times\frac{\mathbf{b}}{B}\simeq\frac{2}{B}\mathbf{b}\times\kappa$$

and to split the $E\times B$ advection term into a divergence-free advection term, and a divergence term:

$$\frac{\partial n}{\partial t} = -\frac{1}{B}\mathbf{b}\times\nabla\phi\cdot\nabla n - n\nabla\cdot\left(\frac{1}{B}\mathbf{b}\times\nabla\phi\right) \tag{32}$$

then approximate

$$\nabla\cdot\left(\frac{1}{B}\mathbf{b}\times\nabla\phi\right)\simeq\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla\phi \tag{33}$$

It is usual to neglect the poloidal derivative (y) terms in the $E\times B$ advection operator. In Clebsch coordinates this term looks like

$$\frac{1}{B}\mathbf{b}\times\nabla\phi\cdot\nabla n = \frac{\partial\phi}{\partial x}\frac{\partial\phi}{\partial z}-\frac{\partial\phi}{\partial z}\frac{\partial\phi}{\partial x}$$

For axisymmetric flows the z derivatives are zero, so this term vanishes, leaving only the compression term. This leaves a minimal GAM model:

$$\begin{aligned}\frac{\partial\Omega}{\partial t} &= \frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla p + \nabla_{||}j_{||} && (34)\\ \frac{\partial n}{\partial t} &= -n\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla\phi && (35)\\ \Omega &= \frac{m_in_0}{B_0^2}\nabla_\perp^2\phi && (36)\end{aligned}$$

where n_0 and B_0 are constants in space and time, and an isothermal approximation is used here:

$$p = enT_0 \qquad (37)$$

Note also that here ϕ is assumed to be approximately constant on flux surfaces, which will need to be enforced numerically using an Ohm's law.

6.2.2 Analytic solution

Starting with the density equation, we look for solutions of the form

$$n\left(x,\theta,t\right) = \hat{n}\left(\theta\right)e^{ikx-iwt}$$

and potential ϕ:

$$\phi\left(x,\theta,t\right) = \hat{\phi}e^{ikx-iwt}$$

Linearising equation 35, assuming a simple circular cross-section, large aspect-ratio, and keeping only the poloidal flow

$$\mathbf{b}\times\kappa\cdot\nabla \rightarrow \frac{1}{R}\sin\theta\frac{\partial}{\partial x} \qquad (38)$$

we get

$$\hat{n} = n_0\frac{2k}{BR\omega}\sin\theta\hat{\phi} \qquad (39)$$

Hence if we assume $\hat{\phi}$ is independent of θ then $\hat{n}$ has a $\sin\theta$ dependence.

The parallel current term will act to equalise potential over a flux surface, but provided this occurs sufficiently rapidly we can remove it from the analysis and assume that ϕ is constant on flux surfaces. To remove the parallel current term from the vorticity equation, average over a flux surface by defining $\langle\cdot\rangle$ so that $\left\langle\nabla_{||}\right\rangle = 0$. For large aspect ratio:

$$\langle\cdot\rangle = \oint \cdot d\theta$$

This gives:

$$\begin{aligned}
\frac{\partial}{\partial t}\langle\Omega\rangle &= \left\langle \frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla p\right\rangle && (40)\\
-i\omega\frac{m_i n_0}{B^2}\left(-k^2\right)\left\langle\hat{\phi}\right\rangle &= \left\langle 2\frac{eT_0}{BR}\sin\theta\left(ik\right)\hat{n}\right\rangle && (41)\\
\omega k^2\frac{m_i n_0}{B^2} &= \left\langle 4\frac{k^2 e n_0 T_0}{\omega B^2 R^2}\sin^2\theta\right\rangle && (42)
\end{aligned}$$

Most terms cancel, leaving

$$\omega^2 = \frac{eT_0}{m_i}\frac{2}{R^2}$$

i.e. the frequency depends on the sound speed $c_s = \sqrt{eT_0/m_i}$ and major radius R. This is the correct dependency for GAMs in the simple electrostatic, large aspect ratio limit when parallel flows are neglected.

6.2.3 Mass and energy conservation

For accurate and stable numerical simulations the mass and energy of the system should be conserved over long times.

Starting by multiplying the vorticity equation by ϕ

$$\phi\Omega = \phi\frac{m_i n_0}{B_0^2}\nabla_\perp^2\phi = \frac{m_i n_0}{B_0^2}\left[\nabla\cdot\left(\phi\nabla_\perp\phi\right) - \left|\nabla_\perp\phi\right|^2\right]$$

$$\begin{aligned}
\frac{\partial}{\partial t}\left(\phi\Omega\right) &= \phi\frac{\partial\Omega}{\partial t} + \Omega\frac{\partial\phi}{\partial t} && (43)\\
&= \phi\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla p + \phi\nabla_{||}j_{||} + \Omega\frac{\partial\phi}{\partial t} && (44)
\end{aligned}$$

$$\Omega\frac{\partial\phi}{\partial t} = \frac{m_i n_0}{B_0^2}\left[\nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla_\perp\phi\right) - \frac{1}{2}\frac{\partial}{\partial t}\left|\nabla_\perp\phi\right|^2\right]$$

$$
\begin{aligned}
\frac{\partial}{\partial t}\left[\frac{1}{2}\frac{m_in_0}{B_0^2}\left|\nabla_\perp\phi\right|^2\right] &= -\phi\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla p - \phi\nabla_{||}j_{||} && (45)\\
&+ \frac{m_in_0}{B_0^2}\left[\frac{\partial}{\partial t}\nabla\cdot\left(\phi\nabla_\perp\phi\right) - \nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla_\perp\phi\right)\right] && (46)
\end{aligned}
$$

This equation describes the change of $E \times B$ kinetic energy. The first line (eq 45) contains transfer terms from other forms of energy, whilst the second term describes fluxes from the boundaries. By setting either $\phi = 0$ or $\nabla_\perp\phi\cdot\mathbf{S} = 0$ at the boundary the fluxes go to zero at the boundary.

Many choices for the parallel current are possible, but a simple form is

$$
E_{||} = -\partial_{||}\phi = \eta j_{||} \qquad (47)
$$

where η is the resistivity. In this case

$$
\begin{aligned}
\nabla_{||}\left(\phi j_{||}\right) &= \phi\nabla_{||}j_{||} + j_{||}\partial_{||}\phi && (48)\\
&= \phi\nabla_{||}j_{||} - \eta j_{||}^2 && (49)\\
-\phi\nabla_{||}j_{||} &= -\eta j_{||}^2 - \nabla_{||}\left(\phi j_{||}\right) && (50)
\end{aligned}
$$

and so this term is always a sink of energy, and boundary fluxes go to zero if $\phi = 0$ or $j_{||} = 0$ at the boundary.

The curvature transfer term can be derived by starting from

$$
\nabla\cdot\left(\phi p\frac{2}{B}\mathbf{b}\times\kappa\right) = \phi\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla p + p\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla\phi + p\phi\nabla\cdot\left(\frac{2}{B}\mathbf{b}\times\kappa\right) \qquad (51)
$$

$$
\begin{aligned}
\frac{\partial}{\partial t}\left[\frac{1}{2}\frac{m_in_0}{B_0^2}\left|\nabla_\perp\phi\right|^2\right] &= p\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla\phi - \eta j_{||}^2 && (52)\\
&+ \frac{m_in_0}{B_0^2}\left[\frac{\partial}{\partial t}\nabla\cdot\left(\phi\nabla_\perp\phi\right) - \nabla\cdot\left(\frac{\partial\phi}{\partial t}\nabla_\perp\phi\right)\right] && (53)\\
&- \nabla\cdot\left(\phi p\frac{2}{B}\mathbf{b}\times\kappa\right) - \nabla_{||}\left(\phi j_{||}\right) && (54)\\
&+ p\phi\nabla\cdot\left(\frac{2}{B}\mathbf{b}\times\kappa\right) && (55)
\end{aligned}
$$

Multiplying equation 35 by the constant eT_0 we get

$$\frac{\partial p}{\partial t} = -p\frac{2}{B}\mathbf{b}\times\kappa\cdot\nabla\phi \tag{56}$$

It can be seen that this term balances the first term on the right of equation 52. The total energy:

$$E = \int_V dV\left[\frac{1}{2}\frac{m_i n_0}{B_0^2}\left|\nabla_\perp\phi\right|^2 + p\right] \tag{57}$$

is conserved (apart from resistive losses) so long as the terms in equations 53, 54 and 55 vanish.

1. Equations 53 and 54 are divergences, and all go to zero at the boundaries if $\phi = 0$ at the boundary.

2. Equation 55 is not a divergence, so can produce sources and sinks of energy in the domain, not just at the boundary. Since p and ϕ can be arbitrary functions, the curvature vector must satisfy

$$\nabla\cdot\left(\frac{2}{B}\mathbf{b}\times\kappa\right) = 0 \tag{58}$$

   The toroidal component of this curvature vector $\frac{2}{B}\mathbf{b}\times\kappa$ doesn't affect the conservation properties; the radial x component is essential for the GAM. Either the x component has to be constant, or the y component must also be included.

7 Conclusions

A set of test cases have been described, starting from simple slabs and progressing first to more complex geometries, and then to integrating the elliptic solver as part of a time-evolving system. Criteria for correctness have been specified: l_2 and l_∞ error norms for tests with an analytic solution (manufactured in complex cases). For time-evolving problems energy conservation is a useful measure, because it has a strong connection to numerical stability and physical correctness of the result. These tests will be used in future reports to test elliptic solver implementations.

8 References

[1] BOUT++ contributors. BOUT++ manual: Field-aligned coordinates. `https://bout-dev.readthedocs.io/en/latest/user_docs/coordinates.html#id1`.

[2] BOUT++ contributors. BOUT++ manual. `https://bout-dev.readthedocs.io/`.

[3] FreeGS contributors. FreeGS: Free boundary Grad-Shafranov solver in python. `https://github.com/bendudson/freegs`.

[4] Antoine J. Cerfon and Jeffrey P. Freidberg. "One size fits all" analytic solutions to the Grad–Shafranov equation. *Phys. Plasmas*, 17:032502, 2010, doi:10.1063/1.3328818.

[5] John Omotani. Cerfon-freidberg geometry generator in python. `https://github.com/johnomotani/CerfonFreidbergGeometry`.

Excalibur-Neptune report 2047356-TN-03-2

Task 1.2 Elliptic solver implementation

Ben Dudson, Peter Hill, Ed Higgins, David Dickinson, and Steven Wright

University of York

David Moxey

University of Exeter

June 17, 2021

Contents

1 Executive summary **1**

2 Problem description **2**

3 Implementation **3**

3.1 1D solvers 4

3.2 2D solvers 5

3.3 3D solvers 6

4 Testing **6**

4.1 Correctness testing 6

4.2 Performance testing 9

5 Conclusions **11**

6 References **12**

1 Executive summary

Reference elliptic solver implementations in BOUT++ [1] are described, along with the techniques which use simplified methods to effectively precondition more complex problems. Implementations in BOUT++ include previously developed PETSc-based solvers, and a new Hypre implementation.

The results of an axisymmetric Alfvén wave test are presented, showing differences in numerical stability between direct and iterative solvers for these modes.

The Hypre-based solver has been profiled on the Lassen supercomputer [2] in collaboration with LLNL. This indicates that GPUs can result in significant speedups of the matrix solves (e.g. factor of 6.9 here, on 4 GPUs vs 40 CPUs), but that other parts of the problem setup and solve can dominate and must be

mitigated to achieve good performance.

2 Problem description

Elliptic problems appear as important elements of fluid and (gyro-)kinetic plasma models. Here we focus on the solution of the electrostatic potential ϕ, which is required in order to advance the model equations, and acts in a similar way to the stream function in incompressible fluid flow. The usual technique used to calculate ϕ in edge simulation codes is to evolve the vorticity ω in time, and at each timestep to invert a second order elliptic equation of the form:

$$\nabla \cdot \left(\frac{m_i n}{B^2} \nabla_\perp \phi \right) = \omega \tag{1}$$

where m_i is the ion mass; B is the magnetic field strength which varies in space, but typically not strongly in time; n is the plasma density, which does vary in time and space. This time variation poses a challenge for methods which require matrix assembly, because the matrix elements then change in time. In some situations and models this density is replaced by a constant. This is often called the "Boussinesq" approximation, by analogy to buoyancy-driven fluid flow.

The elliptic operator in equation 1 includes an operator $\nabla_\perp = \nabla - \mathbf{b}\mathbf{b}\cdot\nabla$ which is the component of the gradient perpendicular to the magnetic field unit vector $\mathbf{b}$. In the case where $\mathbf{b}$ is constant, equation 1 becomes a 2D problem, but in toroidal fusion systems such as tokamaks $\mathbf{b}$ varies in space (and in principle also in time). The result is that the 2D surface becomes a complicated helix which wraps around the torus. In general this helix does not close on itself, and so the 2D surface fills the 3D domain. Equation 1 is therefore a 3D problem, but one which in some cases can be approximated by a 2D system, when gradients along the magnetic field $\mathbf{b}$ are neglected.

3 Implementation

The BOUT++ code includes several solvers for potential, because these are commonly needed in the drift-reduced fluid models it is designed to solve. The system to be inverted is in general 3-dimensional, but can be simplified in many cases. There are therefore several implementations

- 3D solvers, using PETSc and Hypre. These have been developed relatively recently; the Hypre implementation has been developed in the last year.
- 2D solvers, where the derivatives of the electrostatic potential ϕ along the magnetic field are neglected. In BOUT++ the potential is solved on toroidal planes (in ψ, ζ) rather than poloidal planes. The toroidal plane requires lower resolution, and means that the geometry coefficients are constant in one dimension (toroidal angle ζ). Disadvantages include a singularity in the coordinates near the X-point. (Note that often ϕ is often used for both toroidal angle and electrostatic potential).
- 1D solvers, where the 2D domain is Fourier decomposed into toroidal modes. This leads to a set of independent complex tridiagonal systems, one for each mode, which can be solved in parallel. This is possible in an axisymmetric system, like a tokamak, if the variation of density in toroidal angle is neglected (the Boussinesq approximation).

These approximations (neglecting parallel derivatives, and density constant in toroidal angle) tend to be good for high mode-number instabilities and turbulence, but become worse at lower mode-numbers (say toroidal mode number < 5). In particular for the axisymmetric mode these approximations become poor.

For many applications the lower dimensional applications give a good approximation, and can be trivially parallelised. They therefore make good preconditioners for the more complex problems. One quite common application of this is the "Naulin method" [3], where a fast solver using the Boussinesq approximation (constant density) is used in a Picard iteration to find the solution with varying density:

$$\nabla \cdot \left(\frac{m_i n}{B^2}\nabla_\perp \phi\right) = \omega$$

$$
\begin{aligned}
n &= n_0 + \delta n \\
\nabla \cdot \left(\frac{m_i n_0}{B^2}\nabla_\perp \phi^{k+1}\right) &= \omega - \nabla \cdot \left(\frac{m_i \delta n}{B^2}\nabla_\perp \phi^{k}\right)
\end{aligned}
$$

where n_0 is constant (in a way which simplifies the solve), and k is the iteration number. Since ϕ is evolving in time, typically this iteration starts from the solution at the previous time step, and tends to converge to acceptable tolerances in a small number of iterations (2-3).

3.1 1D solvers

If the density is assumed constant in toroidal angle, then the 2D system of equations can be Fourier decomposed into toroidal modes. Each mode is coupled in the radial (ψ) direction through a second-order ODE. Using 2nd-order central differencing, this results a complex tridiagonal system for each mode. Both direct and iterative methods are implemented in BOUT++:

Direct partitioning: This is a method described in a preprint by Austin et al [T.Austin, M.Berndt, D.Moulton, preprint LA-UR-03-4149, 2004]. For each 1D system, each processor reduces its local rows by eliminating rows until there are only two rows per processor. These rows are then gathered onto one processor, solved using the fast serial Thomas algorithm, and scattered back. The solution to this reduced system is then substituted back in to solve for the remaining rows on each processor. This results in an all-to-one and a one-to-all communication pattern. Performance is improved in the BOUT++ implementation by solving for all 1D systems in parallel, and grouping the rows into larger messages. In the current implementation the reduced systems are shared between all processors, so each processor sends the rows for some systems to one processor, some to the next processor, and so on. This balances the work, but results in all-to-all communications. On Archer2 this has been found to have much worse multi-node scaling than previous machines; This is currently being investigated with Archer2 support. A possible optimisation might be to gather onto fewer processes, which would result in more load imbalance, but fewer messages.

Cyclic Reduction: This is a direct algorithm which works like a multi-level extension of the partitioning algorithm described above. This algorithm has very good theoretical scaling, with the number of communication steps scaling logarithmically with the number of processors. Joseph Parker (UKAEA) recently implemented this algorithm in BOUT++, using an MIT-licensed library by Ji-Hoon Kang (KAIST) [4].

Multigrid: Over the last year or so, Joseph Parker (UKAEA) has implemented two multigrid algorithms in BOUT++, including a novel variation which has been submitted for publication. These have shown significantly better performance on Archer2 than either of the direct solvers described above.

3.2 2D solvers

2D solvers do not rely on Fourier decomposition into toroidal modes, and so can take into account corrections for density or (in principle) geometry. Implementations include:

Naulin method: As described above, this method uses the 1D solver to correct for non-constant density iteratively.

Geometric multigrid: This solver was implemented under a EUROfusion HLST project. It works, but has been found to be hard to extend or diagnose due to insufficient documentation.

PETSc: The Portable, Extensible Toolkit for Scientific Computation [5] provides a wide range of linear (and nonlinear) solvers, and interfaces to third-party libraries. Code in BOUT++ constructs the matrix elements and passes the matrix to PETSc to solve. There are actually two separate 2D solvers: One which solves in the $\psi - \phi$ toroidal plane, and another (called LaplaceXY) which solves in the $\psi - \theta$ poloidal plane. The first can solve accurately for high toroidal mode numbers, while the second was added specifically in order to solve the axisymmetric component. PETSc provides many options, and while these have not been exhaustively tested, the highest performance has typically been obtained by having PETSc pass the matrix through to Hypre [6].

3.3 3D solvers

In general the equation for the potential (equation 1) is 3D, even though it contains only the component of ϕ perpendicular to the magnetic field. The axisymmetric (toroidal mode number $n = 0$) and high modes ($n > 5$) can be solved to acceptable accuracy by 2D approximations, but the low n ($1-5$) modes require a 3D solve. There are currently two solvers implemented in BOUT++:

PETSc: Chris MacMackin and John Omotani (UKAEA) implemented this in 2019, in the process creating a wrapper around PETSc to simplify the process of creating matrices and vectors from BOUT++ data structures.

Hypre: During this ExCALIBUR-Neptune project, Peter Hill has worked with staff at LLNL (particularly Steven Glenn), to adapt the PETSc interface to Hypre. This was needed in order to make use of the latest Hypre versions, which can run on GPUs. It also eliminated a small overhead in run time, and complexity of library dependencies, which came from passing data through PETSc. During this Neptune project we have worked with LLNL to characterise and optimise this solver for GPU performance on Lassen.

4 Testing

4.1 Correctness testing

The nature of the correctness tests were described in report 2047356-TN-02-2 (task 1.1). Here we report on the results of the Alfvèn wave test, comparing direct solvers (”Tri” and ”LaplaceXZ”, both using partitioning; see section 3.1), with an iterative solver in 2D using PETSc (”LaplaceXY”, section 3.2). This was done in both shifted circle (cbm18) geometry and DIII-D X-point geometry; the X-point geometry case is the more interesting, and reported here.

Using DIII-D equilibrium, shot 119919, with grid shown in figure 1 The Alfvén wave problem (report 1.1) is simulated for the axisymmetric modes only (toroidal mode number $n = 0$).

To compare the results with and without poloidal derivatives in LaplaceXY, the

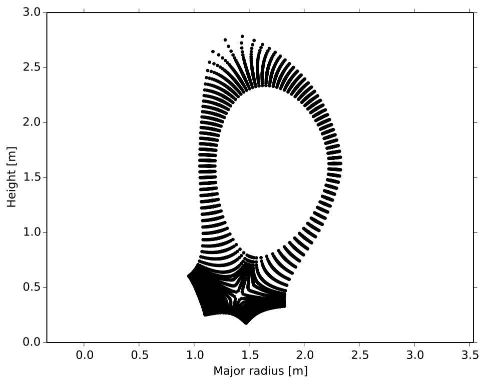

Figure 1: DIII-D 68 × 64 single null grid used for Alfvén wave test

simulation is run for a short time so that the vorticity is nearly the same in the two cases. The results in figure 2 compare the potential ϕ at the first radial cell outside separatrix ($x = 25$) as a function of poloidal cell index y for the Laplacian (X-Z) and LaplaceXY (X-Y) solvers: Over most of the domain the

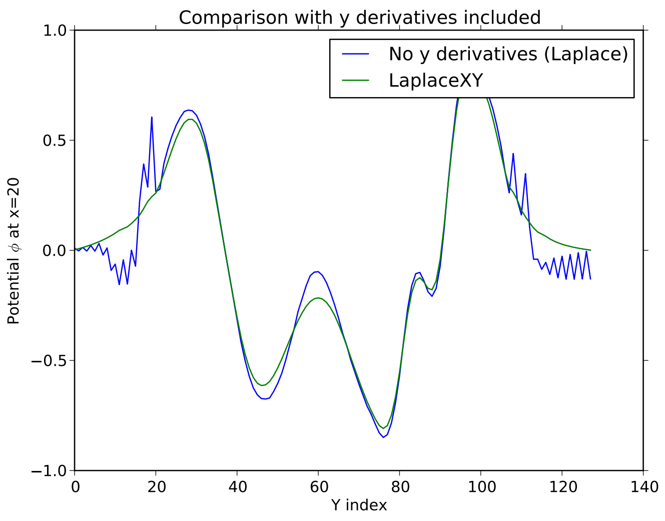

Figure 2: Electrostatic potential ϕ along the poloidal coordinate (y) just outside the separatrix. Blue line from solve neglecting poloidal derivatives; Green line from solve including poloidal derivatives.

result from the two solvers is quite close, as expected since poloidal derivatives should be small relative to radial (x) derivative terms. The X-point branch cuts at $y = 15$ and $y = 111$ are clearly seen as locations where jumps in the potential occur when the Laplacian (X-Z) solver is used (blue line). Including the poloidal derivatives acts to smooth the potential across this location (green line).

It was found that running the simulations for longer resulted in growing instabilities in some cases but not others. The system of equations does not contain a physical growing mode, so this is the result of a numerical instability. The source of this instability would initially appear to be neglecting poloidal derivatives (blue line in figure 2) but this was not the case: Iterative solvers which neglect these derivatives were found to run stably, though with more noise than simulations which include poloidal derivatives.

Figure 3 summarises the results found: It shows a comparison between the iterative (PETSc; GMRES + SOR) `LaplaceXY` solver without poloidal derivatives; the original $X - Z$ `Laplacian` (`Tri` serial implementation) direct solver; and the `LaplaceXZ` direct solver at a point in the private flux region. Note that the

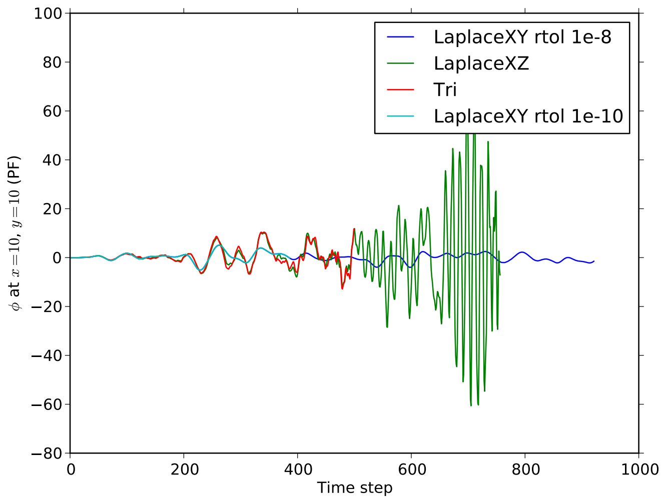

Figure 3: Comparison of solvers with poloidal derivatives removed, so only radial (x) derivatives. LaplaceXY is iterative (PETSc) while LaplaceXZ and Tri are direct; LaplaceXZ uses the same discretisation as LaplaceXY, while Tri uses a different discretisation.

direct solvers (Tri and LaplaceXZ) become unstable and have growing oscillations, while the iterative solver (LaplaceXY) remains stable. The error in the iterative method is not playing a significant role, as reducing tolerances from (rtol=1e-8, atol=1e-12) to (rtol=1e-10, atol=1e-14) makes little difference.

Conclusions drawn from these experiments are that:

- Including poloidal derivatives does not appear to be essential for the stability of $n = 0$ Alfvén wave simulations, but including them smooths the potential, and is probably important for accuracy.
- Formulating the Laplacian inversion in a conservative form (LaplaceXZ vs Tri) makes a small difference, but is probably not essential here
- There is a problem in the direct solvers for the axisymmetric component of ϕ. Direct methods tested are the serial Thomas algorithm and the direct partition algorithm. Both result in growing oscillations. This is likely because both these algorithms assume diagonal dominance, which is only marginal for axisymmetric modes when poloidal derivatives are neglected.
- The PETSc iterative solver gives very similar results to the direct solver (relative difference $\sim 10^{-6}$) on individual solves, but in the case of the direct solver this error builds up and results in a growing numerical instability.
- Preconditioning the iterative solver with a direct solver, effectively correcting the error in the direct solve, was found to remove the numerical instability.

4.2 Performance testing

Thorough performance testing of the direct and iterative tridiagonal (1D) solvers has been performed by Joseph Parker (UKAEA) on Archer 2. That work is detailed in a paper "Parallel tridiagonal matrix inversion with a hybrid multigrid–Thomas algorithm method" by J.T.Parker, P.A.Hill, D.Dickinson and B.D.Dudson, which is currently under review at the Journal of Computational and Applied Mathematics.

Performance testing of the 2D Hypre solver on Lassen has been carried out on Lassen [2] by Steven Glenn, Aaron Fisher and Holger Jones (LLNL), whom we have worked with during this project. Lassen is a 23 PFLOP system located at Lawrence Livermoe National Laboratory (LLNL), which has an IBM Power9/NVIDIA V100 architecture similar to the Sierra and Summit supercomputers, but with 40 Power9 CPUs and 4 V100 GPUs per node.

The results in figure 4 show the timing of a Hypre solver on a large 4000x4000 mesh, on either all 40 CPUS, or all 4 GPUs of a single node. Note that in the 4 GPU case, 4 CPUs are also used, to coordinate data transfers and execute parts of the code not running on the GPU.

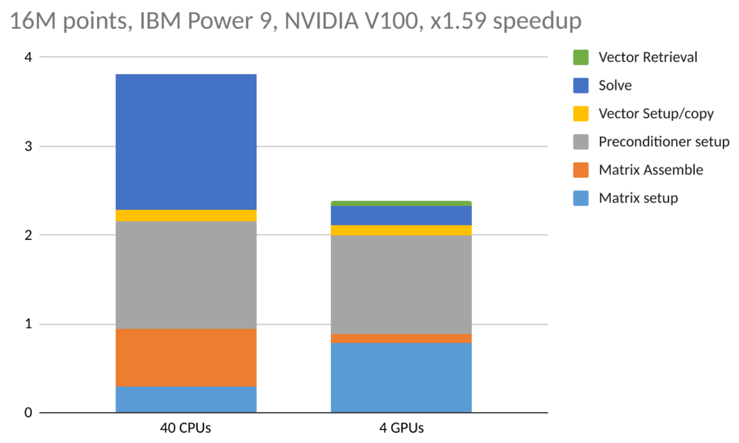

Figure 4: Timing of a Laplacian solve using Hypre on a Lasses node. Using either 40 CPUs (left) or 4 GPUs (right). Timing is broken down into stages, from matrix assembly to result vector retrieval.

Figure 4 shows that the solve time (dark blue) is 6.9 times faster on GPUs than CPUs, but is only a relatively small part of the run time (9% on GPUs; 40% on CPUs). Matrix assembly is also faster on GPUs (6x), but other parts of the solve either don't speed up (e.g preconditioner setup), or slow down (matrix setup). These steps are either not yet ported to GPUs, or require relatively complex data structure manipulation which don't tend to perform well on GPU architectures. If the matrix must be assembled every solve, this figure shows that the overall speed is only increased by a factor of 1.6.

These results indicate that matrices and preconditioners must be re-used many times, if good performance is to be achieved: Matrix and preconditioner setup takes approximately 5 times as long as vector setup, solve, and retrieval together in this case.

As discussed in section 3, iterative methods can be used to calculate corrections, to include variations in density for example. This approach could be used to improve performance, by keeping the matrix and preconditioner on the GPU fixed, and performing a small number of iterations to correct for a time-varying matrix.

5 Conclusions

Performance and correctness tests of BOUT++ elliptic solver implementations have been described. Reference implementations using PETSc and Hypre have been shown to be robust, and can make use of GPU architectures. Direct solvers based on cyclic reduction or similar tridiagonal methods can be highly efficient, but are limited to cases where the density is assumed constant (Boussinesq approximation), and can be numerically unstable under some circumstances. These deficiencies can be corrected by employing a Picard iteration to correct for non-constant density, or by using the direct solver as a preconditioner for PETSc or Hypre's iterative methods.

6 References

[1] BOUT++ contributors. BOUT++ manual. `https://bout-dev.readthedocs.io/`.

[2] Livermore Computing Center. Lassen. `https://hpc.llnl.gov/hardware/platforms/lassen`.

[3] Magnussen, M. L. Department of Physics, Technical University of Denmark. Global numerical modeling of magnetized plasma in a linear device. `https://backend.orbit.dtu.dk/ws/portalfiles/portal/133385828/`, 2017.

[4] Ji-Hoon Kang. Parallel tri-diagonal matrix solver using cyclic reduction (CR), parallel CR (PCR), and Thomas+PCR hybrid algorithm. `https://github.com/jihoonakang/parallel_tdma_cpp`.

[5] PETSc contributors. Portable, Extensible Toolkit for Scientific Computation (PETSc). , `https://www.mcs.anl.gov/petsc/`.

[6] Hypre contributors. Hypre high performance preconditioners. `https://github.com/hypre-space/hypre`.

Excalibur-Neptune report 2047356-TN-04-2

Task 2.1 1D fluid model tests

Ben Dudson, Peter Hill, Ed Higgins, David Dickinson, and Steven Wright

University of York

David Moxey

University of Exeter

June 16, 2021

Contents

1 Executive summary **1**

2 Single species **2**

2.1 Nonlinear heat conduction 2

2.2 Uniform source, outflow boundary 4

2.3 Half source, outflow boundary 5

2.4 Half source, sheath boundary 5

3 Recycling **7**

3.1 Recycling sources 8

3.2 Fluid neutral gas species 9

3.3 Kinetic neutral gas species 11

4 Multiple species **12**

4.1 Separate electron and ion temperatures 12

4.2 Neutral gas model 13

4.3 Impurities 13

5 References **14**

1 Executive summary

The deliverable for this task is to define ”Test cases for system 2-3”, a 1D fluid solver with UQ and realistic boundary conditions. The aims of this is to provide a set of problems relevant to the Neptune use case, which run quickly enough to be used as part of a fast development cycle. These will be used to compare model implementations, and test approaches to Uncertainty Quantification (UQ) and preconditioning of complex nonlinear systems involving two-way coupling between plasma dynamics, atomic reactions, and plasma-wall interactions.

2 Single species

These tests exercise individual parts of the fluid solver, excluding interactions between fluids (e.g. plasma, neutrals) which complicate the full system of equations.

2.1 Nonlinear heat conduction

Heat diffusion is a standard problem in numerical methods, but heat conduction in plasmas has some features which can introduce challenges:

- The diffusion is strongly anisotropic, many (> 6) orders of magnitude difference between direction parallel and perpendicular to the magnetic field.
- The diffusion is nonlinear: In the collisional (Spitzer-Harm / Braginskii limit), the thermal diffusion coefficient depends on the temperature $T^{5/2}$.

Some useful scenarios which are relevant to the simulation of the tokamak edge and scrape-off layer are:

1. A domain with Dirichlet boundary conditions on both sides, fixing the temperature at both sides. The equation for the heat flux is

   $$q = -\kappa_0 T^{5/2}\mathbf{b}\cdot\nabla T = \text{const}$$

   where κ_0 is a constant, calculated by integrating over a perturbed Maxwellian distribution function (see e.g Braginskii). This heat flux is along the magnetic field, which has a unit vector $\mathbf{b} = \mathbf{B}/\left|B\right|$. With a heat flux q constant along the magnetic field, the temperature $T\left(l\right)$ has an analytic solution

   $$T_0^{7/2} - T^{7/2} = \frac{7}{2}\frac{ql}{\kappa_0}$$

where l is the distance along the domain, and T_0 is a constant.

2. A fixed (low) temperature at the "target" end of the domain, and a fixed input power q in the "upstream" boundary. This power input can be implemented either as a Robin boundary condition, setting the gradient of temperature T depending on the temperature at the boundary, or as a zero-gradient (Neumann) boundary condition with a source of power over an extended region.

3. A model which captures some of the effects of impurity and atomic interactions, and allows the heat flux $q(l)$ to be a function of distance along the domain l, is a variation on the Lengyel model [1], which is widely used in divertor modelling. The main feature of this approach is that by assuming that the radiated power is a function only of temperature, an analytic solution can be obtained. The divergence of the heat flux is related to the radiation loss:
$$\nabla\cdot\mathbf{b}q \simeq \frac{dq}{dl} = -n^2Q(T)$$
where n is the plasma density, and the equality is exact in a constant magnetic field. By assuming that the pressure $nT = p$ is constant, density is also a function of temperature only.
$$q = -\kappa_0 T^{5/2}\underbrace{\frac{dT}{dl}}_{\frac{dq}{dl}\frac{dT}{dq}} = \kappa_0 T^{5/2} n^2 Q(T)\frac{dT}{dq}$$
and so
$$\int_{\text{target}}^{\text{upstream}} q\,dq = \kappa_0 p^2 \int_{\text{target}}^{\text{upstream}} T^{1/2}Q(T)\,dT$$
where the target and upstream are usually taken to be the wall and the tokamak outboard midplane or divertor entrance respectively, but can be any two points along the field line. Labelling "target" with subscript "t", and "upstream" with subscript "u":
$$q_u^2 - q_t^2 = 2\kappa_0 p^2 \int_{T_t}^{T_u} T^{1/2}Q(T)\,dT$$
By choosing an suitable cooling curve function $R(T)$, this equation can be compared to numerical solutions. One possible choice is the approximation

for nitrogen radiation used in Lipschultz 2016[2]:

$$Q = 5.9 \times 10^{-34} \frac{(T - 1\text{eV})^{1/2}\,(80\text{eV} - T)}{1 + 3.1 \times 10^{-3}\,(T - 1\text{eV})^2} \text{Wm}^3$$

for $1\text{eV} < T < 80\text{eV}$, and $Q = 0$ outside this range.

Note that the above model assumes a constant magnetic field, so that $\nabla \cdot \mathbf{b} = 0$. For conventional large aspect-ratio tokamaks this can be a reasonable approximation, but for spherical tokamaks (such as MAST-U, STEP) the variation in B has a significant impact. The Lengyel model is only a very rough guide to real experiments, but is useful here as an analytic solution to test against.

Moving beyond 1D into 2D and 3D tokamak geometry test cases: Analytic tokamak equilibria with X-points can be created based on work by Cerfon and Freidberg (a useful tool was created by John Omotani[3]. These equilibria were used in [4] to develop benchmark cases, and test a range of numerical schemes.

2.2 Uniform source, outflow boundary

Now adding fluid equations, for a single species particle density, energy (or pressure or temperature), and momentum along the magnetic field.

A 1D domain, with two boundaries:

- No-flow upstream. This can be implemented as a symmetry boundary: Zero-gradient density, pressure and temperature, and zero-value flow velocity and heat flux boundary conditions.
- Free outflow. This can be implemented in finite difference/finite volume methods by extrapolating all quantities with e.g. constant gradient. To preserve positive definite density and temperatures, it can be useful to extrapolate the logarithms of these thermodynamic quantities.

Inside the domain a uniform source of particles and (internal) energy, which then flow towards the free outflow boundary.

This system should evolve to a steady state in which the particle flux increases linearly with distance from the no-flow boundary.

2.3 Half source, outflow boundary

This is a variation on the above test, which tests the ability of the numerical scheme to resolve abrupt changes in sources, and the resulting abrupt changes in gradients.

The boundary conditions are the same as for the previous test, but the sources only fill half of the domain closest to the no-flow boundary. In this case the particle source is uniform in the first half of the domain, so the particle flux starts from zero at the no-flow boundary, and increases linearly with distance until reaching the end of the source region, and is then constant between the end of the source region and the free outflow boundary.

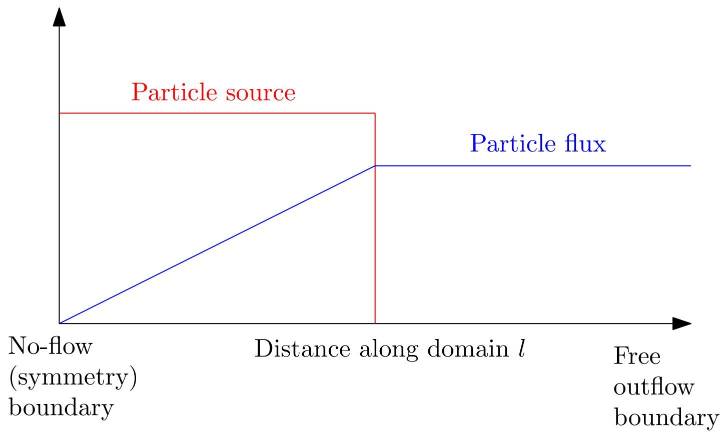

Figure 1: Sources and particle flux in ”Half source, outflow boundary” case

2.4 Half source, sheath boundary

This has the same sources as the previous case, but now tests the implementation of a sheath boundary condition. Two important components of this boundary condition in a fluid model are:

1. Imposed out-flow velocity at the sound speed, generally called the Bohm condition, or Bohm-Chodura in a magnetised plasma. The flow speed into the sheath depends in general factors such as the currents and voltages, whether species should be considered isothermal or adiabatic, and can be quite complex in situations with multiple species (see e.g. [5]). A simple and widely used approximation is the isothermal sound speed (see e.g. Stangeby):

   $$c_s = \sqrt{\frac{e\left(T_e + T_i\right)}{m_i}}$$

   where the electron and ion temperatures T_e and T_i are in units of eV, m_i is the ion mass (in kg), and the sound speed c_s is in units of m/s.

2. Heat conduction to the target. In addition to the energy loss expected because the fluid is flowing to the target, there is also typically an additional loss of energy. Physically what is happening at the target is that slow electrons are reflected, while energetic electrons reach the target, so that the electrons are rapidly cooled. Some of this energy is transferred to accelerating ions into the target, via the sheath voltage. The energy flow through the sheath is often characterised by the energy flux:

   $$q_{\text{sheath}} = \gamma_{sh} enTc_s$$

   with a different γ_{sh} for each species. The actual power flow to the target is complex, depending like the flow speed on the electric fields and currents, and the sheath heat loss coefficientγ_{sh} can vary considerably (by an order of magnitude), particularly during transients.

If solving a single fluid, i.e a single density, energy and momentum equation, a common assumption is that the ions and electrons have equal densities ($n = n_e = n_i$), temperatures ($T = T_e = T_i$) and velocities ($v = v_e = v_i$). This means that the total pressure $p = enT_e + enT_i = 2enT$. The sheath velocity is then

$$c_s = \sqrt{\frac{2eT}{m_i}}$$

In a fluid model with ratio of specific heats $\gamma = 5/3$ (typical for plasma simulations), the energy flow is

$$q_{fluid} = \left(\frac{5}{2}p + \frac{1}{2}m_iv^2\right)v + q_{cond}$$

where q_{cond} is the heat conduction. At the sheath where $v = c_s$ we have:

$$\begin{aligned} q_{fluid} &= \left(\frac{5}{2}2enT + \frac{1}{2}m_i\frac{2eT}{m_i}\right)c_s + q_{cond} \\ &= 6enTc_s + q_{cond} \end{aligned}$$

so by comparing q_{fluid} with q_{sheath}, if $\gamma_{sh} = 6$ then there should be no heat conduction through the sheath. More typical values are γ_{sh} between 6.5 and 9. The additional heat flux can be used to calculate the temperature gradient at the sheath in a Robin-type boundary, or removed from the last cell through the boundary with the target.

This more complex system still has analytic solutions, depending only on the given input power, and the upstream density.

3 Recycling

The half source, sheath boundary test case is a reasonable model of a low density plasma (what Stangeby describes as a "simple SOL"), but is missing a crucial ingredient for tokamak edge and divertor simulation: the interaction with neutral gas.

When plasma meets a material surface a sheath is formed, a small region with strong electric fields, which accelerates ions towards the surface. When ions hit the surface they tend to pick up electrons and become neutral atoms. Some of those atoms will reflect from the surface, others will stick to the surface and perhaps combine into molecules, or become embedded inside the surface before diffusing out again. The details depends on what the wall is made of (e.g. carbon or tungsten), but the key feature is that the majority of the impinging ions ($> 99\%$) will typically come back into the plasma as neutral atoms or molecules.

The atoms and molecules which come off the material surface back into the plasma typically encounter high electron temperatures, and are quickly ionised and converted into ions again. These ions then return to the wall, where they again become neutral atoms. The majority of the ions flowing to the wall are typically undergoing this *recycling* process, so that the flow of ions from upstream (regions away from the wall and divertor) are typically small relative to the flow of ions to the surface.

This recycling process of continually ionising atoms removes energy from the electrons, both in overcoming the ionisation potential, and also in radiation from the relaxation of intermediate excited states of the atoms. This radiation can also be critical to reducing heat flows to the surfaces, and is enhanced by a feedback mechanism: As the plasma (electrons) cool, the radiated energy per ionisation rapidly increases at low temperatures (< 5eV). This can lead to the plasma "detaching" from the wall, reducing the heat and particle fluxes, in a process similar to the condensation instability seen in space plasmas.

3.1 Recycling sources

A straightforward modification to the 1D single fluid model used in section 2, is to add sources to represent the recycled flux of particles.

- The power source is kept away from the "target", to represent the flow of heat from the main plasma.
- Rather than the particle source being at the same location as the power source, the particle source is now put close to the target.

A reasonable choice of particle source is an exponential decay, with highest source at the target. This is to represent the mean free path of neutral atoms entering the plasma. It's important to note that the decay length for this source should be significantly ($10 - 100$ times) longer in the direction along the magnetic field than the ionisation mean free path. This is because neutral atoms are travelling away from the wall, but the magnetic field is at a shallow angle (typically a few degrees) to the wall. Neutral atoms may only travel a short distance from the wall, but this can correspond to a relatively long distance along the magnetic field in the 1D domain simulated here.

To represent the loss of energy from ionisation and the associated excitation radiation, a sink of energy proportional to the particle source can be added. The ionisation potential for hydrogen atoms is 13.6eV, and dissociation potential for molecules about 4.5eV per molecule, but the energy lost per ionisation varies significantly: Neutral species can enter the plasma with some energy (e.g Franck-Condon energy, around 3.5eV but with wide variation), lowering the effective energy cost; as discussed above, excitation radiation increases the effective energy loss per ionisation. Typical values used in the literature are around 30eV, which is probably a reasonable value to use for this test case.

With this fixed particle source (and so fixed power sink), some care should be taken that the temperature is not driven to zero by removing more power than is put in. This is quite straightforward if a fixed energy cost per ionisation is used, as then the total power sink can be balanced against the power input. If a temperature-dependent ionisation is used, however, then this can be more difficult.

3.2 Fluid neutral gas species

A more sophisticated neutral gas model represents the interaction of plasma and gas as two fluid species, one for the plasma and one for the neutral atoms. Further fluid species can also be added to represent molecules, short-lived species (eg H^-, H_3^+) or a number of vibrationally excited states. The network of reactions between these species can become quite complicated, even for something as "simple" as hydrogen.

Evolving neutral gas as a species modifies the reactions: The ionisation source of plasma particles is now evolving with the system state, rather than being imposed as an input parameter. The volumetric source S, in units of particles per second per cubic meter is

$$S = n_e n_a \left<\sigma v\right>_{iz}$$

where n_e is the number density of electrons, n_a the number density of neutral atoms, and $\left<\sigma v\right>_{iz}$ is the reaction rate in units of m^3s^{-1}. This rate is averaged over a Maxwellian distribution, and is typically derived from a 0-D collisional-radiative model, as an effective rate which averages over a number of processes. These are tabulated in several databases; a commonly used one in plasma physics is Amjuel [6], used in the EIRENE monte-carlo code[1]. Reaction 2.1.5FJ in Amjuel describes ionisation of neutral atoms by electrons:

$$e + H(1s) \rightarrow e + H^+ + e$$

In addition to the ionisation, other reactions now need to be included. Most important is charge exchange (CX) between plasma ions and neutrals. In this reaction, the electron in a neutral atom swaps to the ion, represented as:

$$p + H(1s) \rightarrow p + H(1s)$$

(Amjuel reaction 0.1T). The left and right side of this equation are the same because the reaction swaps an ion and neutral atom but doesn't affect the number of particles of each species. This reaction often has a higher likelihood than ionisation, and provides a strong coupling between the plasma and neutral species. There is no net source or sink of particles for the plasma, but there is a source or sink of energy (equation 1) and momentum (equation 2), depending on the relative temperatures and flow velocities of the plasma ions (subscript 'i') and neutral atoms (subscript 'a'):

$$S_{\text{energy}} = \frac{3}{2}e\left(T_a - T_i\right) n_i n_a \left<\sigma v\right>_{cx} \quad (1)$$

$$S_{\text{momentum}} = m_i\left(v_a - v_i\right) n_i n_a \left<\sigma v\right>_{cx} \quad (2)$$

Once this reaction is included, the strong interaction between the plasma and neutral atom flows will tend to force the atoms into a very narrow layer close to the target, which can be challenging to resolve. Cross-field diffusion of neutral atoms (which can freely move across the magnetic field, unlike the plasma ions), together with refinement of the grid resolution near the target, are typically needed to widen and resolve this layer.

The handling of charge exchange reactions in a fluid model is problematic: Charge exchange produces a population of fast-moving neutral atoms with a quite different energy and momentum distribution to the atoms from the wall. Unless these atoms are strongly coupled to each other, which is rarely the case

[1]The EIRENE Fortran code reads the data tables from the Amjuel LaTeX source file.

in tokamaks except in specific high density locations, then a single fluid approximation is unlikely to be good. Occasionally in the literature another fluid species is used to represent the charge exchanged neutrals. In 1D there is also the challenge of how the transport of neutrals, particularly charge-exchanged fast neutrals, across the plasma should be represented: The plasma is not really a 1D tube, but a relatively thin sheet, from which neutrals can escape, carrying momentum and energy with them. Representing this, and the resulting interactions with in-vessel components like walls, baffles and pumps, requires at least a 2D representation.

3.3 Kinetic neutral gas species

In most of the tokamak the ratio of the mean free path to the local length-scales of density, temperature etc (the Knudsen number) is much larger than 1, putting the transport in the regime of rarefied gas dynamics, and implying that a kinetic (not fluid) treatment is needed. Unfortunately in other regions, typically near the target, the Knudsen number can be much less than 1, so that kinetic models become highly inefficient (because they're simulating a fluid situation). There are several kinetic plasma neutral models, EIRENE probably being the most well known, and several efforts ongoing to develop hybrid fluid-kinetic models.

Coupling a 1D plasma model to EIRENE would be a non-trivial task, not least because EIRENE is not publicly available or open source in any meaningful way[2]. It is likely that examining the SOLPS code, and benchmarking against SOLPS, would be the most direct way to achieve this.

The main applications of such a 1D EIRENE coupling would be to compare against the fluid model; to use it as a platform for studying model order reduction of the neutral model to simpler and faster models; to test preconditioning strategies; uncertainty quantification involving coupled fluid-monte-carlo coupled algorithms; and to gain experience of carrying out the coupling, to apply to more complex 2D and 3D models. It is likely however that there are quicker, more direct ways of achieving these goals, such as using SOLPS directly.

[2]Access to EIRENE is in principle available from Juelich on request

4 Multiple species

Real plasmas of interest are not pure hydrogen, but contain a mixture of different species. A reactor will have deuterium and tritium isotopes, helium, wall materials such as tungsten and beryllium, seeded impurities such as argon, and trace impurities of e.g. oxygen. Not all of these need to be simulated in all cases, but the capability to model multiple species will become increasingly essential.

4.1 Separate electron and ion temperatures

The most important, but also probably simplest, variation on the models described in previous sections, is to separate the ion and electron temperatures. Close to the target, the low temperatures and high densities often lead to a strong coupling between electron and ion temperatures, but upstream this coupling becomes weaker: Typical measurements on present-day tokamaks find ion temperatures around a factor of two higher than electron temperatures around the outboard midplane.

Separating out the ion and electron temperatures involves:

- Evolving separate temperature or (internal) energy equations for the ions and electrons. The density and velocity equations are still the same, due to quasineutrality of the plasma and the absence of net currents in 1D simulations.
- Using different heat conduction coefficients for electrons and ions
- Separate sheath heat transmission coefficients (γ_{sh}, section 2.4)
- Carefully tracking the contribution of reactions to the energy balance of each species: Ionisation, for example, is an energy loss for electrons, but an energy gain for the ions, as energy is transferred from atoms into ions.
- Adding a coupling between electrons and ions (see Braginskii), due to collisions between them.

4.2 Neutral gas model

As mentioned in section 3.2, the plasma chemistry of even pure hydrogen plasmas can be quite complicated: There are a number of different pathways by which a ionisation (for example) can occur, some involving catalytic interactions between molecules, atoms and plasma ions. Atomic reaction rates are sensitive to the starting state of the molecule (e.g. high vibrational states are more likely to dissociate) or atom (highly excited states are more likely to ionise). In some cases it is likely that these states can persist for long enough (metastable states) that they can be transported around the domain and should be tracked as separate species. It is currently not well understood in the plasma community how complex the model needs to be, and what errors are made when simplified models are used.

4.3 Impurities

Finally, more complex models can be built to study the transport of multiple species, and their multiple charge states. For low-Z species each charge state is typically evolved as a separate species, with the reactions between them (typically ionisation, recombination, and charge exchange with hydrogen); higher Z materials like tungsten often need a charge-state "bundling" treatment, where a range of charge states is evolved as a single fluid, with effective reaction rates between bundled states. The model reduction needed to do this accurately and efficiently is an active area of research.

Once multiple species are included, particularly where their masses are comparable to each other, and where their concentrations are not "trace" level, the calculation of collisions between species becomes complicated. Hirshman and Sigmar published models; Zhdanov is a well known model; and there has been some work recently on improving these models and implementing them in simulation codes.

Unfortunately once models become this complex, analytic test cases can no longer be found. Instead the best option is probably a Method of Manufactured Solutions (MMS) test.

5 References

[1] L L Lengyel. Analysis of Radiating Plasma Boundary Layers (IPP 1/191). `http://hdl.handle.net/11858/00-001M-0000-0027-6A04-1`.

[2] Bruce Lipschultz, Felix I. Parra, and Ian H. Hutchinson. Sensitivity of detachment extent to magnetic configuration and external parameters. *Nucl. Fusion*, 56:056007, 2016, doi:10.1088/0029-5515/56/5/056007.

[3] John Omotani. Cerfon-freidberg geometry generator in python. `https://github.com/johnomotani/CerfonFreidbergGeometry`.

[4] M.Held, M.Wiesenberger, and A.Stegmeir. Three discontinuous Galerkin schemes for the anisotropic heat conduction equation on non-aligned grids. *Comp. Phys. Comm.*, 199:29–39, 2016, doi:10.1016/j.cpc.2015.10.009.

[5] D.Tskhakay and S.Kuhn. Boundary conditions for the multi-ion magnetized plasma-wall transition. *J. Nucl. Mat.*, 337–339:405–409, 2005, doi:10.1016/j.jnucmat.2004.10.073.

[6] D. Reiter. The data file AMJUEL:Additional Atomic and Molecular Data for EIRENE. `http://www.eirene.de/html/amjuel.html`, 2020.